



Nombre de la práctica	LENGUAJE C			No.	9,10,11
Asignatura:	MÉTODOS NÚMERICOS	Carrera:	ISIC	Duración de la práctica (Hrs)	

VAZQUEZ MONTIEL MARIAN ELIZABETH

**I. Competencia(s) específica(s):**

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

Casa, escuela.

**III. Material empleado:**

Dev c++

While.pdf

Arreglo.pdf

For.pdf

**IV. Desarrollo de la práctica:**

Permite que un conjunto de sentencias puedan ser ejecutadas repetidamente según el estado de una expresión lógica (condición).

## WHILE

El ciclo while funciona de la siguiente manera, declaramos primero la inicialización de la variable, dentro del ciclo se pone la condición, después el incremento.

El programa funciona de la siguiente manera, declaramos o estándares de entra y salida, la clase principal, un mensaje se imprime hasta que cumpla la condición en este caso hasta que contador sea menor que 3, cuando pasa por el incremento se aumenta 1.

Terminado el ciclo se termina el programa con un fin.

home > mary > Escritorio > c > C while.hola.c > main()

```
1 #include <stdio.h>
2 int main (){
3     int contador = 0;
4     while ( contador < 3 ){
5         printf ("Hola \n");
6         contador++;
7     }
8     printf ("Fin");
9     return 0;
10 }
```

mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/

mary@mary-HP-240-G3-Notebook-PC:~/Escritorio\$ cd c/
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c\$ gcc while\_hola.c -o d
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c\$ ./d
Hola
Hola
Hola
Finmary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c\$



Este programa imprime todos los números del 1 al 1000. Se realiza por medio del while donde la variable se incrementa a uno cada vez que se cumple la condición, así sucesivamente hasta que la condición de como falsa se terminara el ciclo y dejara de imprimir los números. Recordemos que el incremento se realiza en la segunda vuelta no a la primera vez que entra.

```
home > mary > Escritorio > c > C while_num.c > main()
1 #include <stdio.h>
2 int main (){
3     int numero = 1;
4     while (numero<=1000){
5         printf ("%d, ", numero);
6         numero++;
7     }
8     return 0;
9 }
```

OUTPUT TERMINAL DEBUG CONSOLE ... 1: bash + - ×

```
x
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./x
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 3
6, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,
53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 6
9, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
```

Vamos a mostrar el cuadrado y el cubo de 1 al 500. Se crea de la siguiente manera, ponemos la condición donde el numero debe ser menor o igual a 500, dentro del ciclo realizamos la operación donde se multiplica dos veces el número y otra donde se multiplica tres ocasiones, mostramos un mensaje con el número y el resultado de las 2 operaciones, además del incremento.

```
home > mary > Escritorio > c > C White2.c > main()
6     int num=1;
7     int x;
8     int z;
9
10    while (num<=500)
11    {
12        x=num*num;
13        z=num*num*num;
14
15        printf ("%d,%d,%d\n", num,x,z);
16        num++;
17    }
18    system("pause");
19    return 0;
20 }
```

OUTPUT TERMINAL DEBUG CONSOLE ... 1: bash + - ×

```
488, 238144, 116214272
489, 239121, 116930169
490, 240100, 117649000
491, 241081, 118370771
492, 242064, 119095488
493, 243049, 119823157
494, 244036, 120553784
495, 245025, 121287375
496, 246016, 122023936
497, 247009, 122763473
498, 248004, 123503992
499, 249001, 124251499
500, 250000, 125000000
501, 11 pause not found
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```



El ejercicio nos pide que sea divisible entre 2, 7, 3 y que sea mayor a 0.

Se realizó ciclos anidados, donde declaramos la variable de inicialización así como las variables que posteriormente vamos a ocupar, nuestro ciclo de acuerdo a la condición a respetar, dentro de este while se declaró un ciclo for donde ingresamos más condiciones para imprimir en pantalla lo que se requiere.

En el otro ciclo de igual manera tenemos más condiciones que se deben de considerar, para imprimir lo requerido.

Después tenemos el incremento de la variable.

