

**UNIVERSIDAD AUTÓNOMA DE BAJA  
CALIFORNIA**  
**Facultad de Ingeniería, Arquitectura y Diseño**

**Ingeniero en Software y Tecnologías Emergentes**



**Nombre Alumno:**  
Eliel Alfonso Ontiveros Ojeda

**Grupo:**  
932

**Practica #6**  
Estructuras de Control Repetitivas

06/09/2023

## Introducción

Las estructuras de control repetitivas, también conocidas como estructuras iterativas, son agrupaciones de código diseñadas para repetir un conjunto de declaraciones relacionadas. Esta repetición (o iteración) puede repetirse cero o más veces, hasta que algún valor de control o condición haga que la repetición cese. Usamos controles iterativos cuando necesitamos hacer la misma tarea más de una vez, basándonos en alguna condición lógica.

Las estructuras de control repetitivas se pueden categorizar en dos tipos generales: bucles indeterminados y bucles determinados. La diferencia radica en que, con una estructura de bucle determinado, normalmente se puede predecir exactamente cuántas veces se repetirá el bucle; mientras que, con una estructura de bucle indeterminado, este no siempre es el caso.

Un bucle indeterminado se repite mientras una condición se evalúa a un cierto valor booleano. Los bucles indeterminados deben usarse cuando el programador no sabe exactamente cuántas veces necesita ocurrir la iteración. Lógicamente, sin embargo, tanto los bucles indeterminados como los determinados pueden escribirse para ser equivalentes.

## Competencia

1. **Comprensión de la lógica de control:** Entender cómo funcionan los bucles y cuándo usarlos.
2. **Desarrollo de eficiencia:** Aprender a usar bucles para evitar la repetición innecesaria de código.
3. **Resolución de problemas:** Aplicar bucles para resolver problemas complejos en programación.


## Fundamentos

Las estructuras de control repetitivas, o bucles, permiten repetir un conjunto de declaraciones. Los bucles pueden ser indeterminados (se repiten mientras una condición sea verdadera) o determinados (se repiten un número específico de veces). Los bucles indeterminados pueden ser pre-test (evalúan la condición antes de ejecutar el cuerpo del bucle) o post-test (evalúan la condición después de ejecutar el cuerpo del bucle).

## Procedimiento

1. PROGRAMA QUE PREGUNTE LA CANTIDAD DE VECES QUE DESEA QUE SE REALICE EL PROGRAMA DE FIBONACCI.


```
1 void menuFibonacci()
2 {
3     int op;
4
5     start:
6     printf("Menu Fibonacci:\n");
7     printf("1 - For\n");
8     printf("2 - While\n");
9     printf("3 - Do While\n");
10    printf("Selecciona una opcion\n");
11    scanf("%i", &op);
12
13    switch (op)
14    {
15        case 1:
16            fibonacciFor();
17            break;
18        case 2:
19            fibonacciWhile();
20            break;
21        case 3:
22            fibonacciDoWhile();
23            break;
24        default:
25            printf("Opcion no valida");
26            goto start;
27            break;
28    }
29 }
```



```

1 void fibonacciFor()
2 {
3     int num;
4     int n1 = -1, n2 = 1, aux = n1+n2;
5
6     printf("Numero para la serie fibonacci\n");
7     scanf("%i", &num);
8     for (int i = 0; i < num ; i++)
9     {
10         aux = n1 + n2;
11         n1 = n2;
12         n2 = aux;
13         printf("%i\n", aux);
14     }
15
16 }


```



```

1 void fibonacciWhile()
2 {
3     int num;
4     int n1 = -1, n2 = 1, aux = n1+n2;
5
6     printf("Numero para la serie fibonacci\n");
7     scanf("%i", &num);
8
9     int i = 0;
10    while (i < num)
11    {
12        aux = n1 + n2;
13        n1 = n2;
14        n2 = aux;
15        printf("%i\n", aux);
16        i++;
17    }
18 }

