1. Estructuras de control repetitivas

Al finalizar la presente sesión de laboratorio, el estudiante debe ser capaz de

- Almacenar datos numéricos del mismo tipo en un arreglo
- Modificar el orden de ejecución de las sentencias de un programa a través de sentencias `for`
- Implementar instrucciones repetitivas empleando sentencias `for`
- Anidar una sentencia `for` dentro de otra

Arreglos numéricos. Un arreglo es un grupo indexado de lugares de almacenamiento de datos que tienen el mismo nombre y se distinguen unos a otros por un subíndice, un número entre corchetes que sigue al nombre de la variable.

1.1. La sentencia `for`

`for` (*ini* ; *condi* ; *salto*)
{
sentencia ;
}

1. Se ejecuta la sentencia *ini*.
2. La expresión *condi* es evaluada.
   a) Si *condi* se evalúa como verdadero, *sentencia* es ejecutada y luego *salto*; después, se vuelve a 2.
   b) Si *condi* se evalúa como falso, la sentencia `for` termina.

N.B.: Ambas líneas: `if` (*ini* ; *condi* ; *salto*) y *sentencia* ; se consideran que constituyen la sentencia `for` completa. No son sentencias separadas.

Una sentecia `for` puede controlar la ejecución de múltiples sentencias, a través del uso de una **sentecia compuesta** o **bloque**.

Una sentencia `for` anidada dentro de otra es cuando una sentecia `for` está dentro de otra.

Ejercicio 1.1. Escriba un programa que pida ingresar un entero positivo desde el teclado y muestre si dicho número es primo o no.

Ejercicio 1.2. Implemente un programa que pida ingresar dos enteros positivos desde el teclado y muestre el máximo común divisor de ellos empleando un algortimo que NO sea el de Euclides.

Ejercicio 1.3. Cree un programa que pida ingresar dos enteros positivos desde el teclado y muestre el máximo común divisor de ellos empleando el algoritmo de Euclides.

Ejercicio 1.4. Escriba un programa que pida ingresar un entero n mayor que y dibue un triángulo de altura $2n-1$ y ancho n. Por ejemplo, para $n=4$ se tendrá:

```
*
**
***
****
***
**
*
```

Ejercicio 1.5. Pida ingresar un entero $n > 1$ dibuje un triángulo de altura $2n-1$ y ancho n. Por ejemplo, para $n=2$ se tendría:

```
*
* *
*
```

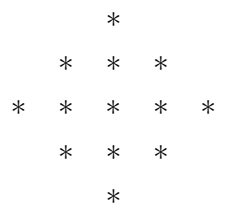
Ejercicio 1.6. Pida ingresar un entero $n > 1$ dibuje un triángulo de altura n y ancho $2n-1$. Por ejemplo, para $n=2$ se tendría:

```
  *
* * *
```

Ejercicio 1.7. Pida ingresar un entero n mayor que uno y dibuje un reloj de arena hechado de altura $2n+1$. Por ejemplo, para $n=2$ se tendrá:

```
*       *
* *   * *
* * * * *
* *   * *
*       *
```

Ejercicio 1.8. Cree un programa que pida ingresar número impar n mayor que 1 y dibuje un rombo cuyas diagonales miden n. Por ejemplo, para $n=5$ se tendrá:



Ejercicio 1.9. Implemente un programa que pida ingresar un entero n mayor que uno y dibuje la siguiente figura del modo detallado abajo. Por ejemplo, para $n = 2$ y $n = 3$ se tendrá, respectivamente:

Ejercicio 1.10. Se pide ingresar un margen de error ($\epsilon > 0$) y se muestre el valor de la primera suma parcial que con un margen de error de ϵ se aproxime al valor de la siguiente serie:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots = \ln(2).$$

Ejercicio 1.11. Cree un programa que pida ingresar 6 números enteros. Luego determine y muestre el mayor y menor de ellos.

Ejercicio 1.12. Indique lo que se mostrará en la pantalla al ejecutar cada programa y explique por qué.

1.
```c
#include <stdio.h>

int main ( )
{
    int i, j;

    for(j = 2 ; j >= -2 ; j--)
    {
            for(i = -2 ; i <= 2 ; i++)
                {
                    if (-abs(i) <= j && j <= abs(i))
                                printf("*");
                        else
```

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

2.
```
#include <stdio.h>

int main ( )
{
        int x, y;

        for(y = 2 ; y >= -2 ; y--)
        {
                for(x = -2 ; x <= 2 ; x++)
                {
                if (abs(x) + abs(y) <= 2)
                                printf(" ");
                    else
                                printf("*");
                }
                printf("\n");
        }
        return 0;
}
```

Ejercicio 1.13. Indique lo que se mostrará en la pantalla (y explique por qué) al ejecutar el siguiente programa e ingresar:

1. el valor de 4,
2. el valor de 7 y
3. el valor de 10

```
#include <stdio.h>

int main ( )
{
        unsigned int cont = 0, i, N;
```

```
        printf (" Ingrese un entero mayor que 1: ");
        scanf ("%u", &N);

        for (i = 1 ; i <= N ; i++)
                if (N % i == 0)  cont = cont + 1;

        if (cont == 2)
                printf (" %u es primo.\n", N);
        else
                printf (" %u no es primo.\n", N);
}
```

Ejercicio 1.14. Indique lo que se mostrará en la pantalla al ejecutar cada programa y explique por qué.

1. ```
   #include <stdio.h>
   #define pi 3.14

   int main ( )
   {
           int i;
           float radios[5] = {1.0, 2.0, 3.0, 4.0, 5.0};

           for(i = 0 ; i < 5 ; i++)
           {
                   printf(" El circulo de radio %.1f", radios[i]);
                   printf (" tiene perimetro %.2f \n", 2*pi*radios[i]);
           }
   }
   ```

2. ```
   #include <stdio.h>

   int main ( )
   {
           unsigned int i, fibonacci[5];

           fibonacci[0] = 1;
           fibonacci[1] = 1;

           for(i = 2 ; i < 5 ; i++)
                   fibonacci[i] = fibonacci[i-1] + fibonacci[i-2];
   ```

```
        for(i = 0 ; i < 5 ; i++)
                printf (" fibonacci[%u] = %u \n", i, fibonacci[i]);
}
```

Ejercicio 1.15. José desea formar un triángulo con monedas, que se describe como sigue:

O
OO
OOO
OOOO

Pepito está interesado en formar un triángulo con la máxima altura posible. Implemente un programa que lee la cantidad de monedas disponibles y muestre la altura del triángulo más alto que José puede formar con dicha cantidad de monedas. Por ejemplo, con cinco monedas el triángulo con la máxima altura posible sería

O
OO

Mientras que con siete monedas sería

O
OO
OOO