

```
1 #include <stdio.h>
2 #include <string.h>
3
4 // Programación I - Ciclos.
```

```
5
6 int Ciclos()
```

```
7 {
```

```
8
9     char prof[20];
```

```
10
11     strcpy(prof, "Miguel Silva C.");
```

```
12
13     printf("Prof. %s", prof);
```

```
14
15
16
17     return 0;
```

```
18
19 }
20
21
22
23
```



```
1 // Programación I - Ciclos.
```

2 3 4 Concepto Ciclos() {

5
6 Son estructuras de control que
7
8 permiten ejecutar un bloque de
9
10 código repetidamente mientras se
11
12 cumpla una condición específica, o
13
14 un número determinado de veces. Son
15
16 fundamentales para la implementación
17
18 de la repetición y la iteración en
19
20 un programa.

```
21 };
```



```
1 // Programación I - Ciclos
```

```
2  
3 En C, los ciclos más comunes son:
```

```
4  
5 while(condición)
```

```
6 {
```

```
7  
8 While: El bucle while ejecuta  
9 un bloque de código mientras  
10 se cumpla una condición  
11 específica.  
12  
13
```

```
14 La condición se evalúa antes de cada  
15 iteración.  
16  
17
```

```
18  
19 };
```



```
// Programación I - Ciclos
```

En C, los ciclos más comunes son:

```
do  
{
```

do-while: Similar a *while*, pero la condición se evalúa después de cada iteración.

Esto garantiza que el código se ejecute al menos una vez, incluso si la condición es falsa desde el principio.

```
}While(condición);
```



En C, los ciclos más comunes son:

```
for(inicialización; condición; actualización)
```

```
{
```

For: Especifica un contador de iteraciones, una condición de fin y una expresión de actualización en una sola línea.

Útil cuando se sabe cuántas veces se debe repetir el bloque de código.

```
};
```



```
1 // Programación I - Ciclos
```

Ejemplo while:

```
2
3
4
5 #include <stdio.h>
6
7
8
9
10 int main()
11 {
12     int valor, contador;
13
14     printf("Ingrese el valor para contar: ");
15     scanf("%d", &valor);
16
17     contador = 1; //Se inicializa el contador.
18
19     while(contador <= valor)
20     {
21         // Se imprime el contenido del
22         // contador en una sola línea.
23         printf("%d ", contador);
24         // Se incrementa en 1 el contador;
25         contador++;
26     }
27
28     return 0;
29 }
```

Código

Salida

```
Ingrese el valor para contar: 10
1  2  3  4  5  6  7  8  9  10
...Program finished with exit code 0
Press ENTER to exit console.
```



// Programación T - Ciclos

Ejemplo do-while:

Código

```
1 // Programación T - Ciclos
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int valor, contador;
8
9     printf("Ingrese el valor para contar: ");
10    scanf("%d", &valor);
11
12    contador = 1; //Se inicializa el contador.
13
14    do
15    {
16        // Se imprime el contenido del
17        // contador en una sola línea.
18        printf("%d ", contador);
19        // Se incrementa en 1 el contador;
20        contador++;
21    }while(contador <= valor);
22
23    return 0;
24 }
```

Salida

```
Ingrese el valor para contar: 10
1  2  3  4  5  6  7  8  9  10

...Program finished with exit code 0
Press ENTER to exit console. □
```



Ejemplo for:

Código

```
1 // Programación I - Ciclos
2
3
4
5 #include <stdio.h>
6
7 int main()
8 {
9     int valor, contador;
10
11     printf("Ingrese el valor para contar: ");
12     scanf("%d", &valor);
13
14     for(contador=1; contador<=valor; contador++)
15     {
16         // Se imprime el contenido del
17         // contador en una sola línea.
18         printf("%d ", contador);
19     }
20
21     return 0;
22 }
23
```

Salida

```
Ingrese el valor para contar: 10
1  2  3  4  5  6  7  8  9  10
...Program finished with exit code 0
Press ENTER to exit console.
```



1 // Programación T - Ciclos

2
3 Ejemplo 2, while:

4 #include <stdio.h>

Código

5
6
7 int main() {

8 int contador = 0;

9
10
11 printf("Ejemplo de ciclo while:\n");

12 while (contador < 5) {

13 printf("Contador: %d\n", contador);

14 contador++;

15
16 }

17
18
19 return 0;

20
21
22 }
23

Salida

Ejemplo de ciclo while:

