FUNDAMENTOS DE INFORMÁTICA

PROFESORA: ING. SILVIA PATRICIA BARDELLI

INGENIERA SILVIA PATRICIA BARDELLI

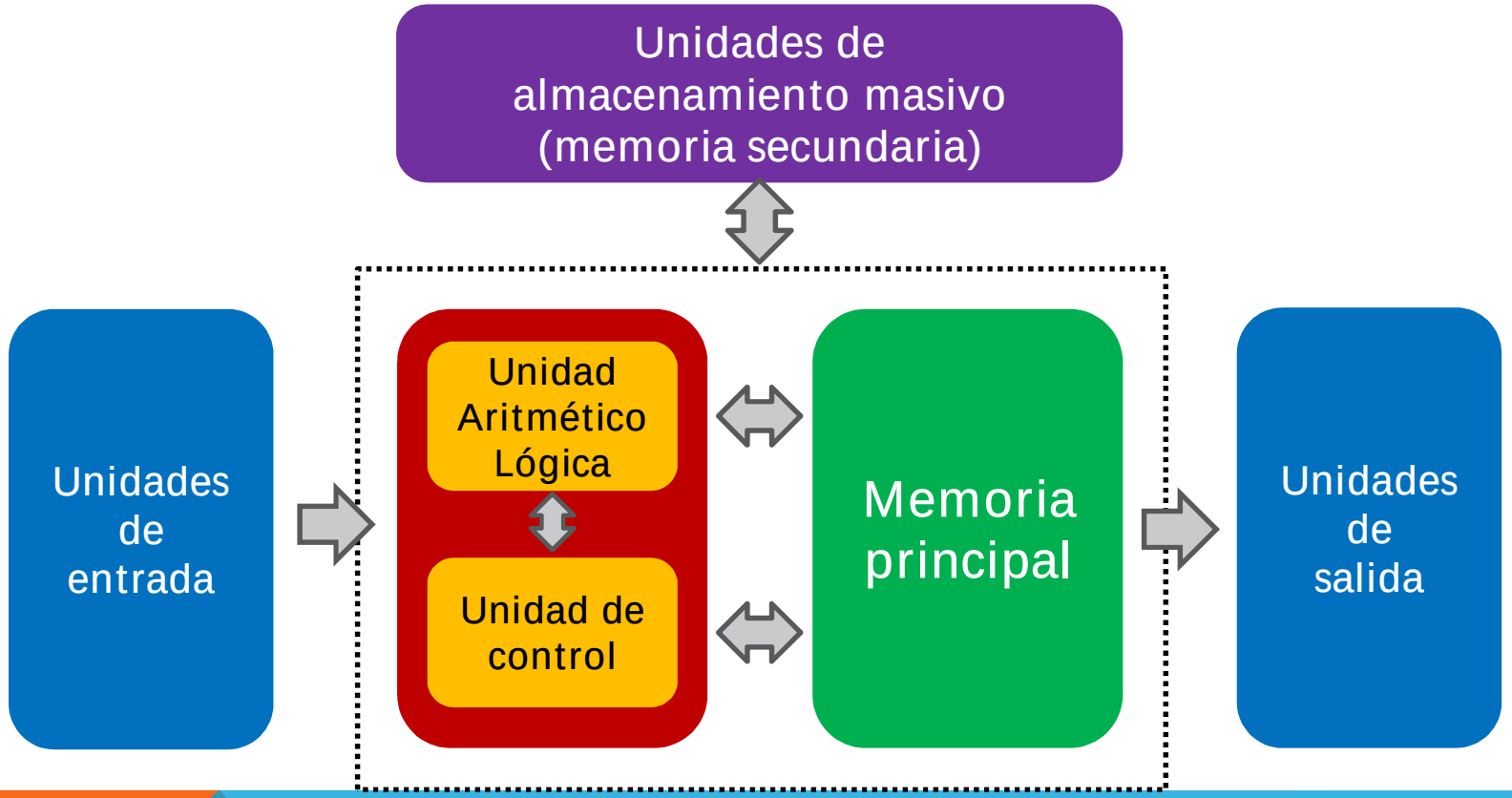
CLASE 1

INGENIERA SILVIA PATRICIA BARDELLI

CLASE 1

Temas:

- Repaso general


Unidades de almacenamiento masivo (memoria secundaria)
Unidades de entrada
Unidad Aritmético Lógica
Unidad de control
Memoria principal
Unidades de salida

CONCEPTOS FUNDAMENTALES

Algoritmo:

Es una secuencia finita y repetible de pasos que describe el proceso a seguir para solucionar un problema dado.

CONCEPTOS FUNDAMENTALES

- Secuencia: Significa que los pasos están ordenados.
- Finita: Que tiene un final.
- Repetible: Partiendo de las mismas condiciones iniciales, el resultado debe ser siempre el mismo.

CONCEPTOS FUNDAMENTALES

Programa:

Es la implementación de un algoritmo en algún lenguaje de programación.

CONCEPTOS FUNDAMENTALES

Lenguaje de Programación:

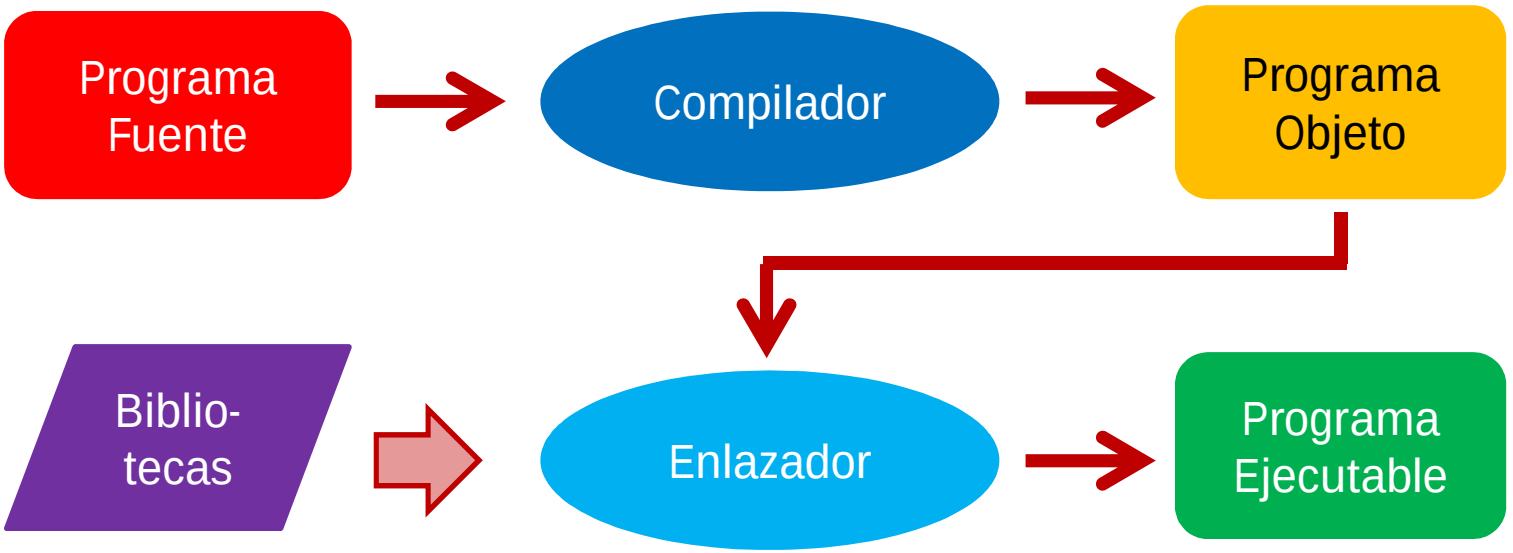
Es un lenguaje formal compuesto por un conjunto de símbolos y reglas sintácticas que se utiliza para darle instrucciones a una computadora.

TRADUCTORES DE LENGUAJES

Compilador:

- Convierte el ***programa fuente***, escrito en un lenguaje de alto nivel, en un ***programa objeto***.
- Este programa objeto debe ser ***enlazado*** o ***vinculado*** con bibliotecas proporcionadas por el fabricante, creando así el ***programa ejecutable***.

