

## GUÍA N°6

### LENGUAJE C: ESTRUCTURAS, MEMORIA DINÁMICA Y RECURSIÓN

#### I. ESTRUCTURAS.

1. Considere las siguientes definiciones y responda: (*Prueba*)

```
struct fecha{
    int Dia;
    int Mes;
    int Anno;
};

typedef struct fecha tFecha;

struct persona{
    char Nombres[100];
    char ApPaterno[100];
    char ApMaterno[100];
    char RUT[9]; // No almacena ni puntos ni guión, pero SI el DV en la última posición.
    tFecha Nacimiento;
};

typedef struct persona tPersona;

struct producto{
    char Numero[4]; //Identificador del Producto Bancario.
    long int Saldo;
    tFecha Apertura;
    int tipo; //1: Cuenta Corriente, 2: Cuenta de Ahorro.
};

typedef struct producto tProducto;

struct cliente{
    tPersona DatosPersonales;
    int NumProductos; //Cantidad de productos bancarios que posee.
    tProducto DatosProductos[10]; //Un cliente puede tener a lo más 10 productos.
    tFecha Afiliacion;
};

typedef struct cliente tCliente;

tCliente BANCO[1000];
```

Indique la **secuencia de instrucciones** que permita:

- Ingresar al cliente número 1 del banco la fecha de su nacimiento que corresponde al 24 de abril de 1990.
- Corregir el dígito verificador del cliente número 40 del BANCO, cambiándolo por una ‘K’.
- Agregar un nuevo producto (ya tiene 3) al 5° cliente del BANCO, que corresponde a:
  - a. Una cuenta de ahorro,
  - b. Con el identificador “BC82”,
  - c. Con un saldo de \$10000,
  - d. Con una fecha de apertura correspondiente a hoy.

Además, debe indicar en la información del BANCO que tal cliente tiene un producto más.

- Corregir el nombre del décimo cliente del BANCO: se llama Juan Martínez Pezoa, pero su nombre aparece como Juana.
- Almacenar la fecha de nacimiento del cliente 52, sólo se sabe que hoy tiene 39 años, 8 meses y 6 días.



2. Considere las siguientes definiciones y responda: (*Prueba*)

```
struct fecha{
    int Dia;
    int Mes;
    int Anno;
};

typedef struct fecha tFecha;

struct persona{
    char Nombres[100];
    char ApPaterno[100];
    char ApMaterno[100];
    char RUT[9]; // No almacena ni puntos ni guión, pero SI el DV en la última posición.
    tFecha Nacimiento;
};

typedef struct persona tPersona;

struct producto{
    char Numero[4]; //Identificador del Producto Bancario.
    long int Saldo;
    tFecha Apertura;
    int tipo; //1: Cuenta Corriente, 2: Cuenta de Ahorro.
};

typedef struct producto tProducto;

struct cliente{
    tPersona DatosPersonales;
    int NumProductos; //Cantidad de productos bancarios que posee.
    tProducto DatosProductos[10]; //Un cliente puede tener a lo más 10 productos.
    tFecha Afiliacion;
};

typedef struct cliente tCliente;

tCliente BANCOS[1000][2];
int cantidadClientesItau;
int cantidadClientesSantander;
```

- Escriba una función en C que permita ingresar la información de un cliente nuevo en cualquiera de los 2 bancos. Asuma que las variables cantidadClientesItau y cantidadClientesSantander mantienen la cantidad actual de clientes de cada banco. Por ejemplo, si cantidadClientesItau = 300, quiere decir que existe el registro de 300 clientes en el banco Itau.
- Escriba una función en C que permita imprimir por pantalla el nombre, apellido paterno, rut, producto y saldo los clientes del banco Santander con productos cuyos saldos sean negativos.

## II. RECURSIÓN.

1. Dado la siguiente función en C:

```
int XXX(int *A, int b, int c, int d)
{
    if (c == b)
        return d;
    else
    {
        if (A[c] < d)
            return XXX(A, b, c+1, A[c]);
        else
            return XXX(A, b, c+1, d);
    }
}
```

Suponga que se llama a esta función desde `main` con los siguientes valores:

<b>A</b>	=	<table><tr><td>20</td><td>32</td><td>5</td><td>1</td><td>8</td></tr></table>	20	32	5	1	8
20	32	5	1	8			
		<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table>	0	1	2	3	4
0	1	2	3	4			

b = 5

c = 1

d = A[0]

Responda:

- ¿Qué retorna la función con estos valores en sus parámetros de entrada?
- Si el arreglo A estuviera ordenado en forma ascendente, cuántas veces se llamaría a la función **XXX**?