Contador: 0

Contador: 1

Contador: 2

Contador: 3

Contador: 4



1 // Programación T - ~ los
2
3 Ejemplo 2, do-while:

Código

```
5 #include <stdio.h>
6
7 int main() {
8     int contador = 0;
9
10
11     printf("Ejemplo de ciclo do-while:\n");
12     do {
13         printf("Contador: %d\n", contador);
14         contador++;
15     } while (contador < 5);
16
17     return 0;
18
19
20
21 }
22
23
```

Salida

```
Ejemplo de ciclo do-while:
Contador: 0
Contador: 1
Contador: 2
Contador: 3
Contador: 4
```



1 // Programación T - ~ los
2
3 Ejemplo 2, do-while:

Código

```
5  
6 #include <stdio.h>  
7  
8 int main() {  
9     printf("Ejemplo de ciclo for:\n");  
10    for (int contador = 0; contador < 5; contador++) {  
11        printf("Contador: %d\n", contador);  
12    }  
13  
14  
15    return 0;  
16  
17 }
```

Salida

```
Ejemplo de ciclo for:  
Contador: 0  
Contador: 1  
Contador: 2  
Contador: 3  
Contador: 4
```



EJERCITACIÓN (Level 1):

1. Crear un menú con todos los ejercicios de las prácticas anteriores para que pueda ejecutarse tantas veces como se requiera.
2. Crear un programa que calcule la suma de los primeros N números naturales.
3. Calcular el factorial de un número ingresado por el usuario.
4. Imprimir una tabla de multiplicar para un número ingresado por el usuario ($1*n$, $2*n$, $3*n$... $n*n$).
5. Encontrar el máximo y el mínimo de un conjunto de números ingresados por el usuario.
6. Imprimir los números pares del 1 al N (ingresado por el usuario).
7. Imprimir los números impares del 1 al N (ingresado por el usuario).
8. Ingresar un número entero e indicar cuántos dígitos tiene.



EJERCITACIÓN (Level 2):

1. Calcular e imprimir la suma de los dígitos pares e impares de un número entero ingresado por el usuario.
2. Imprimir todos los números primos dentro de un rango dado por el usuario.
3. Imprimir los primeros N términos de la serie de Fibonacci.
4. Determinar si un número natural ingresado por el usuario es primo.
5. Imprimir la secuencia de Collatz para un número ingresado por el usuario.
6. Determinar si un número entero ingresado por el usuario es un número de Armstrong (o narcisista).
7. Calcular e imprimir la suma de los N primeros términos de la serie armónica.
8. Determinar si un número entero ingresado por el usuario es un número perfecto.



EXPLICACIÓN (Level 2):

1. Fibonacci:

- Explicación: La secuencia de Fibonacci es una serie de números en la que cada número es la suma de los dos números anteriores. Comienza con 0 y 1, y los siguientes números son la suma de los dos números anteriores.
- Fórmula: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... F_n

2. Collatz:

- Explicación: La secuencia de Collatz, también conocida como conjetura de Collatz, es una secuencia generada a partir de un número entero siguiendo ciertas reglas. Si el número es par, se divide por 2; si es impar, se multiplica por 3 y se suma 1.
- Fórmula: No hay una fórmula matemática explícita para la secuencia de Collatz. Se define de manera recursiva como
$$a_{n+1} = \begin{cases} a_n/2, & \text{si } a_n \text{ es par} \\ 3a_n + 1, & \text{si } a_n \text{ es impar} \end{cases}$$

3. Armstrong o narcisista:

- Explicación: Un número de Armstrong (o narcisista) es un número en el que la suma de las potencias de sus dígitos es igual al propio número.
- Fórmula: Un número de Armstrong de n dígitos, $abcd... = a^n + b^n + c^n + d^n + ...$



EXPLICACIÓN (Level 2):

4. Número armónico:

- Explicación: La serie armónica es una serie infinita en la que cada término es la inversa del número natural correspondiente.
- Fórmula: La serie armónica se representa como $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$.

5. Número perfecto:

- Explicación: Un número perfecto es un número entero que es igual a la suma de sus divisores propios positivos, excluyendo el número mismo.
- Fórmula: Un número perfecto N es aquel para el cual la suma de sus divisores propios (D) es igual a N , es decir, $D_1 + D_2 + \dots + D_n = N$, donde D es un divisor propio de N .



REALIZADO POR MIGUEL SILVA C.

 miguel.silva@inspt.utn.edu.ar

© Esta presentación cuenta con derechos de autor.