PROCESO DE COMPILACIÓN

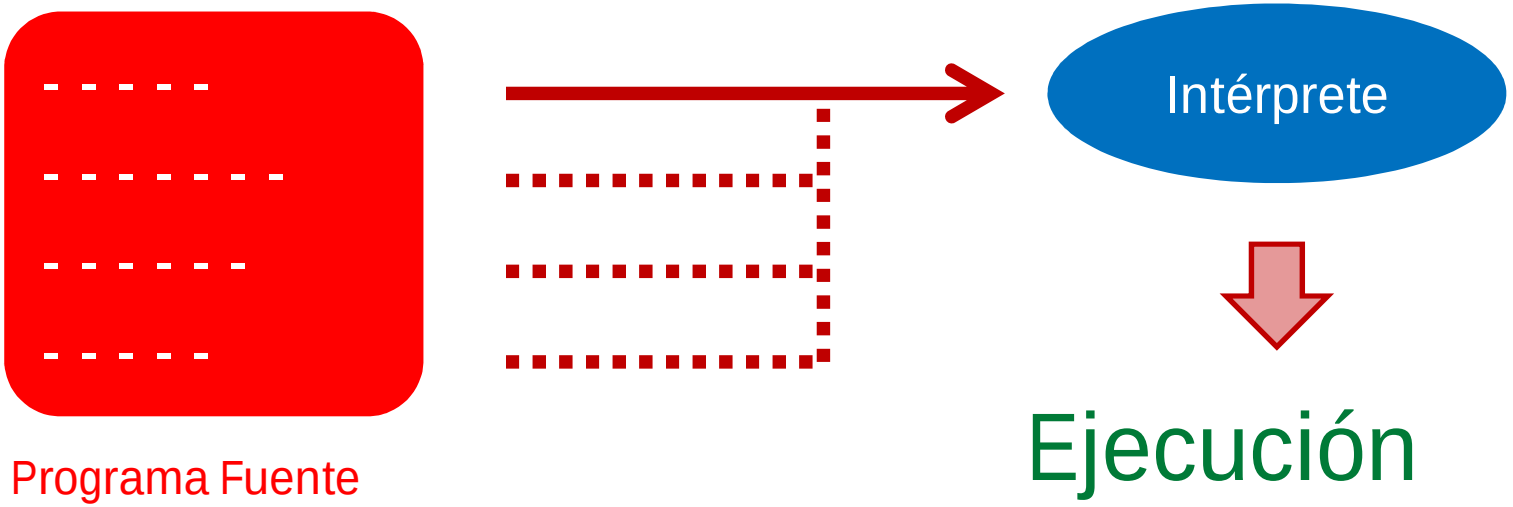


TRADUCTORES DE LENGUAJES

Intérprete:

- Traduce una por una las líneas del programa fuente, ejecutándolas inmediatamente.

PROCESO DE INTERPRETACIÓN



COMPILADOR O INTÉRPRETE?

Compilador:

- Realiza la traducción en tiempo de desarrollo, es decir cuando el programa aún no se está ejecutando.
- Se compila UNA vez.
- Se ejecuta MUCHAS veces.

COMPILADOR O INTÉRPRETE?

Intérprete:

Realiza la traducción a medida que el programa se está ejecutando.

Si una instrucción se repite 1000 veces, será traducida 1000 veces.

COCINAR UNA HAMBURGUESA

- Encender el fuego de la hornalla.
- Colocar la plancha sobre él.
- Colocar la hamburguesa sobre la plancha o parrilla.

1. Esperar un momento

- Determinar si está cocida del lado inferior.
- Sí, continuar
- No, ir a 1
- Dar vuelta la hamburguesa

2. Esperar un momento

- Determinar si está cocida del lado inferior
- Sí, fin
- No, ir a 2

CLASE 2

INGENIERA SILVIA PATRICIA BARDELLI

Variables

Reglas para crear nombres de variables:

- §Sólo letras, números y el guión bajo.
- §No pueden comenzar con un número.
- §No pueden coincidir con las palabras reservadas del lenguaje.

Variables

```
a = 3                    # Asignación de constante
b = a                    # Asignación de variable
c = a + b + 1            # Asignación de expresión
pi = 3.1416
print("La variable a contiene", a, "b contiene", b, "y c
contiene", c)
print("La variable pi contiene", pi)
```

Variables

+ Suma

* Multiplicación

// División entera

** Potenciación

- Resta

/ División real

% Módulo o resto

Ingreso de datos por teclado

```
n = input("Ingrese un número entero: ")
n = int(n)
```

-- - o también - - -

```
n = int(input("Ingrese un numero entero: "))
```

Además existe la función float( ) para poder ingresar números reales.

Ejemplo

```
# Leer dos números enteros y guardarlos en dos variables.
# Luego intercambiar sus valores e imprimir su contenido.

a = int(input("Ingrese un número entero: "))
b = int(input("Ingrese otro número entero: "))
print("A contiene", a, "y B contiene", b)
c = a
a = b
b = c
print("Ahora A contiene", a, "y B contiene", b)
```

CLASE 3

INGENIERA SILVIA PATRICIA BARDELLI

Estructura Secuencial

- Trabajando de esta manera las posibilidades de resolución de problemas son limitadas. Sólo cálculos e impresiones.

- Para que un programa sea ***realmente útil*** es necesario que sea capaz de ***tomar decisiones*** y actuar en consecuencia.

Estructura Alternativa o Condicional

- Por eso, además de la estructura secuencial, existen dos estructuras más en el mundo de la Programación Estructurada.

- La segunda que veremos se denomina ***Estructura Alternativa o Condicional***.

Instrucción if

Formato 1

if ***<condición>***:

. . . . . .

. . . . . .

. . . . . .

Ejemplo Nro 1

```
# Leer un número entero e imprimir un
    # mensaje indicando si es mayor que 5.
    n = int(input("Ingrese un número: "))
    if n > 5:
        print("El número es mayor que 5")
    # Fin del programa
```

Instrucción if

Formato 2

```
if <condición>:
        . . . . . .
        . . . . . .
else:
        . . . . . .
        . . . . . .
```

Ejemplo Nro 2

```
# Leer la calificación que obtuvo un alumno en un
# examen final e imprimir un mensaje indicando si
# aprobó o no la materia. Se aprueba con 4.
nota = int(input("Ingrese la calificación: "))
if nota >= 4:
    print("El alumno aprobó la materia")
else:
    print("El alumno no aprobó la materia")
```

Instrucción if

Formato 3

```
if <condición>:
        . . . . . .
elif <condición>:
        . . . . . .
else:
        . . . . . .
```

Ejemplo Nro 3

```
# Leer un número e informar si es positivo, negativo o cero.
n = int(input("Ingrese un número entero: "))
if n > 0:
    print("El número es positivo")
elif n < 0:
    print("El número es negativo")
else:
    print("El número es cero")
```

Operadores Lógicos

Operador ***and (Y)***:

Cond. 1	Cond. 2	Cond. 1 and Cond. 2
V	V	V
V	F	F
F	V	F
F	F	F

Operadores Lógicos

Operador ***or (O)***:

Cond. 1	Cond. 2	Cond. 1 or Cond. 2
V	V	V
V	F	V
F	V	V
F	F	F

Operadores Lógicos

Operador *not (NO)*:

Condición	not Condición
V	F
F	V

Ejemplo Nro 4

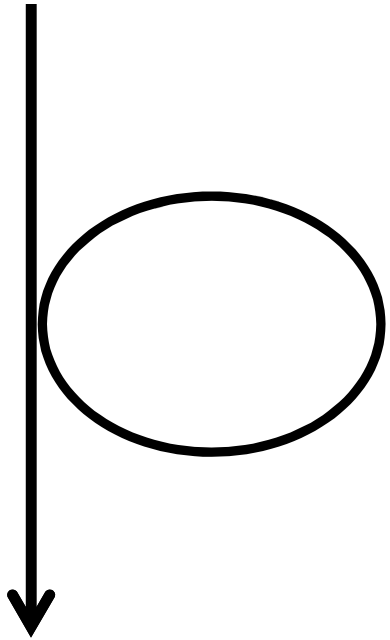
```
# Leer un número entero e imprimir un mensaje indicando
# si corresponde a un número válido de mes.
mes = int(input("Ingrese un número de mes: "))
if mes >= 1 and mes <= 12:
    print("El mes es válido")
else:
    print("El mes es inválido")
```

CLASE 4

INGENIERA SILVIA PATRICIA BARDELLI

ESTRUCTURA ITERATIVA, CICLO O BUCLE

En esta estructura el programa repetirá una porción de su código una cierta cantidad de veces, y luego seguirá adelante.



