PROGRAMACIÓN I - UNIDAD IV: ÁMBITO DE VARIABLES, ORDENAMIENTO Y BUSQUEDAS

Autor:Ing. Lucía Amanda Melfi

Control de versiones:

Versión	Descripción cambios	Fecha
1.0	Inicial	14/06/2021
1.1	Se agrega ordenamiento y ejemplo de pseint para búsqueda binaria	22/6/2021
1.2	Se corrigió ejercicio ejemplo búsqueda dicotomica	30/6/2021

Contenido

Variables locales y variables globales

Según el lugar en el cual aparece la definición **de la** variable, una variable se puede definir como:

- **Variable global:** Es aquella que se ubica al exterior de cualquier bloque de definición de funciones o de cualquier otro bloque.
- **Variable local:** Es aquella que se ubica al interior de un bloque.

Ejemplo 1

Realizar un programa que permita almacenar 5 números reales en un arreglo, y que al final se imprima la suma de cada uno de los números ingresados en la misma.

Ejemplo en pseudocódigo con variables locales:

```
Algoritmo Ejemplo

    calcula_suma

FinAlgoritmo

SubProceso calcula_suma()

    Definir nros, suma Como Real // Variables locales
    Definir i Como Entero // Variable local
    Dimension nros[5]

    suma = 0;

    Para i←1 Hasta 5 Con Paso 1 Hacer
        Escribir Sin Saltar "Ingresa el número de la posición ", i
        Leer nros[i]
        suma = suma + nros[i]
    Fin Para

    Escribir "El resultado de la suma de todos los números es: ", suma

FinSubProceso
```

```
PSeInt - Ejecutando proceso EJEMPLO - Goo...
rollapp.com/client?sessionId=86e2e0f7-f484-43cd-97ac...
*** Ejecución Iniciada. ***
Ingresa el número de la posición 1> 3
Ingresa el número de la posición 2> 2
Ingresa el número de la posición 3> 4
Ingresa el número de la posición 4> 5
Ingresa el número de la posición 5> 6
El resultado de la suma de todos los números es: 20
*** Ejecución Finalizada. ***
```

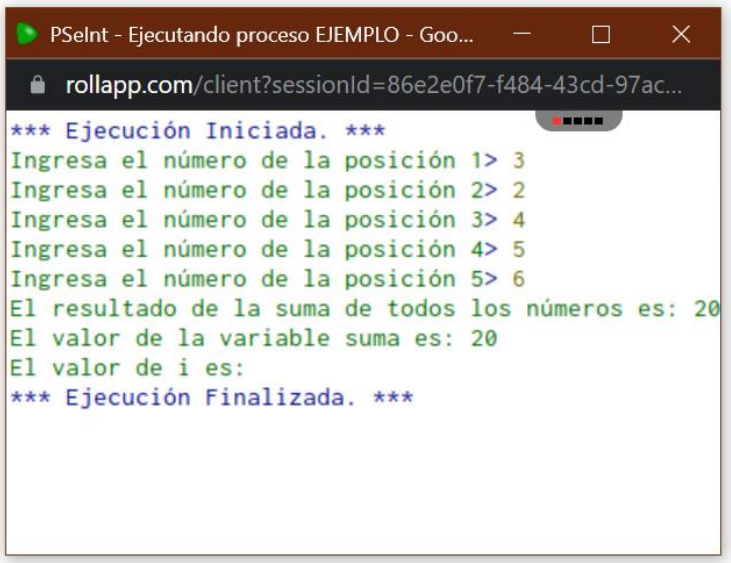
Las variables *nros*, *suma*, *i* son variables locales que tienen vigencia dentro del bloque de repetición. Si intentáramos invocarla por fuera del bloque de repetición, no podríamos acceder a su valor como se muestra a continuación:

```
Algoritmo Ejemplo

    calcula_suma

    Escribir "El valor de la variable suma es: ", suma

FinAlgoritmo

SubProceso calcula_suma()

    Definir nros, suma Como Real // Variables locales
    Definir i Como Entero // Variable local
    Dimension nros[5]

    suma = 0;

    Para i←1 Hasta 5 Con Paso 1 Hacer
        Escribir Sin Saltar "Ingresa el número de la posición ", i
        Leer nros[i]
        suma = suma + nros[i]
    Fin Para

    Escribir "El resultado de la suma de todos los números es: ", suma

FinSubProceso
```

```
PSeInt - Ejecutando proceso EJEMPLO - Goo...
rollapp.com/client?sessionId=86e2e0f7-f484-43cd-97ac...
*** Ejecución Iniciada. ***
Ingresa el número de la posición 1> 3
Ingresa el número de la posición 2> 2
Ingresa el número de la posición 3> 4
Ingresa el número de la posición 4> 5
Ingresa el número de la posición 5> 6
El resultado de la suma de todos los números es: 20
El valor de la variable suma es:
*** Ejecución Finalizada. ***
```

