

Ensenada, Baja California a 05 de octubre del 2020



Practica No.3

Estructuras repetitivas

Profesor: Roilhi Frajo Ibarra Hernández

Alumno: Fabian Diaz Fajardo

Grupo: 021

## **Introducción.**

Un ciclo es una parte del programa con estructura repetitiva. Una serie de instrucciones se ejecuta una y otra vez hasta que se alcanza cierto objetivo o condición. Los ciclos son importantes para cálculos que requieren pasos repetitivos y el procesamiento constante de varias entradas de datos. En esta práctica se aprenderá a implementar ciclos en C++ así como escribir programas que implementen estructuras repetitivas en el mundo real.

## **Objetivos.**

- Desarrollar y codificar en C++ estructuras de repetición: ciclos “while” y “do while”.
- Encontrar errores en ciclos: ciclos infinitos, condiciones no cumplidas, etc
- Comprender ciclos anidados
- Implementar programas que lean y procesen varios datos
- Resolver problemas matemáticos y de ingeniería donde se implementen las estructuras de control de selección por medio de la codificación de un programa en C++.

## Desarrollo de la práctica.

### 4.1. Implementaciones de ciclos.

4.1.1.- Para las siguientes líneas de código genere una tabla con los valores de i, j y n en cada iteración:

a) `int i=0; int j=10; int n=0;`

`while (i<j){i++; j--; n++;}`

Taller 3

4.1 Implementaciones de ciclos

4.1.1.- Para las siguientes líneas de código genere una tabla con los valores de i, j y n en cada iteración:

a) `int i=0; int j=10; int n=0;`  
`While (i<j){i++; j--; n++;}`

| i | j  | n | (i<j) |
|---|----|---|-------|
| 0 | 10 | 0 | ✓     |
| 1 | 9  | 1 | ✓     |
| 2 | 8  | 2 | ✓     |
| 3 | 7  | 3 | ✓     |
| 4 | 6  | 4 | ✓     |
| 5 | 5  | 5 | X     |
| 6 | 4  | 6 | X     |

b) `int i=0; int j=0; int n=0;`

b) `int i=0; int j=0; int n=0;`

`while (i<10){i++; n=n+i; j++;}`

b) `int i=0; int j=0; int n=0;`  
`while (i<10){i++; n=n+i; j++;}`

| i  | n  | j  | (i<10) |
|----|----|----|--------|
| 0  | 0  | 0  | ✓      |
| 1  | 1  | 1  | ✓      |
| 2  | 3  | 2  | ✓      |
| 3  | 6  | 3  | ✓      |
| 4  | 10 | 4  | ✓      |
| 5  | 15 | 5  | ✓      |
| 6  | 21 | 6  | ✓      |
| 7  | 28 | 7  | ✓      |
| 8  | 36 | 8  | ✓      |
| 9  | 45 | 9  | X      |
| 10 | 55 | 10 | X      |
| 11 | 66 | 11 | X      |