INSTRUCCIÓN WHILE

while ***<condición>***:

. . . . . .

. . . . . .

. . . . . .

EJEMPLO

```
# Imprimir los números enteros entre 1 y 100

a = 1
while a <= 100:
    print(a)
    a = a + 1
# Fin del programa
```

DEFINICION CONTADOR

Cuando una variable es modificada en una cantidad fija respecto de su valor anterior, se la denomina ***contador***.

ATENCIÓN

¿Qué ocurre si olvidamos la línea que incrementa el contador?

```
a = 1
while a <= 100:
    print(a)
    a = a + 1
# Fin del programa
```

EJEMPLO

```
suma = 0
cant = 0
n = int(input("Ingrese un número o -1 para terminar: "))
while n != -1:
    suma = suma + n
    cant = cant + 1
    n = int(input("Ingrese un número o -1 para terminar: "))
if cant != 0:
    prom = suma/cant
    print("El promedio es", prom)
else:
    print("No se ingresaron valores")
```

DEFINICIÓN ACUMULADOR

Cuando una variable es modificada en una cantidad cambiante respecto de su valor anterior, se la denomina ***acumulador***.

EJEMPLO

Objetivo:

Leer un conjunto de números enteros e imprimir el mayor. El fin de los datos se indica con -1.

¿Cómo podemos proceder para hallar el máximo?

EJEMPLO

```
n = int(input("Ingrese un número o -1 para terminar: "))
mayor = n
while n != -1:
    if n > mayor:
        mayor = n
    n = int(input("Ingrese un número o -1 para terminar: "))
print("El mayor es", mayor)
```

CLASE 5

INGENIERA SILVIA PATRICIA BARDELLI

FUNCIONES

Una función es un módulo independiente de programa que realiza una tarea específica.

PROFESORA ING. SILVIA PATRICIA BARDELLI

EJEMPLO 1: CÁLCULO DE UN PROMEDIO

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total

# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado)
```

VENTAJAS DE TRABAJAR CON FUNCIONES

1. Es posible dividir el programa en módulos más pequeños, para que cada función realice una tarea concreta. Esto facilita el desarrollo y la comprensión del programa.

VENTAJAS DE TRABAJAR CON FUNCIONES

2. Las funciones pueden invocarse muchas veces dentro del mismo programa, evitando la reiteración innecesaria de código.

VENTAJAS DE TRABAJAR CON FUNCIONES

3. Una misma función puede ser utilizada en otros programas, evitando tener que volver a escribir código ya escrito.

TIPOS DE PARÁMETROS

- Existen parámetros ***formales*** y parámetros ***reales***.
- Los formales son los que se escriben en el encabezado de la función.

PARÁMETROS

- Los parámetros reales son los que se escriben en la llamada o invocación.
- Pueden llevar el mismo nombre o nombres diferentes.

TIPOS DE PARÁMETROS

- Existen parámetros ***formales*** y parámetros ***reales***.
- Los formales son los que se escriben en el encabezado de la función.

PARÁMETROS

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total
```

Parámetros formales a y b

Parámetros reales X e y

```
# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado)
```

ÁMBITO DE LAS VARIABLES

total sólo puede usarse dentro de la función

x e y sólo pueden usarse en el programa principal

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total

# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado
```

IMPORTANTE

Las funciones deben escribirse antes que el programa principal.

Ni durante, ni después.

El programa principal debe ubicarse siempre al final del código.

EJEMPLO 3

Objetivo:

Escribir una función que reciba como parámetros la fecha de nacimiento de una persona y la fecha actual, y devuelva la edad de la persona.

PROGRAMA COMPLETO

```
def leerentero(min, max):
    print("Ingrese un número entre entre", min, "y", max, end="")
    a = int(input())   # ¿Por qué el mensaje no se muestra aquí?
    while a<min or a>max:
        print("Valor incorrecto. Debe estar entre", min, "y", max)
        a=int(input("Ingrese un numero entero: "))
    return a

def calcularedad(dnac, mnac, anac, dact, mact, aact):
    edad = aact - anac
    if mact < mnac:
        edad = edad - 1
    elif mact == mnac and dact < dnac:
        edad = edad - 1
    return edad
```

```
def leerfecha( ):
    print("Dia? ")
    dia = leerentero(1, 31)
    print("Mes? ")
    mes = leerentero(1, 12)
    print("Año? ")
    año = leerentero(1583,2100)
    return dia, mes, año

# Programa principal
print("Ingrese su fecha de nacimiento:")
dnac,mnac,anac = leerfecha()
print("Ingrese la fecha de hoy:")
dhoy,mhoy,ahoy = leerfecha()
edad=calcularedad(dnac,mnac,anac,dhoy,mhoy,ahoy)
print("Ud. tiene",edad,"años")
```

IMPORTANTE

```
def esprimo(n):
    divisor = 2
    while divisor < n:
        if n % divisor == 0:
            return False
        divisor = divisor + 1
    return True
# Programa principal
num = int(input("Ingrese un número
entero: "))
if esprimo(num):
    print(num, "es primo")
else:
    print(num, "no es primo")
```

Se considera una ***mala práctica de programación*** colocar un return dentro de un ciclo.

BIEN!!

```
def esprimo(n):
    primo = True
    divisor = 2
    while divisor < n:
        if n%divisor==0:
            primo = False
        divisor = divisor + 1
    return primo

# Programa principal
num = int(input("Ingrese un número entero: "))
if esprimo(num):
    print(num, "es primo")
else:
    print(num, "no es primo")
```

CLASE 6

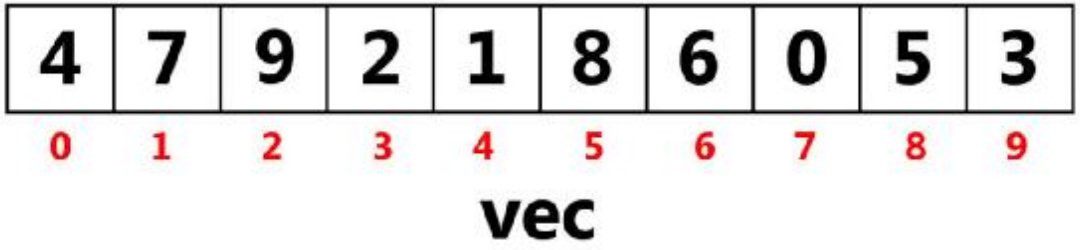
INGENIERA SILVIA PATRICIA BARDELLI

INTRODUCCIÓN

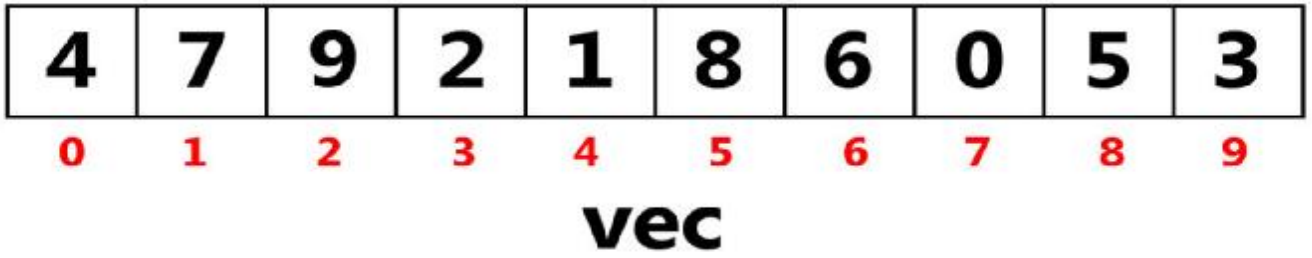
```
a = int(input("Ingrese un número: "))
b = int(input("Ingrese otro número: "))
c = int(input("Y otro más: "))
print(c, b, a)
```

ARREGLOS

Para eso se utiliza un ***subíndice***, que identifica la posición de cada variable. Equivale al ***número de orden*** de la variable dentro del arreglo.



SUBÍNDICES




```
print(vec[2])       # Constante
a = 4
print(vec[a])       # Variable
print(vec[a+3])     # Expresión
```

EJEMPLO

```
# Leer 100 números e imprimirlos en orden inverso
vec = [ ]
i = 0
while i < 100:
   n = int(input("Ingrese un número: "))
   vec.append(n)
   i = i + 1
# Impresión
i = 99
while i >= 0:
   print(vec[i])
   i = i - 1
```

RESOLUCIÓN DEL EJEMPLO NRO 2

```
# Imprimir una lista por pantalla
lista = [4, 7, 2, 9, 5]
x = 0
while x < 5:
    print(lista[x], end=" ")
    x = x + 1
```

EJEMPLO NRO 4

Objetivo:

Leer un conjunto de números y guardarlos en una lista, finalizando la carga con -1. Luego buscar el mayor elemento leído, mostrarlo y eliminarlo del arreglo.

Imprimir por pantalla la lista antes y después del borrado.

