Bucles

For

Estructura

```
for(initialization; condition; step)
{
    code_block
}
```

Aplicaciones

- útil cuándo se desea ejecutar un bloque de código un número específico de veces.
- Para recorrer elementos iterables (listas, arrays, ...)

Ejemplo ejercicios

- Sumatoria

$$\sum_{i=1}^{N} \frac{(-1)^i X^{2i}}{(2i)!}$$

- Productorio

$$\prod_{i=1}^{N} \frac{(-1)^i X^{2i}}{(2i)!}$$

- Recorrer el array `a = {'h', 'o', 'l', 'a'    }`

While

Estructura

```
while(condition)
{
    code_block
}
```

```
while(true)
{
    x += 1
    if(x == 4){
        break;
    }
}
```

Aplicaciones

- Ejecutar un bloque de código mientras se cumpla una condición.
- Se usa en casos donde no se sabe la cantidad exacta de iteraciones que se deben realizar.

Do While

Estructura

```
do
{
```

```
    code_block
} while(condition);
```

Aplicaciones

- Se usa en casos donde no se sabe la cantidad exacta de iteraciones que se deben realizar.
- Usar cuando se necesita que el bloque de código se ejecute al menos una vez.

Bucles anidados

Pueden estar conformados por todos los tipos de bucles vistos previamente (`for`, `while` y `do while`)

- Sumatoria

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{(-1)^i X^{2i}}{(2i)!}$$

$$\sum_{i=1,j=1}^{N}\frac{(-1)^i X^{2i}}{(2i)!}$$

```
for(int i = 1, int j = 1; condition; i ++, j++)
{
    code_block
}
```

```
for(int i = 1, int j = 1; condition; step)
{
    for(initialization; condition; step)
    {
        code_block
    }
}
```

```
while(condition)
{
    while(condition)
    {
        code_block
    }
}
```

```
do
{
    do
    {
        code_block
    } while(condition);
} while(condition);
```

Aplicaciones

- Utilizar cuando se desean recorrer estructuras con más de una dimención (en este caso la mejor opción es un `for`).
- Evitar el uso exesivo de las mismas, la complejidad es exponencial.
- Para mitigar la complejidad de bucles anidados, usar `break` y `continue`.

Break / Continue

Break

Esta sentencia termina el bucle más interno

```
for(int i = 0; i <10; i++)
{
  cout << i << "\n";
  if (i == 4) {
    break;
  }
}
```

Continue

Le indica al bucle que inicie la operación de la siguiente iteración, esto le permite saltarce la ejecución del bloque de código para casos específicos.

```
for(int i = 0; i <10; i++)
{
  if (i == 4) {
    continue;
  }
  cout << i << "\n";
}
```