



**CICLO 1**

[FORMACIÓN POR CICLOS]

# Fundamentos de **PROGRAMACIÓN**



Ingeni@  
Soluciones TIC



UNIVERSIDAD  
DE ANTIOQUIA

Facultad de Ingeniería



Lectura

---

# IDENTIFICAR

*el uso de vector*



En la práctica existen situaciones en las cuales se necesita manejar un gran volumen de datos del mismo tipo y que implican una gran cantidad de variables, lo cual hace que la solución algorítmica sea muy rígida e impráctica. En este módulo presentamos una de estas situaciones y su solución, utilizando arreglos de una dimensión denominados vectores.

## Descripción del problema.

Consideremos la situación de que se hizo un **censo** en la ciudad de Medellín y que se grabó un archivo en disco, llamado censo, que contiene la siguiente información en cada registro: municipio, dirección y número de personas. Cada registro contiene los datos correspondientes a cada vivienda visitada.

Interesa procesar el archivo y calcular el total de personas que viven en Medellín.

Basta con definir un acumulador para ir sumando las personas que viven en cada vivienda y al final mostrar este resultado. Un programa para efectuar esta tarea es:

```
acuMed = 0
np = int(input("Entre número de personas en la
vivienda "))
while np >= 0:
    acuMed = acuMed + np
    np = int(input("Entre número de personas en
la vivienda "))
print("Total personas en Medellin", acuMed)
```

Si el censo se hace en dos municipios, Medellín y Bello, y los códigos asignados a los municipios son:

mpio = {  
1. Medellín: acuMed  
2. bello: acuBello}

un algoritmo para procesar el archivo censo y calcular e imprimir el total de habitantes de cada municipio es:

```
acuMed = 0
acuBello = 0
mpio = int(input("Entre código de municipio "))
while mpio != 0:
    np = int(input("Entre número de personas en la
                    vivienda "))
    if mpio == 1:
        acuMed = acuMed + np
    else:
        acuBello = acuBello + np
    mpio = int(input("Entre código de municipio "))
print("Total personas en Medellín", acuMed)
print("Total personas en Bello", acuBello)
```

Observe que en este algoritmo ya necesitamos dos acumuladores: uno para el total de habitantes en Medellín (**acuMed**) y otro para el total de habitantes de Bello (**acuBello**). Es importante notar también que en este programa se considera que el usuario no se equivoca tecleando el código del municipio. Si el usuario teclea un número diferente de 0 y 1, el programa asume que tecleó un 2.

Si el censo hubiera sido para Medellín, Bello, Itagüí y Envigado, se le asigna un código a cada municipio:

Municipio =

1. Medellín
2. Bello
3. Itagüí
4. Envigado

Al tener cuatro municipios se necesita un acumulador para cada municipio. Llamémoslos **acuMed**, **acuBello**, **aculta** y **acuEnv**. Nuestro programa será:



```

acuMed = 0
acuBello = 0
aculta = 0
acuEnv = 0
mpio = int(input("Entre código de municipio "))
while mpio != 0:
    np = int(input("Entre número de personas "))
    if mpio == 1:
        acuMed = acuMed + np
    elif mpio == 2:
        acuBello = acuBello + np
    elif mpio == 3:
        aculta = aculta + np
    elif mpio == 4:
        acuEnv = acuEnv + np
    else:
        print("Código de municipio inválido ")
    mpio = int(input("Entre código de municipio "))
print("Habitantes Medellín", acuMed, "Habitantes Bello",
acuBello)
print("Habitantes Itagüí", aculta, "Habitantes Envigado",
acuEnv)

```

Observe que en este nuevo algoritmo ya se necesitan cuatro acumuladores, uno por cada municipio que se procese. Fíjese también que hemos controlado que el código del municipio sea inválido.

Si el censo hubiera sido para los 125 municipios del departamento, se requerirían 125 acumuladores, uno para cada municipio. Tendríamos que manejar 125 variables, nuestra instrucción con la estructura **if – elif – else** sería muy extensa y el algoritmo sería completamente impráctico, ya que cada vez que varíe el número de municipios debemos modificar nuestro programa.