```
def imprimirlista(vec):
    largo = len(vec)
    for i in range(largo):
       print(vec[i],end=" ")
    print()


# Programa principal
v = [ ]
n = int(input("Ingrese un numero o -1 para terminar: "))
while n != -1:
    v.append(n)
    n = int(input("Ingrese un numero o -1 para terminar: "))
largo = len(v)
if largo == 0:
    print("No se ingresaron valores")
else:
    mayor = v[0]
    pos = 0
    for i in range(largo):
        if v[i] > mayor:
            mayor = v[i]
            pos = i

  imprimirlista(v)
    print("El máximo es",mayor,"y se encontró en la posición",pos)
    print("Borrando el",mayor)
    del v[pos]
    imprimirlista(v)
```

NOVEDADES DEL EJEMPLO

Instrucción del:

Permite borrar elementos de una lista.
También sirve para eliminar variables o listas completas.

```
del x      # Borra la variable x
del lista[4]      # Borra el elemento de la posición 4
del lista         # Borra la lista completa
```

EJEMPLO:

Uso de for y range( )

```
# Imprimir los números del 1 al 100
for numero in range(1, 101):
    print(numero, end=" ")
```

Cuando ***range( )*** se usa con dos parámetros el primero indica el inicio de la secuencia, mientras que el segundo señala el final de la misma. El valor final no está incluido.

EJEMPLO

range( ) con incremento:

```
# Imprimir los números impares del 1 al 100
for impar in range(1, 101, 2):
    print(impar, end=" ")
```

Cuando ***range( )*** se usa con tres parámetros, el tercero actúa como ***incremento***. Con sólo dos parámetros el incremento es 1.

EJEMPLO:

```
# Imprimir los números del 100 al 1
for i in range(100, 0, -1):
    print(i, end=" ")
```

Cuando el incremento es negativo el valor inicial debe ser mayor que el valor final.

CLASE 7

INGENIERA SILVIA PATRICIA BARDELLI

EJEMPLO

```
import random

def lanzardado( ):
    return random.randint(1, 6)

# Programa principal
dado = lanzardado( )
print(dado)
```

NOVEDADES

Importación de módulos

Todo módulo que desee utilizarse debe ser ***importado*** (incluido) al comienzo del programa.

Función randint(mínimo, máximo)

Genera un número entero al azar entre los límites suministrados, ambos incluidos. El nombre de la función debe ir precedido por el del módulo, separados por un punto.

BÚSQUEDA SECUENCIAL

- La búsqueda secuencial es la más sencilla de las búsquedas que pueden realizarse sobre una lista.
- Consiste en ir recorriendo la lista elemento por elemento hasta encontrar el valor buscado o hasta llegar al final, lo que significa que el valor no se encontraba presente.

EJEMPLO Nº 2

Objetivo:

Cargar una lista con números al azar entre 1 y 100, donde la cantidad de elementos será ingresada por el usuario.

Luego se solicita ingresar un valor y buscarlo en la lista, informando su ubicación o -1 si no se lo encuentra

```
import random

def cargarlista(cantidad):
    lista = []
    for i in range(cantidad):
        lista.append(random.randint(1,100))
    return lista

def imprimirlista(lista):
    for i in range(len(lista)):
        print(lista[i],end=" ")
    print()

def busquedasecuencial(lista, dato):
    i = 0
    while i < len(lista) and lista[i] != dato:
        i = i + 1
    if i < len(lista):
        return i
    else:
        return -1

# Programa principal
cant = int(input("¿Cuántos elementos desea cargar? "))
milista = cargarlista(cant)
imprimirlista(milista)
n = int(input("Ingrese el número a buscar: "))
pos = busquedasecuencial(milista,n)
if pos >= 0:
    print("El elemento",n,"se encontró en la posición", pos)
else:
    print("El valor",n,"no se encontró en la lista")
```

ORDENAMIENTO DE VECTORES

- Los elementos que se desea ordenar deben estar almacenados en una estructura de datos, y por eso se suelen usar arreglos (listas o vectores).

- Existen muchos métodos de ordenamiento. En este curso veremos tres: Selección, Intercambio e Inserción.

MÉTODO DE SELECCIÓN

- Consiste en buscar el menor elemento de todo el arreglo e intercambiarlo con el de la primera posición.

- Luego se busca el segundo menor elemento y se lo intercambia con el de la segunda posición, y así sucesivamente

MÉTODO DE SELECCIÓN

```
def metododeseleccion(v):
    largo = len(v)
    for i in range(largo - 1):
        for j in range(i+1, largo):
            if v[i] > v[j]:
                aux = v[i]
                v[i] = v[j]
                v[j] = aux
```

BÚSQUEDA BINARIA

Procedimiento:

- Se verifica si en la mitad de la lista se encuentra el elemento buscado.
- Si no está, resulta fácil deducir para qué lado podría llegar a encontrarse debido al ordenamiento.
- Se descarta una mitad y se repite el proceso sobre la otra.

BÚSQUEDA BINARIA

```
def busquedabinaria(v, dato):
    izq = 0
    der = len(v) - 1
    pos = -1
    while izq <= der and pos == -1:
        centro = (izq + der) // 2
        if v[centro] == dato:
            pos = centro
        elif v[centro] < dato:
            izq = centro + 1
        else:
            der = centro - 1
    return pos
```

CLASE 8

INGENIERA SILVIA PATRICIA BARDELLI

MÉTODO DE INTERCAMBIO O BURBUJEO

- Se basa en comparar cada elemento con el que tiene a su derecha.
- Si es necesario, se los intercambia.
- Luego se avanza a la siguiente pareja y se repite el proceso.

MÉTODO DE INTERCAMBIO O BURBUJEO

```
def metododeintercambio(v):
    desordenado = True
    while desordenado:
        desordenado = False
        for i in range(len(v)-1):
            if v[i]>v[i+1]:
                aux = v[i]
                v[i] = v[i+1]
                v[i+1] = aux
                desordenado = True
```

MÉTODO DE INSERCIÓN

- Comienza a ordenar a partir del segundo elemento de la lista.

- Consiste en mover cada elemento del arreglo hacia la izquierda, haciéndolo retroceder hasta encontrar su ubicación definitiva.

MÉTODO DE INSERCIÓN

```
def metododeinsercion(v):
    for i in range(1, len(v)):  # empieza del 2º elemento
        aux = v[i]
        j = i
        while j>0 and v[j-1]>aux:
            v[j] = v[j-1]
            j = j-1
        v[j] = aux
```

EJEMPLO

Una empresa cuenta con 50 vendedores, numerados del 1 al 50. Por cada venta realizada se ingresa el número de vendedor y el importe de la misma, donde el número de vendedor -1 indica el final de los datos.

Estos datos no están ordenados.
Realizar un programa para imprimir el total de ventas por vendedor.

EJEMPLO – PROGRAMA COMPLETO

```
def leervendedor(maximo):
    n = int(input("Número de vendedor (-1 para terminar) "))
    while (n!=-1 and (n<1 or n>maximo)):
        print("*** Vendedor inválido ***")
        n = int(input("Número de vendedor (-1 para terminar) "))
    return n