```
home > mary > Escritorio > c > C While3.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int num=1;
5     int x;
6     int z;
7     int num1=1;
8     int x1;
9     while (num<1000){
10         x1=num%2;
11         z=num%7;
12         x=num%3;
13         if (num>0 & x==0){
14             printf("%d----", num);
15         }
16         if (num1>0 & x1==0 & z==0){
17             printf("%d\n", num);
18         }
19         num++;
20     }
21     int n=0;
22     int m;
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
897----900----903----906----909----910
912----915----918----921----924----924
927----930----933----936----938
939----942----945----948----951----952
954----957----960----963----966----966
969----972----975----978----980
981----984----987----990----993----994
[]
```



Este ciclo es un do-while donde esta estructura primero ejecuta el conjunto de instrucciones y después verifica que la condición se cumpla.

El do es se realiza mientras se cumple la condición, además en esta parte igual se realiza el incremento de la variable inicializada, en la segunda entrada al ciclo. Después se encuentra nuestra condición donde el mensaje que está en la parte del do se debe imprimir 3 ocasiones.

```
home > mary > Escritorio > c > C DoWhile.c > ...  
1 #include <stdio.h>  
2 int main(){  
3     int i = 0 ;  
4     do  
5     {  
6         printf("Valor de i = %d\n",i);  
7         i++;  
8     } while (i<3);  
9     printf("fin");  
10    return 0 ;  
11 }  
  
OUTPUT TERMINAL DEBUG CONSOLE ... z:bash + ☰ ^ x  
  
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/$ cd Escritorio/  
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio$ cd c/  
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc DoWhile.c -o d  
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./d  
Valor de i = 0  
Valor de i = 1  
Valor de i = 2  
Finmary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

## ARREGLO

Un arreglo es una variable que hace referencia a varias posiciones de memoria.

Declaramos un arreglo con su tipo de dato, el nombre y si conocemos su tamaño ponerlo dentro de corchetes del caso contrario dejarlo vacío o si conocemos su valor.

Declararemos nuestro arreglo de tamaño 8, después lo ocupamos el arreglo 0 con el valor de 5, el arreglo 1 con valor de 10, arreglo 2 para realizar una suma y el resultado se concatene él. Imprimimos en pantalla el valor de la suma de dos arreglos.

```
home > mary > Escritorio > c > C arreglo1.c > ...
1 #include <stdio.h>
2 int main (){
3     int miArreglo[8];
4     miArreglo[0] = 5;
5     miArreglo[1] = 10;
6     miArreglo[2] = miArreglo[0]+ miArreglo[1];
7     printf("%d",miArreglo[2]);
8     return 0;
9 }
```

OUTPUT TERMINAL ... 2: bash + □ □

```
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc arreglo1.c -o l
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./l
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc arreglo1.c -o l
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./l
15mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ [ ]
```



En el código se declaró un arreglo de tamaño 10, donde por medio de un ciclo for se realiza lo siguiente, se inicializa una variable, la condición que debe de ser menor a 10, el incremento.

Dentro del for al arreglo se le asigna dentro de los corchetes la variable x que se ejecutara con el valor de 10, imprimiendo en pantalla la posición así como el número 10.

En la ejecución del programa nos podemos dar cuenta de que nos muestra la posición y el número 10 en todas.

```
home > mary > Escritorio > c > C_Arrays > ...
1 #include <stdio.h>
2 int main(){
3     int i [10];
4     for (int x = 0; x < 10; x++){
5         i[x] = 10;
6         printf("Posicion %d: %d\n",x,i[x]);
7     }
8     printf("Final");
9     return 0;
10 }
```

```
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./array
Posicion 0: 10
Posicion 1: 10
Posicion 2: 10
Posicion 3: 10
Posicion 4: 10
Posicion 5: 10
Posicion 6: 10
Posicion 7: 10
Posicion 8: 10
Posicion 9: 10
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

En el código se declaró un arreglo de tamaño 10, donde por medio de un ciclo while se realiza lo siguiente, se inicializa una variable, la condición que debe de ser menor a 10, el incremento.

Dentro del while al arreglo se le asigna dentro de los corchetes la variable x que se ejecutara con el valor de 10, imprimiendo en pantalla la posición así como el número 10.

En la ejecución del programa nos podemos dar cuenta de que nos muestra la posición y el número 10 en todas.