c) `int i=0; int j=10; int n=0;`

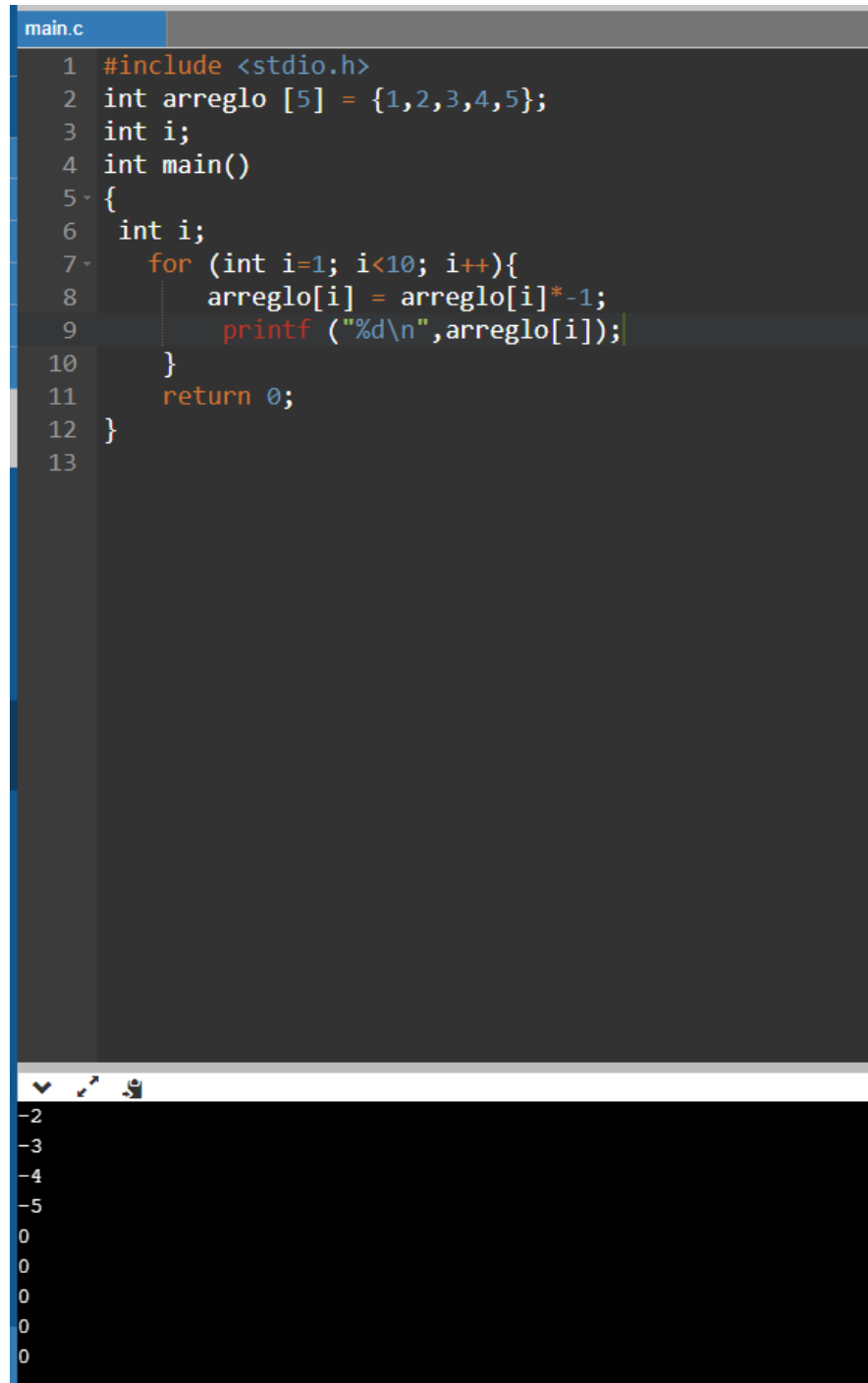
`while (i!=j){i=i+2; j=j-2; n++;}`

c) `int i=0; int j=10; int n=0;`  
`while (i!=j){i=i+2; j=j-2; n++;}`

| i  | j  | n | (i!=j) |
|----|----|---|--------|
| 0  | 10 | 0 | ✓      |
| 2  | 8  | 1 | ✓      |
| 4  | 6  | 2 | ✓      |
| 6  | 4  | 3 | ✓      |
| 8  | 2  | 4 | ✓      |
| 10 | 0  | 5 | ✓      |
| 12 | -2 | 6 | ✓      |
| 14 | -4 | 7 | ✓      |
| 16 | -6 | 8 | ✓      |
| 18 | -8 | 9 | ✓      |
| ∞  | ∞  | ∞ | ∞      |

4.1.2.- ¿Qué es lo que imprimirán en pantalla a la salida las siguientes líneas de código?

d) for (int i=1; i<10; i++){ cout << i <<" ";



```
main.c
1  #include <stdio.h>
2  int arreglo [5] = {1,2,3,4,5};
3  int i;
4  int main()
5  {
6      int i;
7      for (int i=1; i<10; i++){
8          arreglo[i] = arreglo[i]*-1;
9          printf ("%d\n",arreglo[i]);
10     }
11     return 0;
12 }
13
```