VENDEDORES = 50
# Creamos el vector y lo inicializamos con 0
ventas = []
for i in range(VENDEDORES+1):
    ventas.append(0)
# Comenzamos la lectura y acumulación de datos
vendedor = leervendedor(VENDEDORES)
while vendedor!=-1:
    importe = int(input("Importe de la venta? "))
    ventas[vendedor] = ventas[vendedor]+importe
    vendedor = leervendedor(VENDEDORES)
# Imprimir informe final
for i in range(1,VENDEDORES+1):
    print("El vendedor",i,"vendió $ ",ventas[i])
```

Fin del programa de la materia

PROFESORA ING. SILVIA PATRICIA BARDELLI

FUNDAMENTOS DE INFORMÁTICA

PROFESORA: ING. SILVIA PATRICIA BARDELLI

INGENIERA SILVIA PATRICIA BARDELLI

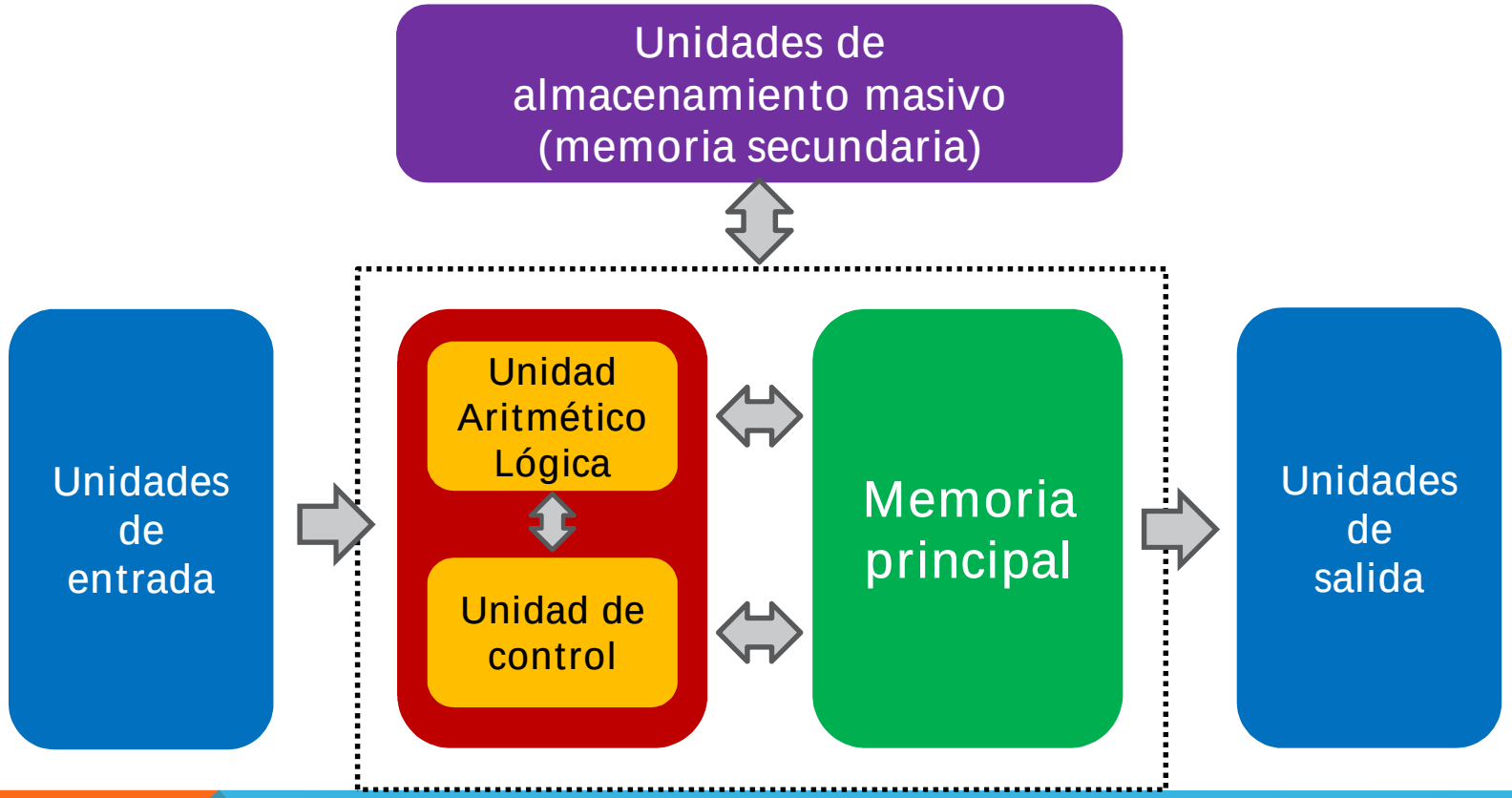
CLASE 1

INGENIERA SILVIA PATRICIA BARDELLI

CLASE 1

Temas:

- Repaso general


Unidades de almacenamiento masivo (memoria secundaria)
Unidades de entrada
Unidad Aritmético Lógica
Unidad de control
Memoria principal
Unidades de salida

CONCEPTOS FUNDAMENTALES

Algoritmo:

Es una secuencia finita y repetible de pasos que describe el proceso a seguir para solucionar un problema dado.

CONCEPTOS FUNDAMENTALES

- Secuencia: Significa que los pasos están ordenados.
- Finita: Que tiene un final.
- Repetible: Partiendo de las mismas condiciones iniciales, el resultado debe ser siempre el mismo.

CONCEPTOS FUNDAMENTALES

Programa:

Es la implementación de un algoritmo en algún lenguaje de programación.

CONCEPTOS FUNDAMENTALES

Lenguaje de Programación:

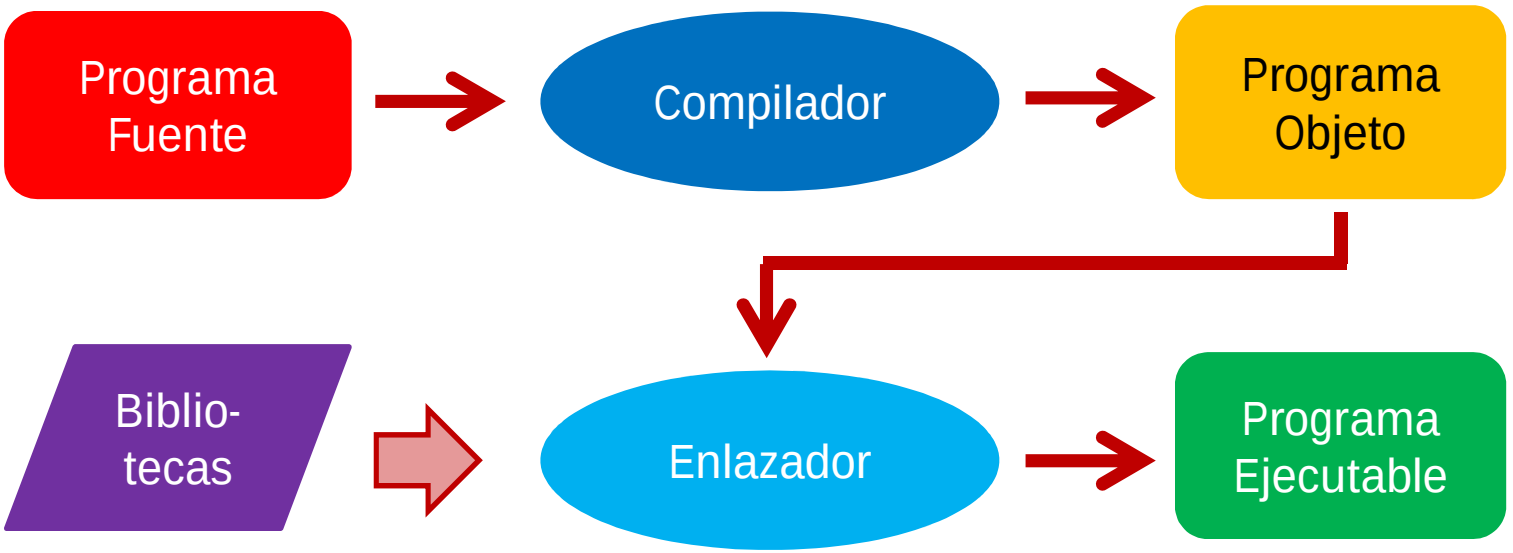
Es un lenguaje formal compuesto por un conjunto de símbolos y reglas sintácticas que se utiliza para darle instrucciones a una computadora.

TRADUCTORES DE LENGUAJES

Compilador:

- Convierte el ***programa fuente***, escrito en un lenguaje de alto nivel, en un ***programa objeto***.
- Este programa objeto debe ser ***enlazado*** o ***vinculado*** con bibliotecas proporcionadas por el fabricante, creando así el ***programa ejecutable***.

PROCESO DE COMPILACIÓN

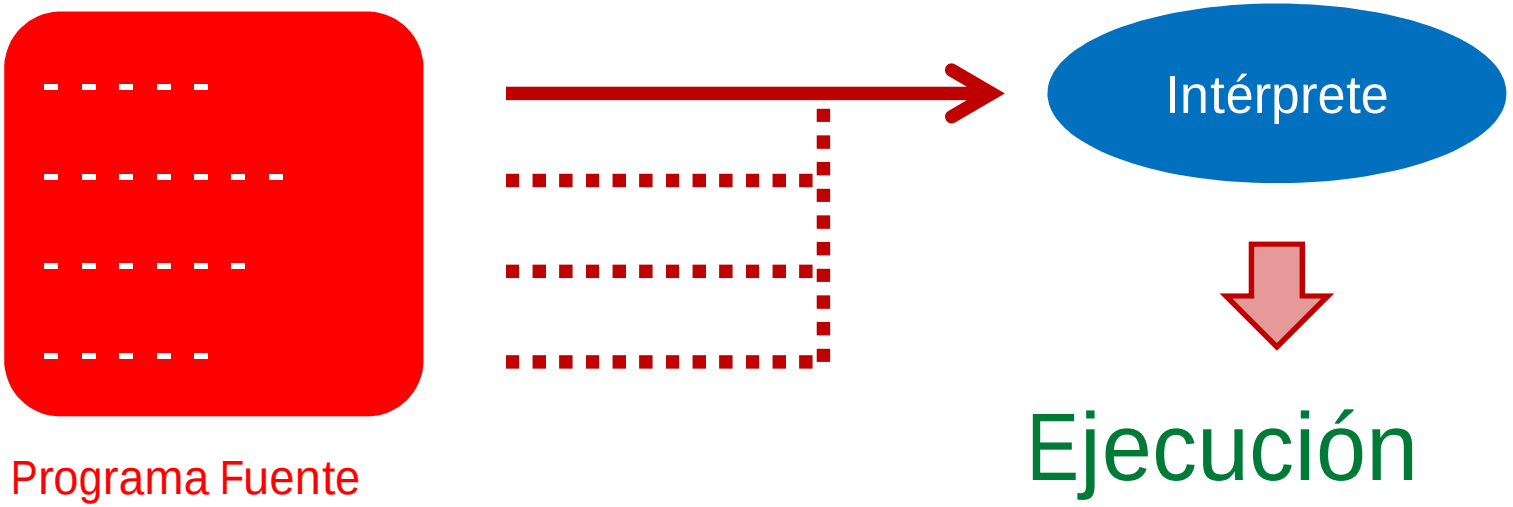


TRADUCTORES DE LENGUAJES

Intérprete:

- Traduce una por una las líneas del programa fuente, ejecutándolas inmediatamente.

PROCESO DE INTERPRETACIÓN



COMPILADOR O INTÉRPRETE?

Compilador:

- Realiza la traducción en tiempo de desarrollo, es decir cuando el programa aún no se está ejecutando.
- Se compila UNA vez.
- Se ejecuta MUCHAS veces.

COMPILADOR O INTÉRPRETE?

Intérprete:

Realiza la traducción a medida que el programa se está ejecutando.

Si una instrucción se repite 1000 veces, será traducida 1000 veces.

COCINAR UNA HAMBURGUESA

- Encender el fuego de la hornalla.
- Colocar la plancha sobre él.
- Colocar la hamburguesa sobre la plancha o parrilla.

1. Esperar un momento

- Determinar si está cocida del lado inferior.
- Sí, continuar
- No, ir a 1
- Dar vuelta la hamburguesa

2. Esperar un momento

- Determinar si está cocida del lado inferior
- Sí, fin
- No, ir a 2

CLASE 2

INGENIERA SILVIA PATRICIA BARDELLI

Variables

Reglas para crear nombres de variables:

- §Sólo letras, números y el guión bajo.
- §No pueden comenzar con un número.
- §No pueden coincidir con las palabras reservadas del lenguaje.

Variables