```
home > mary > Escritorio > c > C_como_funciona > C > main()
1 #include <stdio.h>
2 int main (){
3     int vector[10];
4     int i = 0;
5     while (i<10){
6         vector[i] = 10;
7         i++;
8     }
9     i = 0;
10    while (i<10){
11        printf ("%d", vector[i]);
12        i++;
13    }
14    return 0;
15 }
```

```
Output TERMINAL ...
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc como_funciona.c -o como_funciona
como_funciona.c: In function 'main':
como_funciona.c:3:19: error: invalid initializer
      int vector[0]=10;
                           ^
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc como_funciona.c -o como_funciona
c:0: comando
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./como_funciona
10101010101010101010mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```



Crear un arreglo de 100 posiciones. Llenar el arreglo con la tabla del 2. Mostrar el arreglo en pantalla.

Se establecen los estándares de entrada y salida, la clase principal, un ciclo for en el cual se inicializa la variable, la condición que debe de ser menor a 100, su incremento. El arreglo se le asigna la variable del ciclo for y esta se multiplica por 2 para obtener la tabla de 2.

Se imprime en pantalla el número así como la posición y el resultado, terminado el ciclo muestra un mensaje de fin.

En la compilación nos mostrará lo siguiente.

```
1 #include <stdio.h>
2 int main(){
3     int i [100];
4     for (int x = 0; x < 100; x++){
5         i[x] = 2*x;
6         printf("2 * %d: %d\n",x,i[x]);
7     }
8     printf("Fin");
9     return 0 ;
10 }
11
```

OUTPUT TERMINAL ... 1: bash + ⌂

```
2 * 80: 160
2 * 81: 162
2 * 82: 164
2 * 83: 166
2 * 84: 168
2 * 85: 170
2 * 86: 172
2 * 87: 174
2 * 88: 176
2 * 89: 178
2 * 90: 180
2 * 91: 182
2 * 92: 184
2 * 93: 186
2 * 94: 188
2 * 95: 190
2 * 96: 192
2 * 97: 194
2 * 98: 196
2 * 99: 198
marv@marv-HP-240-G3-Notebook-PC:~/Escritorio/c$
```



Crear un arreglo de 100 posiciones. Llenar el orden inverso al índice. Mostrar el arreglo en pantalla.

El programa tiene declarado un arreglo de tamaño 100, el ciclo contiene una inicialización de 99, la condición de que debe de ser mayor a cero, un decremento. Dentro del ciclo el arreglo con la posición y el número.

```
home > mary > Escritorio > c > C Arre3.c > main()
1 #include <stdio.h>
2 int main(){
3     int i [100];
4     for (int x = 99; x > 0; x--){}
5     i[x]=x;
6     printf("posición %d %d\n",x,i[x]);
7 }
8 printf("Fin");
9 return 0 ;
posicion100 99
posicion99 98
posicion98 97
posicion97 96
posicion96 95
posicion95 94
posicion94 93
posicion93 92
posicion92 91
posicion91 90
Finmary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

#### Datos

$$A = [3, 5, 6, 8, 4, 7, 8, 5, 3, 1]$$

$$B = [3, 4, 6, 8, 9, 1, 2, 3, 0, 9]$$

Realizar las siguientes operaciones

$$A[3] \bmod(B[2]/2)$$

$$B[A[1]] - A[9]$$

$$A[0] + A[1+2]$$

$$A[5] + B[5]$$

$$(A[3]/B[2])/2$$

```
home > mary > Escritorio > c > C Arre4.c > main()
1 #include <stdio.h>
2 int main(){
3     int i[10]={3,5,6,8,4,7,8,5,3,1};
4     int j[10]={3,4,6,8,9,1,2,3,0,9};
5     int a,b,c,d,e;
6     a=i[3]\bmod(j[2]/2);
7     printf("operación %d \n",a);
8     b=i[1]\bmod(i[9]);
9     printf("operación %d \n",b);
10    c=i[0]+i[1+2];
11    printf("operación %d \n",c);
12    d=i[5]+j[5];
13    printf("operación %d \n",d);
14    e=(i[3]/j[2])/2;
15    printf("operación %d \n",e);
16    printf("Fin");
17    return 0 ;
18 }
```

OUTPUT TERMINAL DEBUG CONSOLE --- Bash + ↻ ⌂ ⌄ ⌁ ⌂ ⌁ ⌁

```
gcc: fatal error: no input files
compilation terminated.
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc Arre4.c -o arr4
arr4
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./arr4
operación 2
operación 8
operación 8
operación 8
operación 8
Finmary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

Ingresamos los arreglos ya declarados con su tipo de dato, variables que vamos ocupar.