```



```

1 void fibonacciDoWhile()
2 {
3     int num;
4     int n1 = -1, n2 = 1, aux = n1+n2;
5
6     printf("Numero para la serie fibonacci\n");
7     scanf("%i", &num);
8
9     int i = 0;
10    do
11    {
12        aux = n1 + n2;
13        n1 = n2;
14        n2 = aux;
15        printf("%i\n", aux);
16        i++;
17    } while (i < num);
18 }

```

2. PROGRAMA QUE PIDA UN NÚMERO Y DESPLEGAR LA SALIDA DE FACTORIAL DE UN NÚMERO DADO.

```
1 void menuFactorial()
2 {
3     int op;
4
5     start:
6     printf("Menu Factorial:\n");
7     printf("1 - For\n");
8     printf("2 - While\n");
9     printf("3 - Do While\n");
10    printf("Selecciona una opcion\n");
11    scanf("%i", &op);
12
13    switch (op)
14    {
15        case 1:
16            factorialFor();
17            break;
18        case 2:
19            factorialWhile();
20            break;
21        case 3:
22            factorialDoWhile();
23            break;
24        default:
25            printf("Opcion no valida");
26            goto start;
27            break;
28    }
29 }
```



```
1 void factorialFor()
2 {
3     int result = 1, num;
4
5     printf("Numero a sacar el factorial\n");
6     scanf("%i", &num);
7
8     for (int i = 0; i <= num; i++)
9     {
10         result = result * i;
11     }
12
13     printf("El factorial del numero %i es = %i", num, result);
14 }
```



```
1 void factorialWhile()
2 {
3     int result = 1, num;
4
5     printf("Numero a sacar el factorial\n");
6     scanf("%i", &num);
7
8     int i = 1;
9     while (i <= num)
10    {
11        result = result * i;
12        i++;
13    }
14
15    printf("El factorial del numero %i es = %i", num, result);
16 }
```



```
1 void factorialDoWhile()
2 {
3     int result = 1, num;
4
5     printf("Numero a sacar el factorial\n");
6     scanf("%i", &num);
7
8     int i = 1;
9     do
10    {
11        result = result * i;
12        i++;
13    } while (i <= num);
14
15    printf("El factorial del numero %i es = %i", num, result);
16 }
```

3. PROGRAMA QUE PIDA UN NÚMERO Y DESPLEGAR LA CANTIDAD DE DÍGITOS QUE TIENE EL NÚMERO.

```
1 void menuDigitCounter()
2 {
3     int op;
4
5     start:
6     printf("Menu Contar Digitos:\n");
7     printf("1 - For\n");
8     printf("2 - While\n");
9     printf("3 - Do While\n");
10    printf("Selecciona una opcion\n");
11    scanf("%i", &op);
12
13    switch (op)
14    {
15        case 1:
16            digitCounterFor();
17            break;
18        case 2:
19            digitCounterWhile();
20            break;
21        case 3:
22            digitCounterDoWhile();
23            break;
24        default:
25            printf("Opcion no valida");
26            goto start;
27            break;
28    }
29 }
```



```
1 void digitCounterFor()
2 {
3     int num, aux, dig;
4     int i = 0;
5
6     printf("Numero a contar los digitos\n");
7     scanf("%i", &num);
8
9     aux = num;
10    for (i; aux > 0; i++)
11    {
12        aux = aux / 10;
13    }
14
15    printf("%i", i);
16 }
```



```
1 void digitCounterWhile()
2 {
3     int num, aux, dig;
4
5     printf("Numero a contar los digitos\n");
6     scanf("%i", &num);
7
8     aux = num;
9     int i = 0;
10    while (aux > 0)
11    {
12        aux = aux / 10;
13        i++;
14    }
15
16    printf("%i", i);
17 }
```



```
1 void digitCounterDoWhile()
2 {
3     int num, aux, dig;
4
5     printf("Numero a contar los digitos\n");
6     scanf("%i", &num);
7
8     aux = num;
9     int i = 0;
10    do
11    {
12        aux = aux / 10;
13        i++;
14    } while (aux > 0);
15
16    printf("%i", i);
17 }
```