```
a = 3                    # Asignación de constante
b = a                    # Asignación de variable
c = a + b + 1            # Asignación de expresión
pi = 3.1416
print("La variable a contiene", a, "b contiene", b, "y c
contiene", c)
print("La variable pi contiene", pi)
```

Variables

+ Suma

* Multiplicación

// División entera

** Potenciación

- Resta

/ División real

% Módulo o resto

Ingreso de datos por teclado

```
n = input("Ingrese un número entero: ")
n = int(n)
```

-- - o también - - -

```
n = int(input("Ingrese un numero entero: "))
```

Además existe la función float( ) para poder ingresar números reales.

Ejemplo

```
# Leer dos números enteros y guardarlos en dos variables.
# Luego intercambiar sus valores e imprimir su contenido.

a = int(input("Ingrese un número entero: "))
b = int(input("Ingrese otro número entero: "))
print("A contiene", a, "y B contiene", b)
c = a
a = b
b = c
print("Ahora A contiene", a, "y B contiene", b)
```

CLASE 3

INGENIERA SILVIA PATRICIA BARDELLI

Estructura Secuencial

- Trabajando de esta manera las posibilidades de resolución de problemas son limitadas. Sólo cálculos e impresiones.

- Para que un programa sea ***realmente útil*** es necesario que sea capaz de ***tomar decisiones*** y actuar en consecuencia.

Estructura Alternativa o Condicional

- Por eso, además de la estructura secuencial, existen dos estructuras más en el mundo de la Programación Estructurada.

- La segunda que veremos se denomina ***Estructura Alternativa o Condicional***.

Instrucción if

Formato 1

if ***<condición>***:

. . . . . .

. . . . . .

. . . . . .

Ejemplo Nro 1

```
# Leer un número entero e imprimir un
    # mensaje indicando si es mayor que 5.
    n = int(input("Ingrese un número: "))
    if n > 5:
        print("El número es mayor que 5")
    # Fin del programa
```

Instrucción if

Formato 2

```
if <condición>:
        . . . . . .
        . . . . . .
else:
        . . . . . .
        . . . . . .
```

Ejemplo Nro 2

```
# Leer la calificación que obtuvo un alumno en un
# examen final e imprimir un mensaje indicando si
# aprobó o no la materia. Se aprueba con 4.
nota = int(input("Ingrese la calificación: "))
if nota >= 4:
    print("El alumno aprobó la materia")
else:
    print("El alumno no aprobó la materia")
```

Instrucción if

Formato 3

```
if <condición>:
        . . . . . .
elif <condición>:
        . . . . . .
else:
        . . . . . .
```

Ejemplo Nro 3

```
# Leer un número e informar si es positivo, negativo o cero.
n = int(input("Ingrese un número entero: "))
if n > 0:
    print("El número es positivo")
elif n < 0:
    print("El número es negativo")
else:
    print("El número es cero")
```

Operadores Lógicos

Operador ***and (Y)***:

Cond. 1	Cond. 2	Cond. 1 and Cond. 2
V	V	V
V	F	F
F	V	F
F	F	F

Operadores Lógicos

Operador ***or (O)***:

Cond. 1	Cond. 2	Cond. 1 or Cond. 2
V	V	V
V	F	V
F	V	V
F	F	F

Operadores Lógicos

Operador *not (NO)*:

Condición	not Condición
V	F
F	V

Ejemplo Nro 4

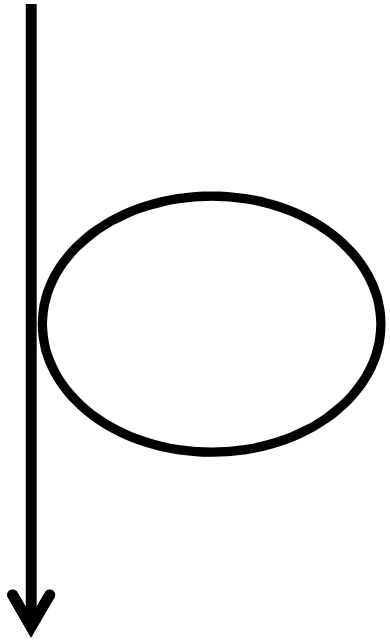
```
# Leer un número entero e imprimir un mensaje indicando
# si corresponde a un número válido de mes.
mes = int(input("Ingrese un número de mes: "))
if mes >= 1 and mes <= 12:
    print("El mes es válido")
else:
    print("El mes es inválido")
```

CLASE 4

INGENIERA SILVIA PATRICIA BARDELLI

ESTRUCTURA ITERATIVA, CICLO O BUCLE

En esta estructura el programa repetirá una porción de su código una cierta cantidad de veces, y luego seguirá adelante.



INSTRUCCIÓN WHILE

while ***<condición>***:

. . . . . .

. . . . . .

. . . . . .

EJEMPLO

```
# Imprimir los números enteros entre 1 y 100

a = 1
while a <= 100:
    print(a)
    a = a + 1
# Fin del programa
```

DEFINICION CONTADOR

Cuando una variable es modificada en una cantidad fija respecto de su valor anterior, se la denomina ***contador***.

ATENCIÓN

¿Qué ocurre si olvidamos la línea que incrementa el contador?