Ingresamos las operaciones de acuerdo a lo que nos pide, imprimimos en pantalla el resultado de la operación, con la variable donde se encuentra el resultado.

Esto se realiza para todas las operaciones.



Crea un arreglo de 20 posiciones. Asígnale a cada elemento un valor. Calcula el promedio de todos los elementos. Calcula la multiplicación de todos los elementos.

El programa tiene declarado un arreglo con su tamaño, el ciclo for donde contiene sus características, dentro del ciclo esta para ingresar datos desde teclado con el mensaje que esta, se lee el arreglo, para poder hacer la multiplicación de todos los números ingresados, se realiza por los números ingresados y el resultado se guarda en la variable, lo mismo pasa para la suma de los números ingresados.

Fuera del ciclo se realiza la división de la suma de todos los números entre el total.

En la ejecución del programa nos podemos dar cuenta que desde teclado ingresamos los números hasta llegar a 20, después se cierra el ciclo y realiza lo último.

```
Arre2.c ● Arre3.c ● Arre4.c ● Arre6.c ● ...  
home > mary > Escritorio > c > Arre6.c > main()  
1 #include <stdio.h>  
2 int main(){  
3     int x[20];  
4     int a=1,b=0;  
5     for (int i = 0; i < 20; i++){  
6         printf("ingresa la %d posicion", (i+1));  
7         scanf("%d",&x[i]);  
8         a=a*x[i];  
9         b=x[i]+b;  
10    }  
11    a=a/20;  
12    printf("multiplicacion %d",b);  
13    printf("promedio %d",a);  
14    printf("Fin");  
15    return 0 ;  
16 }  
OUTPUT TERMINAL ... 1: bash + ^ x  
ingresa la 7 posicion8  
ingresa la 8 posicion9  
ingresa la 9 posicion11  
ingresa la 10 posicion12  
ingresa la 11 posicion13  
ingresa la 12 posicion14  
ingresa la 13 posicion1  
ingresa la 14 posicion2  
ingresa la 15 posicion3  
ingresa la 16 posicion4  
ingresa la 17 posicion16  
ingresa la 18 posicion7  
ingresa la 19 posicion8  
ingresa la 20 posicion9  
multiplicacion 144promedio 14942208Finmary@mary-HP-240-G3-Notebook  
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```



## CICLO FOR

En el código se declaró un arreglo de tamaño 10, donde por medio de un ciclo for se realiza lo siguiente, se inicializa una variable, la condición que debe de ser menor a 10, el incremento.

Dentro del for al arreglo se le asigna dentro de los corchetes la variable i que se ejecutara con el valor de 10.

En la ejecución del programa nos podemos dar cuenta de que nos muestra el arreglo.

```
home > mary > Escritorio > c > C cicloFor.c > main()
1 #include <stdio.h>
2 int main (){
3     int vector [10];
4     int i = 0;
5     for (i=0; i<10; i++){
6         vector[i] = 10;
7     }
8     printf("%d",vector[i]);
9 }
```

  

```
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc cicloFor.c -o
ciclo
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./ciclo
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc cicloFor.c -o
ciclo
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./ciclo
ector -1942027776mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

Ln 8, Col 22 Spaces:4 UTF-8 LF C Linux ⌂ ⌂

En este ejemplo una variable tiene le valor de 50 donde el ciclo for, en la condicon primero debe de hacer una operación en la cual cada numero del 1 al 50 debe ser dividio por 2, despues de esto cada numero se multiplica por 3, 2 se imprime en pantalla, hasta que sea el, numero 50.

Lo podemos ver en la ejecución del programa.

```
home > mary > Escritorio > c > C serie.c > main()
1 #include <stdio.h>
2 int main (){
3     int longitudSerie = 50;
4     int i;
5     for (i = 1; i<=(longitudSerie/2); i++){
6         printf("%d, ", 2*i);
7         printf("%d, ", 3*i);
8     }
9 }
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio$ cd c/
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc serie.c -o ser
ie
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./serie
2, 3, 4, 6, 6, 9, 8, 12, 10, 15, 12, 18, 14, 21, 16, 24, 18, 27, 2
0, 30, 22, 33, 24, 36, 26, 39, 28, 42, 30, 45, 32, 48, 34, 51, 36,
54, 38, 57, 40, 60, 42, 63, 44, 66, 46, 69, 48, 72, 50, 75, mary@
mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

Ln 6, Col 9 Spaces:4 UTF-8 LF C Linux ⌂ ⌂



Escribe un programa que reciba un número N del usuario y haga la suma de todos los números desde 1 hasta N. Ej. >> 5       $1+2+3+4+5 = 15$

Para esto mostramos un mensaje que ingrese un número, después se lee la variable, o sea se escanea para que esta entre en un for donde se ejecutara hasta que sea menor a la variable ingresada, y se concatena el valor que se va sumando cada número, terminado el ciclo se imprime en pantalla el valor de la suma.

Nos pide que ingresemos un número y después nos muestra la suma como en el ejemplo.