## **Solución al problema.**

### **Concepto de vector.**

Presentaremos una solución alterna utilizando una estructura arreglo de una dimensión en la cual definimos un área de memoria con cierto número de posiciones e identificamos cada posición con un número entero.



Veamos dicha estructura:

acmpio	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	3	1	6	28	9	4	5	7	6	3	2	7	5	8

A nuestra estructura la hemos llamado **acmpio** y tiene una capacidad de 14 elementos, los cuales están identificados o numerados desde el 0 hasta el 13. Para referirnos a algún elemento lo hacemos con el nombre de la estructura y un subíndice que indique la posición de dicho elemento.

Si nuestra estructura tiene los datos que se muestran y queremos imprimir el dato que se halla en la posición 4, escribimos:

```
print(acmpio[4])
```

Y el resultado de ejecutar esta instrucción es que escribe el número 9.

Dicha estructura se conoce como un "arreglo de una dimensión" y en el contexto de algoritmia se denomina **vector**.

En general, para referirnos al contenido de una posición, lo hacemos con el nombre del vector y un entero que se refiere a la posición. Dicho entero lo escribimos entre corchetes y lo seguiremos llamando **subíndice**.

Retomando nuestro ejemplo del censo en el departamento de Antioquia con 125 municipios, ya hemos visto que se requieren 125 acumuladores, uno por cada municipio. Los datos a procesar serán un registro por cada vivienda. Cada registro tiene el código del municipio, la dirección de la vivienda y el número de personas que habitan en ella. Un programa para procesar todos los registros y mostrar al final el número de habitantes de cada municipio es:

```
n = 125
acmpio = [0] * (n + 1)
mpio = int(input("Entre código de municipio "))
while mpio != 0:
    np = int(input("Entre número de personas "))
    acmpio[mpio] = acmpio[mpio] + np
    mpio = int(input("Entre código de municipio "))
for i in range(1, n + 1):
    print("Municipio", i, "Habitantes =", acmpio[i])
```

Al vector **acmpio** lo definimos con la instrucción **acmpio = [0] \* (n + 1)**, lo cual significa que la variable acmpio será un vector de 126 elementos, cuyos subíndices van desde 0 hasta 125. Recuerde que son 125 municipios y que los códigos de los municipios empiezan en 1. Es decir, no estamos utilizando la posición 0 del vector. El hecho de usar corchetes en la definición instruye a la máquina para que sepa que la variable acmpio es un arreglo de una dimensión. El 0 entre los corchetes le indica que todos los elementos del vector se inicializan en 0.

Para ser más exhaustivos acerca de cómo definir vectores, veamos dos ejemplos.

La instrucción **V = [0] \* 10** crea un vector denominado **V**, que tiene la siguiente configuración:

	0	1	2	3	4	5	6	7	8	9
V	0	0	0	0	0	0	0	0	0	0

La instrucción **V = [3] \* 12** crea un vector denominado **V**, que tiene la siguiente configuración:

	0	1	2	3	4	5	6	7	8	9	10	11
V	3	3	3	3	3	3	3	3	3	3	3	3

El valor entre corchetes será el valor inicial de cada celda. El hecho de que este valor esté entre corchetes significa que la variable es un arreglo de una dimensión (vector) y el número a continuación del asterisco indica el número de posiciones que tendrá el arreglo. Dichas posiciones van numeradas a partir de 0.

En general, si la definición es **V = [0] \* n**, el vector tendrá n posiciones numeradas desde 0 hasta **n - 1**.

Esa es la razón por la cual, para manejar 125 municipios, defino el vector de 126 elementos en el programa, puesto que los municipios están codificados desde 1 hasta 125. Si definiera el vector de tamaño 125, solo maneja municipios desde 1 hasta 124.

En general, y para mi gusto, siempre defino los vectores con una posición más de las que necesito. Si trabajo con **n** datos defino el vector con **n + 1** elementos. Trabajo los datos desde la posición 1 hasta la posición **n**, y la posición 0 del vector la utilizo para otros fines, dependiendo de la aplicación que esté desarrollando.