```
a = 1
while a <= 100:
    print(a)
    a = a + 1
# Fin del programa
```

EJEMPLO

```
suma = 0
cant = 0
n = int(input("Ingrese un número o -1 para terminar: "))
while n != -1:
    suma = suma + n
    cant = cant + 1
    n = int(input("Ingrese un número o -1 para terminar: "))
if cant != 0:
    prom = suma/cant
    print("El promedio es", prom)
else:
    print("No se ingresaron valores")
```

DEFINICIÓN ACUMULADOR

Cuando una variable es modificada en una cantidad cambiante respecto de su valor anterior, se la denomina ***acumulador***.

EJEMPLO

Objetivo:

Leer un conjunto de números enteros e imprimir el mayor. El fin de los datos se indica con -1.

¿Cómo podemos proceder para hallar el máximo?

EJEMPLO

```
n = int(input("Ingrese un número o -1 para terminar: "))
mayor = n
while n != -1:
    if n > mayor:
        mayor = n
    n = int(input("Ingrese un número o -1 para terminar: "))
print("El mayor es", mayor)
```

CLASE 5

INGENIERA SILVIA PATRICIA BARDELLI

FUNCIONES

Una función es un módulo independiente de programa que realiza una tarea específica.

PROFESORA ING. SILVIA PATRICIA BARDELLI

EJEMPLO 1: CÁLCULO DE UN PROMEDIO

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total

# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado)
```

VENTAJAS DE TRABAJAR CON FUNCIONES

1. Es posible dividir el programa en módulos más pequeños, para que cada función realice una tarea concreta. Esto facilita el desarrollo y la comprensión del programa.

VENTAJAS DE TRABAJAR CON FUNCIONES

2. Las funciones pueden invocarse muchas veces dentro del mismo programa, evitando la reiteración innecesaria de código.

VENTAJAS DE TRABAJAR CON FUNCIONES

3. Una misma función puede ser utilizada en otros programas, evitando tener que volver a escribir código ya escrito.

TIPOS DE PARÁMETROS

- Existen parámetros ***formales*** y parámetros ***reales***.
- Los formales son los que se escriben en el encabezado de la función.

PARÁMETROS

- Los parámetros reales son los que se escriben en la llamada o invocación.
- Pueden llevar el mismo nombre o nombres diferentes.

TIPOS DE PARÁMETROS

- Existen parámetros ***formales*** y parámetros ***reales***.
- Los formales son los que se escriben en el encabezado de la función.

PARÁMETROS

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total
```

Parámetros formales a y b

Parámetros reales X e y

```
# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado)
```

ÁMBITO DE LAS VARIABLES

total sólo puede usarse dentro de la función

x e y sólo pueden usarse en el programa principal

```
# Función
def calcularpromedio(a, b):
    total = (a + b) / 2
    return total

# Programa principal
x = int(input("Ingrese un numero entero: "))
y = int(input("Ingrese otro numero entero: "))
resultado = calcularpromedio(x, y)
print("El promedio de", x, "y", y, "es", resultado
```

IMPORTANTE

Las funciones deben escribirse antes que el programa principal.

Ni durante, ni después.

El programa principal debe ubicarse siempre al final del código.

EJEMPLO 3

Objetivo:

Escribir una función que reciba como parámetros la fecha de nacimiento de una persona y la fecha actual, y devuelva la edad de la persona.

PROGRAMA COMPLETO

```
def leerentero(min, max):
    print("Ingrese un número entre entre", min, "y", max, end="")
    a = int(input())   # ¿Por qué el mensaje no se muestra aquí?
    while a<min or a>max:
        print("Valor incorrecto. Debe estar entre", min, "y", max)
        a=int(input("Ingrese un numero entero: "))
    return a

def calcularedad(dnac, mnac, anac, dact, mact, aact):
    edad = aact - anac
    if mact < mnac:
        edad = edad - 1
    elif mact == mnac and dact < dnac:
        edad = edad - 1
    return edad
```

```
def leerfecha( ):
    print("Dia? ")
    dia = leerentero(1, 31)
    print("Mes? ")
    mes = leerentero(1, 12)
    print("Año? ")
    año = leerentero(1583,2100)
    return dia, mes, año

# Programa principal
print("Ingrese su fecha de nacimiento:")
dnac,mnac,anac = leerfecha()
print("Ingrese la fecha de hoy:")
dhoy,mhoy,ahoy = leerfecha()
edad=calcularedad(dnac,mnac,anac,dhoy,mhoy,ahoy)
print("Ud. tiene",edad,"años")
```

IMPORTANTE

```
def esprimo(n):
    divisor = 2
    while divisor < n:
        if n % divisor == 0:
            return False
        divisor = divisor + 1
    return True
# Programa principal
num = int(input("Ingrese un número
entero: "))
if esprimo(num):
    print(num, "es primo")
else:
    print(num, "no es primo")
```

Se considera una ***mala práctica de programación*** colocar un return dentro de un ciclo.

BIEN!!

```
def esprimo(n):
    primo = True
    divisor = 2
    while divisor < n:
        if n%divisor==0:
            primo = False
        divisor = divisor + 1
    return primo

# Programa principal
num = int(input("Ingrese un número entero: "))
if esprimo(num):
    print(num, "es primo")
else:
    print(num, "no es primo")
```

CLASE 6

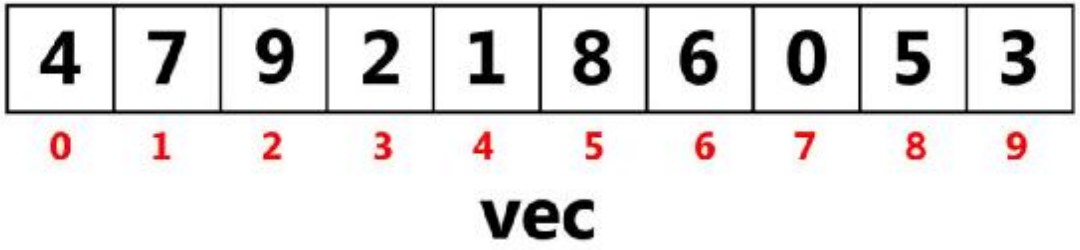
INGENIERA SILVIA PATRICIA BARDELLI

INTRODUCCIÓN

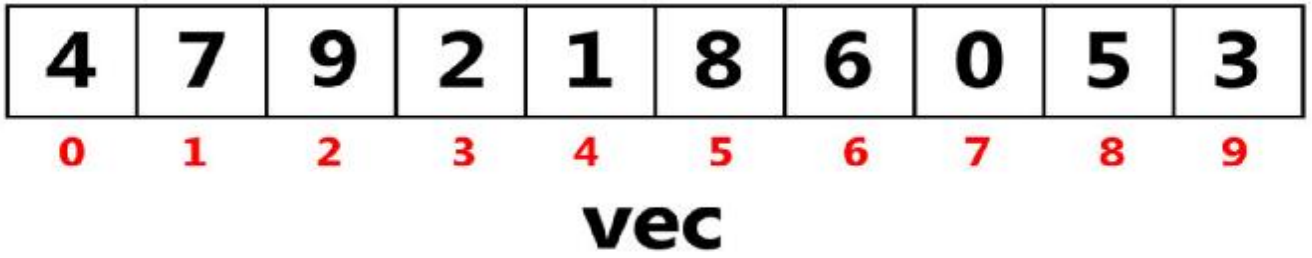
```
a = int(input("Ingrese un número: "))
b = int(input("Ingrese otro número: "))
c = int(input("Y otro más: "))
print(c, b, a)
```

ARREGLOS

Para eso se utiliza un ***subíndice***, que identifica la posición de cada variable. Equivale al ***número de orden*** de la variable dentro del arreglo.



SUBÍNDICES




```
print(vec[2])       # Constante
a = 4
print(vec[a])       # Variable
print(vec[a+3])     # Expresión
```

EJEMPLO

```
# Leer 100 números e imprimirlos en orden inverso
vec = [ ]
i = 0
while i < 100:
   n = int(input("Ingrese un número: "))
   vec.append(n)
   i = i + 1
# Impresión
i = 99
while i >= 0:
   print(vec[i])
   i = i - 1
```

RESOLUCIÓN DEL EJEMPLO NRO 2

```
# Imprimir una lista por pantalla
lista = [4, 7, 2, 9, 5]
x = 0
while x < 5:
    print(lista[x], end=" ")
    x = x + 1
```

EJEMPLO NRO 4

Objetivo:

Leer un conjunto de números y guardarlos en una lista, finalizando la carga con -1. Luego buscar el mayor elemento leído, mostrarlo y eliminarlo del arreglo.

Imprimir por pantalla la lista antes y después del borrado.