Ejemplo en pseudocódigo con variables locales y globales:

```
1  Algoritmo Ejemplo
2
3      Definir nros, suma Como Real // Variables globales
4
5      Dimension nros[5]
6
7      suma = 0;
8
9      calcula_suma(nros, suma)
10
11     Escribir "El valor de la variable suma es: ", suma
12
13     Escribir "El valor de i es: ", i
14
15 FinAlgoritmo
16
17 SubProceso calcula_suma(n, sum Por Referencia)
18
19     Definir i Como Entero // Variable local
20
21     Para i←1 Hasta 5 Con Paso 1 Hacer
22         Escribir Sin Saltar "Ingresa el número de la posición ", i
23         Leer n[i]
24         sum = sum + n[i]
25     Fin Para
26
27     Escribir "El resultado de la suma de todos los números es: ", sum
28
29 FinSubProceso
30
```



En esta resolución se observa las variables globales *nros* y *suma* que pueden ser llamadas por fuera y dentro del bloque de código del subproceso. En cambio, la variable local *i*, solamente tiene vigencia dentro del subproceso ya que no es posible obtener su valor en el programa principal.

Importante: en PSeint para que las variables globales tengan visibilidad dentro del subproceso es necesario pasarlas por referencia en los parámetros de entrada.

Ordenamiento

Muchas veces es mucho más eficiente trabajar con datos ordenados por lo que tenemos que recurrir a la utilización de un algoritmo de ordenamiento que sirve para reorganizar el orden de los elementos de una estructura, como un vector o arreglo.

Podemos ordenar según varios criterios, por ejemplo, se pueden ordenar números o una lista de nombres por orden alfabético, y se pueden ordenar en orden ascendente (de menor a mayor) y en orden descendente (de mayor a menor).

Existen varios tipos de algoritmos de ordenamiento, algunos más eficientes que otros, algunos solo se pueden utilizar con números, etc.

En éste curso vamos a explicar el algoritmo de ordenamiento “Por selección” que es uno de los más sencillos y resulta bastante eficiente para un conjunto no muy grande de datos. El concepto de éste algoritmo es el siguiente:

Siendo N el tamaño del arreglo, vamos a recorrer el arreglo seleccionando en cada iteración un pivote.

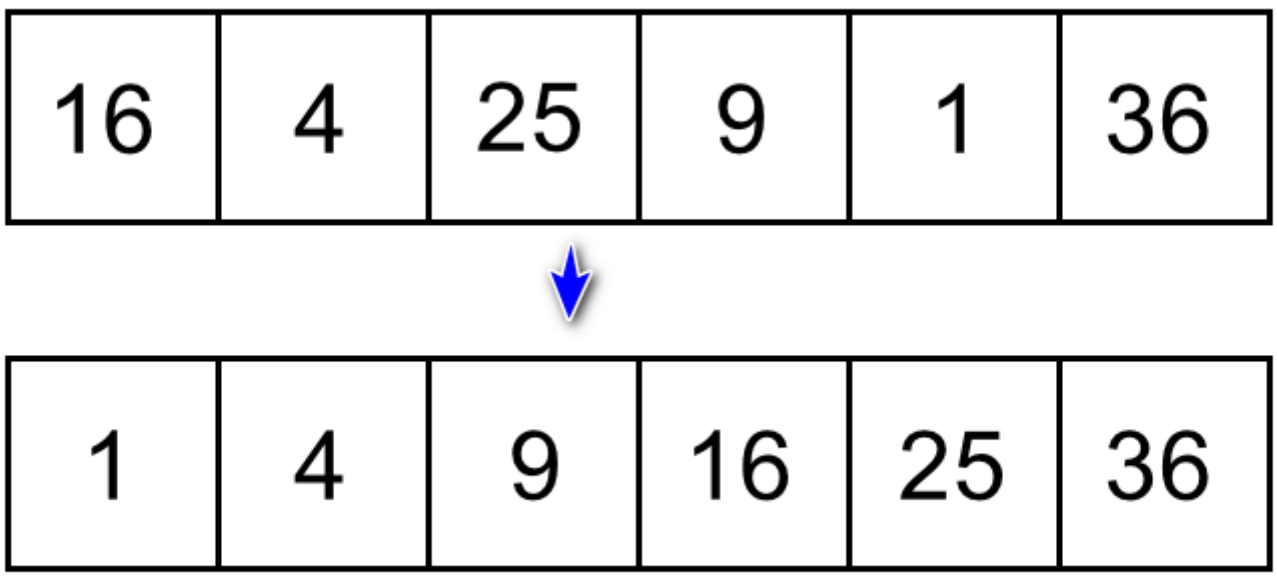
Los pivotes serán:

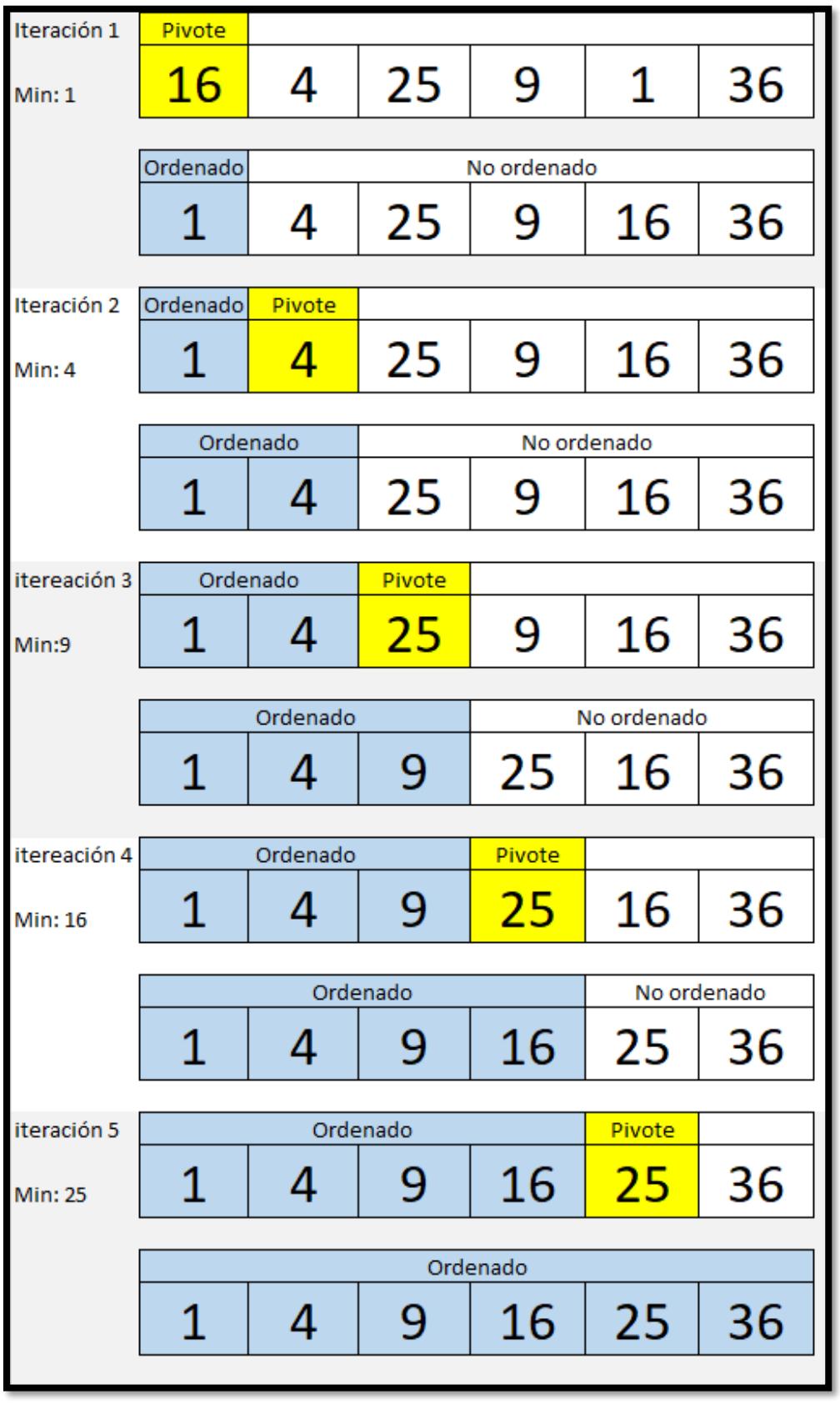
En la primera iteración, el primer elemento del arreglo, en la segunda iteración, el segundo elemento del arreglo.. y así sucesivamente hasta llegar al último pivote que es el penúltimo elemento del arreglo.