2. Dado la siguiente función en C:

```
int YYY(int *A, int b, int c, int d)
{
    int x;

    if (c > b)
        return -1;

    x = (b+c)/2;

    if (A[x] == d)
        return x;
    else
    {
        if (d < A[x])
            return YYY(A, b, x-1, d);
        else
            return YYY(A, x+1, c, d);
    }
}
```

Suponga que se llama a esta función desde `main` con los siguientes valores:

<b>A</b>	=	<table><tr><td>1</td><td>10</td><td>50</td><td>201</td><td>263</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table>	1	10	50	201	263	0	1	2	3	4
1	10	50	201	263								
0	1	2	3	4								

b = 0

c = 4

d = 10

Responda:

- ¿Qué retorna la función con estos valores en sus parámetros de entrada?
- Y ¿Qué retorna la función si  $d = 10$ ?

3. Considere el siguiente programa que posee a la función recursiva “XXXX”, y marque la alternativa correcta (*Prueba*):

```
#include <stdio.h>
#include <stdlib.h>

int XXXX(int *A, int n, int b, int c)
{
    if (b == n)
        return c;
    else
    {
        if (A[b] > c)
            return XXXX(A, n, b+1, A[b]);
        else
            return XXXX(A, n, b+1, c);
    }
}

int main()
{
    int *A, res, n;

    printf("\nIngrese el tamanno del arreglo: ");
    scanf("%d", &n);

    A = (int *)malloc(n * sizeof(int));

    LlenaArreglo(A, n);          //El usuario ingresa los valores al arreglo A.
    res = XXXX(A, n, 1, A[0]);
    printf("\n\nEl resultado es: %d\n", res);

    free(A);
}
```

- ¿Qué hace la función recursiva XXXX?
  - a. Replica el valor que está en la posición 0 del arreglo A, al resto de las posiciones.
  - b. Calcula la cantidad de elementos mayores al número que está en la posición 0.
  - c. Suma los elementos del arreglo entre las posiciones b y c.
  - d. Retorna el mayor valor que posee el arreglo A.
  - e. N.A.
- Si se ejecuta la función recursiva XXXX, considerando la llamada desde main, ¿Cuántas veces se ejecuta el algoritmo XXXX, si el arreglo A posee los enteros: <200, 40, 82, -1, 57, 1000, 32>?
  - a. 5 veces
  - b. 6 veces
  - c. 7 veces
  - d. 8 veces
  - e. N.A.

- Respecto a la pregunta anterior ¿Cuántas copias del arreglo  $A$  y de  $n$ , se crean en memoria, por cada llamada a la función recursiva XXXX?
  - a. 6 copias del arreglo  $A$ , 6 copias de  $n$ .
  - b. 7 copias del arreglo  $A$ , 7 copias de  $n$ .
  - c. 8 copias del arreglo  $A$ , 8 copias de  $n$ .
  - d. 1 copia del arreglo  $A$ , 1 copia de  $n$ .
  - e. 1 copia del arreglo  $A$ , 7 copias de  $n$ .
  - f. N.A.
- Si se ejecuta la función recursiva XXXX, y el arreglo  $A$  posee los enteros:  $\langle 8000, 200, 40, 82, 7435, 57, 134, 853, 246, 373, 1000, 32, 20232, 3446, 12, -10000, 0, 43 \rangle$ , que se imprime por pantalla?
  - a. El resultado es: 20232
  - b. El resultado es: 0
  - c. El resultado es: 8000
  - d. No se puede determinar pues no se conocen los valores de  $b$  ni de  $c$ .
  - e. N.A.

4. Considere la siguiente función recursiva “F1” y responda solo en el espacio entregado (*Prueba*).

```
int F1(int n) /*n debe ser mayor a 0*/
{
    if (n < 10)
        return 1;
    else
        return 1 + F1(n/10);
}
```

a. ¿Qué retornaría la función <b>F1</b> si recibe como parámetro a $n = 15$ ?	<b>R:</b>
b. ¿Qué retornaría la función <b>F1</b> si recibe como parámetro a $n = 94871$ ?	<b>R:</b>
c. ¿Cuántas veces se ejecutaría la función <b>F1</b> recibe como parámetro a $n = 94872341$ ?	<b>R:</b>
d. ¿Qué cálculo realiza la <b>F1</b> ?	<b>R:</b>