```
def imprimirlista(vec):
    largo = len(vec)
    for i in range(largo):
       print(vec[i],end=" ")
    print()


# Programa principal
v = [ ]
n = int(input("Ingrese un numero o -1 para terminar: "))
while n != -1:
    v.append(n)
    n = int(input("Ingrese un numero o -1 para terminar: "))
largo = len(v)
if largo == 0:
    print("No se ingresaron valores")
else:
    mayor = v[0]
    pos = 0
    for i in range(largo):
        if v[i] > mayor:
            mayor = v[i]
            pos = i

  imprimirlista(v)
    print("El máximo es",mayor,"y se encontró en la posición",pos)
    print("Borrando el",mayor)
    del v[pos]
    imprimirlista(v)
```

NOVEDADES DEL EJEMPLO

Instrucción del:

Permite borrar elementos de una lista.
También sirve para eliminar variables o listas completas.

```
del x      # Borra la variable x
del lista[4]      # Borra el elemento de la posición 4
del lista         # Borra la lista completa
```

EJEMPLO:

Uso de for y range( )

```
# Imprimir los números del 1 al 100
for numero in range(1, 101):
    print(numero, end=" ")
```

Cuando ***range( )*** se usa con dos parámetros el primero indica el inicio de la secuencia, mientras que el segundo señala el final de la misma. El valor final no está incluido.

EJEMPLO

range( ) con incremento:

```
# Imprimir los números impares del 1 al 100
for impar in range(1, 101, 2):
    print(impar, end=" ")
```

Cuando ***range( )*** se usa con tres parámetros, el tercero actúa como ***incremento***. Con sólo dos parámetros el incremento es 1.

EJEMPLO:

```
# Imprimir los números del 100 al 1
for i in range(100, 0, -1):
    print(i, end=" ")
```

Cuando el incremento es negativo el valor inicial debe ser mayor que el valor final.

CLASE 7

INGENIERA SILVIA PATRICIA BARDELLI

EJEMPLO

```
import random

def lanzardado( ):
    return random.randint(1, 6)

# Programa principal
dado = lanzardado( )
print(dado)
```

NOVEDADES

Importación de módulos

Todo módulo que desee utilizarse debe ser ***importado*** (incluido) al comienzo del programa.

Función randint(mínimo, máximo)

Genera un número entero al azar entre los límites suministrados, ambos incluidos. El nombre de la función debe ir precedido por el del módulo, separados por un punto.

BÚSQUEDA SECUENCIAL

- La búsqueda secuencial es la más sencilla de las búsquedas que pueden realizarse sobre una lista.
- Consiste en ir recorriendo la lista elemento por elemento hasta encontrar el valor buscado o hasta llegar al final, lo que significa que el valor no se encontraba presente.

EJEMPLO Nº 2

Objetivo:

Cargar una lista con números al azar entre 1 y 100, donde la cantidad de elementos será ingresada por el usuario.

Luego se solicita ingresar un valor y buscarlo en la lista, informando su ubicación o -1 si no se lo encuentra

```
import random

def cargarlista(cantidad):
    lista = []
    for i in range(cantidad):
        lista.append(random.randint(1,100))
    return lista

def imprimirlista(lista):
    for i in range(len(lista)):
        print(lista[i],end=" ")
    print()

def busquedasecuencial(lista, dato):
    i = 0
    while i < len(lista) and lista[i] != dato:
        i = i + 1
    if i < len(lista):
        return i
    else:
        return -1

# Programa principal
cant = int(input("¿Cuántos elementos desea cargar? "))
milista = cargarlista(cant)
imprimirlista(milista)
n = int(input("Ingrese el número a buscar: "))
pos = busquedasecuencial(milista,n)
if pos >= 0:
    print("El elemento",n,"se encontró en la posición", pos)
else:
    print("El valor",n,"no se encontró en la lista")
```

ORDENAMIENTO DE VECTORES

- Los elementos que se desea ordenar deben estar almacenados en una estructura de datos, y por eso se suelen usar arreglos (listas o vectores).

- Existen muchos métodos de ordenamiento. En este curso veremos tres: Selección, Intercambio e Inserción.

MÉTODO DE SELECCIÓN

- Consiste en buscar el menor elemento de todo el arreglo e intercambiarlo con el de la primera posición.

- Luego se busca el segundo menor elemento y se lo intercambia con el de la segunda posición, y así sucesivamente

MÉTODO DE SELECCIÓN

```
def metododeseleccion(v):
    largo = len(v)
    for i in range(largo - 1):
        for j in range(i+1, largo):
            if v[i] > v[j]:
                aux = v[i]
                v[i] = v[j]
                v[j] = aux
```

BÚSQUEDA BINARIA

Procedimiento:

- Se verifica si en la mitad de la lista se encuentra el elemento buscado.
- Si no está, resulta fácil deducir para qué lado podría llegar a encontrarse debido al ordenamiento.
- Se descarta una mitad y se repite el proceso sobre la otra.

BÚSQUEDA BINARIA

```
def busquedabinaria(v, dato):
    izq = 0
    der = len(v) - 1
    pos = -1
    while izq <= der and pos == -1:
        centro = (izq + der) // 2
        if v[centro] == dato:
            pos = centro
        elif v[centro] < dato:
            izq = centro + 1
        else:
            der = centro - 1
    return pos
```

CLASE 8

INGENIERA SILVIA PATRICIA BARDELLI

MÉTODO DE INTERCAMBIO O BURBUJEO

- Se basa en comparar cada elemento con el que tiene a su derecha.
- Si es necesario, se los intercambia.
- Luego se avanza a la siguiente pareja y se repite el proceso.

MÉTODO DE INTERCAMBIO O BURBUJEO

```
def metododeintercambio(v):
    desordenado = True
    while desordenado:
        desordenado = False
        for i in range(len(v)-1):
            if v[i]>v[i+1]:
                aux = v[i]
                v[i] = v[i+1]
                v[i+1] = aux
                desordenado = True
```

MÉTODO DE INSERCIÓN

- Comienza a ordenar a partir del segundo elemento de la lista.

- Consiste en mover cada elemento del arreglo hacia la izquierda, haciéndolo retroceder hasta encontrar su ubicación definitiva.

MÉTODO DE INSERCIÓN

```
def metododeinsercion(v):
    for i in range(1, len(v)):  # empieza del 2º elemento
        aux = v[i]
        j = i
        while j>0 and v[j-1]>aux:
            v[j] = v[j-1]
            j = j-1
        v[j] = aux
```

EJEMPLO

Una empresa cuenta con 50 vendedores, numerados del 1 al 50. Por cada venta realizada se ingresa el número de vendedor y el importe de la misma, donde el número de vendedor -1 indica el final de los datos.

Estos datos no están ordenados.
Realizar un programa para imprimir el total de ventas por vendedor.

EJEMPLO – PROGRAMA COMPLETO

```
def leervendedor(maximo):
    n = int(input("Número de vendedor (-1 para terminar) "))
    while (n!=-1 and (n<1 or n>maximo)):
        print("*** Vendedor inválido ***")
        n = int(input("Número de vendedor (-1 para terminar) "))
    return n

VENDEDORES = 50
# Creamos el vector y lo inicializamos con 0
ventas = []
for i in range(VENDEDORES+1):
    ventas.append(0)
# Comenzamos la lectura y acumulación de datos
vendedor = leervendedor(VENDEDORES)
while vendedor!=-1:
    importe = int(input("Importe de la venta? "))
    ventas[vendedor] = ventas[vendedor]+importe
    vendedor = leervendedor(VENDEDORES)
# Imprimir informe final
for i in range(1,VENDEDORES+1):
    print("El vendedor",i,"vendió $ ",ventas[i])
```

Fin del programa de la materia

PROFESORA ING. SILVIA PATRICIA BARDELLI