```
home > mary > Escritorio > c > C For1.c > main()
1 #include <stdio.h>
2 int main(){
3     int a,b=0;
4     puts("ingresa un numero N");
5     scanf("%d");
6     for (int i=1; i<=b; i++){
7         b+=i;
8     }
9     printf("\nla suma es %d\n",b);
10    return 0;
11 }
```

```
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ gcc For1.c -o for1
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./for1
ingresa un numero N
5
la suma es 15Fin
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

```
home > mary > Escritorio > c > C For2.c > main()
1 #include <stdio.h>
2 int main(){
3     puts("A\b\tA+2\c\tA+4\c\tA+6\b\n");
4     int c=3,d=6,e=9,f=12;
5
6     for (int i = 0; i <=6; i+=2){
7         c=c+i;
8         d=d+i;
9         f=f+i;
10        e=e+i;
11        printf("%d\t%d\t%d\t%d\b\n",c,d,e,f);
12    }
13    printf("Fin");
14    return 0;
15 }
```

```
mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$ ./for2
A      A+2      A+4      A+6
3      6       9       12
5      8       11      14
9      12      15      18
15     18      21      24
Finmary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c$
```

A	A+2	A+4	A+6
3	5	7	9
6	8	10	12
9	11	13	15
12	14	16	18

En este programa tenemos que mostrar lo siguiente para ello puts solo sirve solo para mostrar mensajes, después a variables le declaramos su valor, en el ciclo for la condición debe de ser menor a 6, cada variable debe de hacer diferente operación, mostramos en pantalla el resultado.

Terminado el ciclo mostramos un menaje de fin.



El for anidado es un for dentro de otro for. Por ejemplo; el código declara dos variables el primer for la condición debe de ser menor a 5 y dentro de él se imprime del 1 al 5, después el otro for su condición debe de ser menor a 3, donde en pantalla se debe de mostrar la i tres ocasiones hasta que se cumpla la condición del primer for.

El resultado lo podemos ver en la ejecución y de esta manera te puedes dar cuenta mejor de como funciona el ciclo anidado.

```
home > mary > Escritorio > t > C for_anidado.c > ...
 1 #include <stdio.h>
 2 int main(){
 3     int i;
 4     int j;
 5     for(i=0; i<5; i++){
 6         printf("para id: %d \t", i);
 7         for(j=0; j<3; j++){
 8             printf("%d", j);
 9         }
10         printf("\n");
11     }
12     return 0;
13 }
```

mary@mary-HP-240-G3-Notebook-PC:~/Escritorio/c\$ ./anidado

para id: 0 j0,j1,j2,

para id: 1 j0,j1,j2,

para id: 2 j0,j1,j2,

para id: 3 j0,j1,j2,

para id: 4 j0,j1,j2,

En este código declaramos arreglos de tamaño 10 hasta 10 ocasiones porque debemos realizar la multiplicación del número que el usuario ingrese, ocupamos for anidados donde primero los arreglos e multiplican por el primer número ingresado y en el segundo for por el segundo numero ingresado.

La condición de ambos debe ser menor al número que el usuario ingreso.

Para imprimir los resultados deben de ser todos los arreglos declarados.

### V. Conclusiones:

Los ciclos nos sirven de mucha ayuda cuando queremos realizar problemas de acuerdo a un número establecido por el usuario o por el programador para este ponerlo en la condición y se realice las veces que sean necesarias implementando código ya sea para resolver operaciones y solo mostrar cosas.

Algunos problemas si tuve dificultad para hacerlos ya que debía de analizarlo primero para encontrar la solución que necesito parar llegar con el resultado.