5. Complete las líneas que aparecen en los siguientes códigos, para lograr que cada programa realice la tarea que se indica: (*Prueba*)
- a. Que imprima por pantalla la suma de los elementos de un arreglo de números con decimales, llenado por el usuario.

```
float SumaArreglo(float *A, int n, int pos){
    if (pos >= n)
        return 0;
    else
        return _____ + SumaArreglo(A, n, _____);
}

int main()
{
    float *A;
    int n;

    /*En esta parte del main se le pide al usuario el tamaño del arreglo, se
    reserva memoria para crearlo, y se le pide al usuario que lo llene */

    printf("\nLa suma es: %.1f", SumaArreglo(A, n, 0));
    return 0;
}
```



- b. Que imprima por pantalla si un número entero ingresado por el usuario está o no en una matriz de enteros también llenada por el usuario.

```
int BuscaMatriz(int **Matriz, int m, int n, int valor, int i, int j)
{
    if (i == m)
        return 0;
    if (j == n)
        return BuscaMatriz(Matriz, m, n, valor, i+1, 1);

    if ( _____ )
        return 1;
    else
        return BuscaMatriz(Matriz, m, n, valor, _____, _____);
}

int main()
{
    int **Matriz, m, n, i, valor, resp;

    /*En esta parte del main se le pide al usuario las dimensiones de la matriz, se reserva
    memoria para crearla, y se le pide al usuario que llene la matriz.*/

    printf("Ingrese el numero que desea buscar: ");
    scanf("%d", &valor);

    resp = BuscaMatriz(Matriz, m, n, valor, 0, 0);

    if (resp)
        printf("\nEl elemento SI esta en la Matriz.");
    else
        printf("\nEl elemento NO esta en la Matriz.");

    printf("\n\n");
    return 0;
}
```

6. Construya una función **recursiva** en C que retorne la cantidad de divisores que posee un número *n* (recibido como parámetro). (*Prueba*)
7. Escribir una función recursiva que indique la cantidad de dígitos que posee un número entero.
8. Escribir una función recursiva que sume los dígitos que posee un número entero.

### III. PUNTEROS.

1. Dado la siguiente función en C:

```
int main()
{
    int a, b, *c, *d, e[5], f[5], i;

    for(i=0;i<5;i++)
    {
        e[i] = i;
        f[i] = 10 - i;
    }
    c = &a;
    d = &b;
    return 0;
}
```

Indique cuál es el resultado de las siguientes expresiones, considerando que el resultado de una NO INFLUYE en la siguiente expresión: ´

- b) `*c = *d;`
- c) `*d = 8;`
- d) `*(e+3) = b;`
- e) `d = f;`  
`*d = 5;`
- f) `d = &f[2];`  
`*(d+1) = 4;`

2. Escribir un programa que imprima cada uno de los elementos de un arreglo dos dimensional utilizando un puntero para acceder a los mismos, en lugar de utilizar subíndices. Utilizar el siguiente arreglo y los punteros indicados abajo:

```
int dos_vector[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
int *dos_ptr;
int (*ptr2vector)[4];
int fila, col;
```

3. Indique qué se imprime por pantalla en cada “printf” del siguiente extracto de código:

```
int y, *ip;
y = 12;

printf("\nValor de y %d, de ip %d", y, ip); /*sin asignar ip */

ip = &y;
*ip = *ip + 10;
printf("\nValor de y %d, de *ip %d y de ip %d", y, *ip, ip);

y = *ip + 10;
printf("\nValor de y %d, de *ip %d", y, *ip);

*ip += 1;
printf("\nValor de y %d, de *ip %d", y, *ip);
```

4. Considere el siguiente código en C, y suponga que las variables declaradas en el programa, se crearon en la Memoria Principal (RAM), como muestra la figura de la derecha. (*Prueba*)

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int aux, cont;
    int *p, *q, *r;

1   aux = 0;
2   cont = 1;
3   q = (int *)malloc(sizeof(int));
4   p = &aux;
5   aux = 20;
6   r = &cont;
7   aux = cont;
8   *q = 50;
9   r = q;
10  q = p;
11  r = q
12  printf("\np: %d, *q: %d, *r: %d", *p, *q, *r);
13  printf("\np: %p, q: %p, r: %p", p, q, r);
    printf("\n");
    return 0;
}
```

RAM		
aux	1	
	2	
cont	3	
	4	
p	5	
	6	
q	7	
	8	
r	9	
	10	
	11	

a. Complete la siguiente tabla:

Línea	aux	cont	p	q	r	*(dir 11)
1						
2						
3				11		
4						
5						
6						
7						
8						
9						
10						
11						

b. Señale qué imprime la línea 12 del código.

c. Señale qué imprime la línea 13 del código.

5. Considere el siguiente programa en C. Bajo él, aparece un esquema de la Memoria Principal (RAM): En ella se señala dónde se creó cada variable. (*Prueba*).

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 4

int main()
{
    int a, b, *p, *q, *r, *t;

1   a = 15;
2   b = 35;
3   p = (int *)malloc(MAX*sizeof(int));
4   q = &a;
5   r = q;
6   *q = 10;
7   t = r;
8   *t = 5;
9   *r = 0;
10  *p = b;
11  *(p+1) = *q;
12  *(p+2) = *r;
13  *(p+3) = *t;
14  printf("%d,%d,%d,%d,%d,%d,%d",
        *p, *(p+1), *(p+2), *(p+3), *q, *r, *t);
    printf("\n");
    return 0;
}