The output of the program is shown in the terminal window below the code editor:

```
-2
-3
-4
-5
0
0
0
0
0
0
```

e) `for (int i=1; i<10; i=i*2){ cout << i <<" ";}`

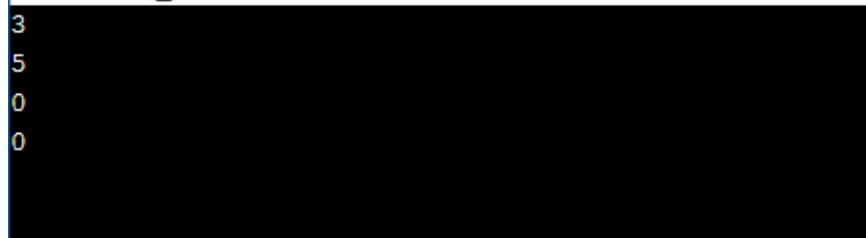
```
main.c
1  #include <stdio.h>
2  int arreglo [5] = {1,2,3,4,5};
3  int i;
4  int main()
5  {
6      int i;
7      for (int i=1; i<10; i=i*2){
8          printf ("%d\n",arreglo[i]);
9      }
10     return 0;
11 }
12
```

2  
3  
5  
0

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

f) for (int i=1; i<10; i++){ if(i%2==0){ cout << i <<" ";}}

```
main.c
1  #include <stdio.h>
2  int arreglo [5] = {1,2,3,4,5};
3  int i;
4  int main()
5  {
6      int i;
7      for (int i=1; i<10; i++){
8          if (i%2==0)
9              printf ("%d\n",arreglo[i]);
10         }
11         return 0;
12     }
13
```



```
3
5
0
0
```



- ¿Qué es un bucle infinito o programa ciclado y qué consecuencias implica en una ejecución?

Bucle infinito en programación es un error que consiste en realizar un ciclo que se repite de forma indefinida ya que su condición para finalizar nunca se cumple.

La consecuencia es que jamás terminaría el programa.

- ¿Qué es un “error por uno” (off-by-one) en programación y qué consecuencias implica?

Un error por uno o error por un paso (en inglés: Off-by-one error -OBOE) es un error de lógica que es el equivalente discreto de una condición de contorno. Ocurre a menudo en la programación de computadoras cuando un bucle iterativo se repite una vez más o una vez menos que lo necesario. Este problema puede presentarse, por ejemplo, cuando un programador comete errores como el uso de la expresión "es menor que o igual a", cuando en realidad debería haber utilizado "es menor que" en una comparación o no toma en cuenta que una secuencia comienza en el cero en lugar de en uno (como por ejemplo con los índices de matriz en varios lenguajes). Esto también puede ocurrir en un contexto matemático.

4.2. Escriba un programa que obtenga los dígitos binarios de un número decimal por medio del cálculo de residuos.

```
1  #include <stdio.h>
2  int main()
3  {
4      int Binario[100];
5      int i=0, j=0;
6      int numero;
7      printf ("Dame el numero: ");
8      scanf ("%d",&numero);
9
10     for (i=100; i>0; i--) {
11         Binario[i]=numero%2;
12         numero=numero/2;
13     }
14     for (i=1; i<=100; i++) {
15         if (Binario [i]==1){
16             for (j=i; j<=100; j++ ){
17                 printf ("%d", Binario[j]);
18             }
19             break;
20         }
21     }
22     return 0;
23 }
```

Dame el numero: 20  
10100

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

4.3 Escriba un programa usando un ciclo “do while” en el cual se envíe un mensaje de error si el usuario no ingresa un número de 4 cifras con dígitos no repetidos. A la salida imprimirá el mensaje “código no válido”. En caso contrario imprimirá el mensaje “código válido”.

**No logre acabarlo, ocupaban la Pc... yo luego lo  
anexo**

**Gracias**

4.4. La secuencia de números de Fibonacci es la siguiente:  $F_1 = 1$ ,  $F_2 = 1$ ,  $F_3 = 2$ , y en general:

$$F_n = F_{n-1} + F_{n-2}.$$

```
main.c
1  #include<stdio.h>
2  int main() {
3      double f1, f2, f3, numero, i;
4      double divs;
5      f1= 1;
6      f2= 1;
7      f3= 2;
8      printf("Ingresa un numero:");
9      scanf("%lf",&numero);
10     for( i=0; i<numero; i++){
11         f3 = f1+f2;
12         printf("%lf\n",f3);
13         f1= f2;
14         f2= f3;
15     }
16     divs= (f3/f1);
17     printf("La aproximacionn es: %.4lf", divs);
18     return 0;
19 }
```

Ingresa un numero:4  
2.000000  
3.000000  
5.000000  
8.000000  
La aproximacionn es: 1.6000