Por cada iteración vamos a buscar el mínimo número que se encuentra entre la posición del pivote y el fin del arreglo. Luego intercambiamos las posiciones entre el pivote y el mínimo.

Ejemplo 2:

Supongamos que partimos de un arreglo de 6 elementos y queremos ordenarlos de menor a mayor.





El pseudocódigo para resolver éste algoritmo es:

```
SubProceso OrdenSeleccion(arreglo, dim) //se pasa un arreglo de tamaño dim
    Definir i, j, pos_menor, aux Como Entero
    Para i←0 Hasta dim-2 Hacer
        pos_menor←i
        Para j←i+1 Hasta dim-1 Hacer
            Si arreglo[j]<arreglo[pos_menor] Entonces
                pos_menor←j
            FinSi
            cant_comparaciones=cant_comparaciones+1
        FinPara
        aux←arreglo[i]
        arreglo[i]←arreglo[pos_menor]
        arreglo[pos_menor]←aux
    FinPara
FinSubProceso
```

Nota: Si se quisiera ordenar en orden descendente (de mayor a menor), solo debe modificarse el signo “<” por el “>”

Algoritmos de Búsqueda

Para encontrar un dato dentro de un arreglo existen diversos algoritmos que varían en complejidad, eficiencia, tamaño del dominio de búsqueda.

Los procesos de búsqueda involucran recorrer un arreglo completo con el fin de encontrar algo. Lo más común es buscar el menor o mayor elemento (cuando se puede establecer un orden), o buscar el índice de un elemento determinado. Para buscar el menor o mayor elemento de un arreglo, podemos usar la estrategia, de suponer que el primero o el último es el menor (mayor), para luego ir comparando con cada uno de los elementos, e ir actualizando el menor (mayor).

Los algoritmos de búsqueda que se explicarán a continuación son: Búsqueda Secuencial y Búsqueda Binaria.

Búsqueda Secuencial

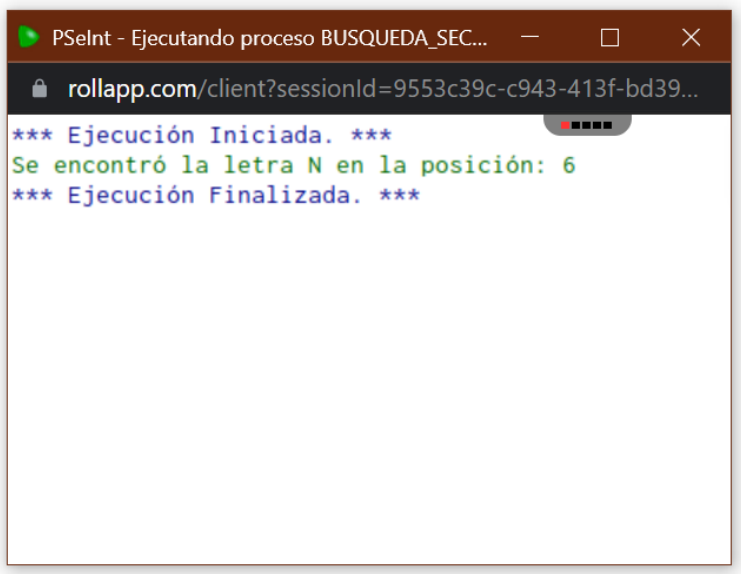
Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta que se encuentra. La velocidad de ejecución depende linealmente del tamaño del arreglo. Considerando la cantidad de comparaciones:

- Mejor Caso: El elemento buscado está en la primera posición. Es decir, se hace una sola comparación
- Peor Caso: El elemento buscado está en la última posición. Necesitando igual cantidad de comparaciones que de elementos el arreglo
- En Promedio: El elemento buscado estará cerca de la mitad. Necesitando en promedio, la mitad de las comparaciones que de elementos.