```

**Se pide:**

Complete la tabla de abajo, que representa a la RAM cuando se ejecuta el programa entregado.

Fíjese que cada columna corresponde a la ejecución de una línea de código (señalada al final de cada columna).

Variables	Dir	RAM												
	↓													
<b>a</b>	1													
<b>b</b>	2													
	3													
<b>p</b>	4			9										
<b>q</b>	5													
<b>r</b>	6													
<b>t</b>	7													
	8													
	9													
	10													
	11													
	12													
	13													
Línea del Código →		1	2	3	4	5	6	7	8	9	10	11	12	13

- Señale qué imprime la línea 14 del código.

6. Considere el siguiente programa en C. A la derecha encontrará un esquema de la RAM donde quedaron las variables definidas al ejecutarlo. (*Prueba*)

```
#include <stdio.h>

int main()
{
    int P=100, Q=200, R=300;
    int *p1, *p2, *p3;

1   printf("\n0. %X %X %X", &P, &Q, &R);
2   printf("\n1. %X %X %X", &p1, &p2, &p3);
3   p1 = &P;
4   p2 = (int *)malloc(sizeof(int));
5   p3 = (int *)malloc(sizeof(int));
6   *p2 = Q;
7   *p3 = R;
8   printf("\n2. %X %X %X", &p1, &p2, &p3);
9   printf("\n3. %X %X %X", p1, p2, p3);
10  printf("\n4. %d %d %d", *p1, *p2, *p3);
11  (*p1)++;
12  (*p2)++;
13  (*p3)++;
14  printf("\n5. %d %d %d", *p1, *p2, *p3);
15  printf("\n6. %d %d %d", P, Q, R);
16  printf("\n7. %X %X %X", p1, p2, p3);
    return 0;
}
```

	dir	RAM
	0	
P	1	
	2	
Q	3	
	4	
R	5	
	6	
	7	
	8	
	9	
	A	
	B	
	C	
p1	D	
	E	
	F	
p2	10	
	11	
	12	
p3	13	
	14	
	15	

a. Complete la siguiente tabla:

Línea de Código	A	B	C	p1	p2	p3	*p1	*p2	*p3
3									
4					8				
5						A			
6									
7									
11									
12									
13									

b. Señale qué se imprime en las líneas de código mencionadas abajo:

Línea de Código	¿Qué se imprime por pantalla?
1	0.
2	1.
8	2.
9	3.
10	4.
14	5.
15	6.
16	7.

7. Considere el siguiente programa en C y responda las preguntas de la derecha. Sólo use ese espacio. (*Prueba*)

```
#include <stdio.h>
int main() {
    int A[MAX], i, *ptro1, *ptro2, *ptro3;

1   for(i=0;i<MAX; i++)
2       A[i] = 2*i;
3   for(i=0;i<MAX; i++)
4       printf("%d\t", A[i]);
5   ptr01 = A;
6   ptr02 = &A[MAX-1];
7   ptr03 = (int *)malloc(sizeof(int));
8   for(i=0;i<MAX/2; i++)
9   {
10      *(ptr03) = *(ptr01+i);
11      *(ptr01+i) = *(ptr02 - i);
12      *(ptr02 - i) = *(ptr03);
13  }
14  for(i=0;i<MAX; i++)
15      printf("%d\t", A[i]);
16
    return 0;
}
```

a. Si MAX es 8, ¿Qué se imprime en la línea 4?

b. Si MAX es 2000 y se acaba de ejecutar la línea 6, ¿Qué valor posee \*ptr02?

c. Si MAX es 1000000 y se acaba de ejecutar la línea 5, ¿&ptr01 == &ptr02?(Responda sólo SI o NO)

d. Si MAX = 8, ¿Qué se imprime en la línea 15?