Ejemplo 3

Buscar el elemento “N” en un arreglo previamente definido y devolver su posición.

```
Algoritmo busqueda_secuencial

    Definir palabra Como Caracter
    Definir i Como Entero

    Dimension palabra[10]
    palabra[1] = "S"
    palabra[1] = "E"
    palabra[3] = "C"
    palabra[4] = "U"
    palabra[5] = "E"
    palabra[6] = "N"
    palabra[7] = "C"
    palabra[8] = "I"
    palabra[9] = "A"
    palabra[10] = "L"

    i = 1

    Mientras i ≤ 10 Hacer
        Si palabra[i] == "N"
            Escribir "Se encontró la letra N en la posición: ", i
            i = 10
        FinSi
        i = i + 1
    FinMientras

FinAlgoritmo
```



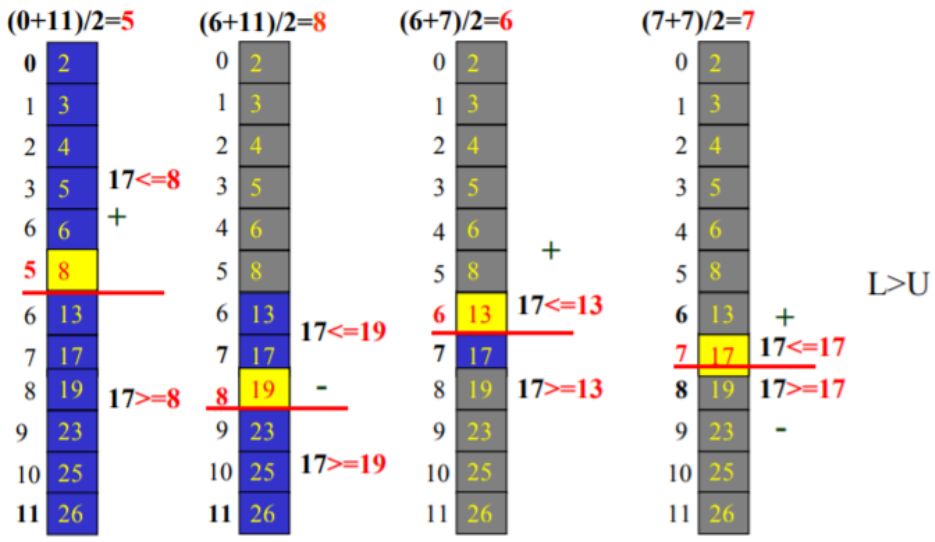
En este ejemplo, se recorre secuencialmente el arreglo hasta encontrar el elemento deseado.

Búsqueda Binaria

Este método de búsqueda de valor funciona con arreglos ordenados. La idea principal es la reducción del espacio en el cual se busca el valor. Al inicio, el espacio es el arreglo completo. Entonces, se toma un índice al interior del intervalo que represente la mitad de la longitud y este elemento (el pivote) se compara con el valor buscado. Si la comparación pone en evidencia la equivalencia, la búsqueda termina con éxito, de lo contrario, si el valor buscado es menor que el elemento pivote, entonces se reduce el intervalo de búsqueda hacia la izquierda. En el caso opuesto, el intervalo se reduce a la derecha.

Ejemplo 4

Encontrar la posición del número 17 en un arreglo numérico realizando una búsqueda binaria.



En este ejemplo gráfico, se observa cada una de las cuatro iteraciones que se necesitaron para encontrar el número deseado.

Ejemplo en PSeInt:

```
//dim es la dimension del arreglo ordenado y buscar es el numero a encontrar
SubProceso BusquedaDicotomica(arreglo, dim, buscar)
    Definir i, centro, inferior, superior como Entero
    Definir encontrado como Logico
    inferior=0
    superior=dim-1
    encontrado=falso
    Repetir
        centro=trunc((inferior+superior)/2)
        si arreglo[centro]=buscar
            Mostrar "El elemento fue encontrado en la posición: ", centro+1
            encontrado=Verdadero
        SiNo
            Si arreglo[centro]<buscar
                inferior=centro+1
            SiNo
                superior=centro-1
            FinSi
        FinSi
        Si inferior>superior
            Mostrar "Número no encontrado"
        FinSi
    Mientras que !encontrado y inferior≤superior
FinSubProceso
```

Bibliografía consultada

Juganaru Mathieu, M. (2015). Introducción a la programación. México D.F, México: Grupo Editorial Patria. Recuperado de https://elibro.net/es/ereader/utnfrro/39449?page=296.