



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Karina García Morales

*Asignatura:* Fundamentos de Programación

*Grupo:* 20

*No. de práctica(s):* No. 8 - Estructuras de repetición.

*Integrante(s):* López Olmedo Ernesto Yael

*No. de lista o brigada:* 27

*Semestre:* 2024-1

*Fecha de entrega:* 17 de Octubre 2023

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

## **Introducción**

### Estructuras de repetición

Como parte del aprendizaje del lenguaje C, los alumnos ya conocen parte de las bases para la creación de programas con procesos sencillos como concatenación de funciones, variables con las cuales tendrán funciones múltiples dentro del programa, por lo que el alumno aún requiere conocer más operadores para facilitar su trabajo en la forma de redacción de sus algoritmos, también con la finalidad de producir estructuras repetitivas con las cuales permitirán realizar múltiples iteraciones en vez de usar estructuras simples con la misma funcionalidad repetidas. Todo esto a través de la experimentación de las estructuras for, while y do-while, que anteriormente el alumno conoció en diagramas de flujo, ahora con la misma lógica, este va a comprobar su escritura(sintaxis) y su función en terminal.

## **Objetivo**

El alumno elaborará programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición.

## Desarrollo

Ahora se empezará por describir la estructura y la forma de operación de cada una de las estructuras de repetición.

-Estructura de control de repetición for:

La estructura for (para), tiene la función de repetir a través de un control ya preestablecido, que se divide en: valor de iniciación, expresión lógica(condición), y la operación de iteración(si el proceso se incrementa o se decrementa). Con esta simple sentencia, permite que el contenido dentro de esta estructura se ejecuta, mientras la condición resulte verdadera. Su estructura es la siguiente forma:

```
for(valor inicial ; expresión lógica(condición) ; iteraciones (Incremento o decremento)
{
    Instrucciones a realizar.
}
```

-Estructura de control repetitiva while:

Esta estructura de control while (hacer), primero comprueba la veracidad de la función lógica, como el caso del if, pero en esta al verificar que es verdadera, ejecuta todas las acciones que estén dentro de la estructura, cuando este bloque sea finalizado vuelve a pasar por el while, y se ejecutará en caso de ser correcta de forma cíclica hasta no serlo. Su estructura es de la siguiente forma:

```
while ( expresión lógica){
    Bloque de instrucciones;
    i++ //Opcional: se puede usar este operador para permitir que sea verdadero
    y repita el proceso.
}
```

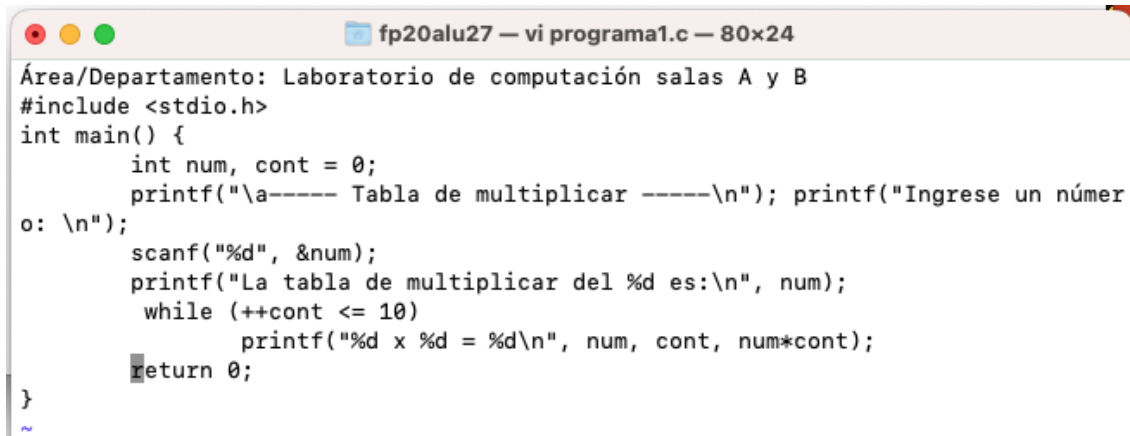
-Estructura de control de repetición do-while:

Resulta como complemento del operador while (mientras), ya que este señala de forma más precisa que es lo que se estará ejecutando a través del do (hacer) de forma inicial, para posterior ejecutar la condición y comprobar si se realiza la reiteración, creando el do-while (hacer mientras). Su estructura es de la siguiente forma:

```
do {
    instrucciones que se ejecuta una vez hasta pasar a while
} while (Expresión lógica);
```

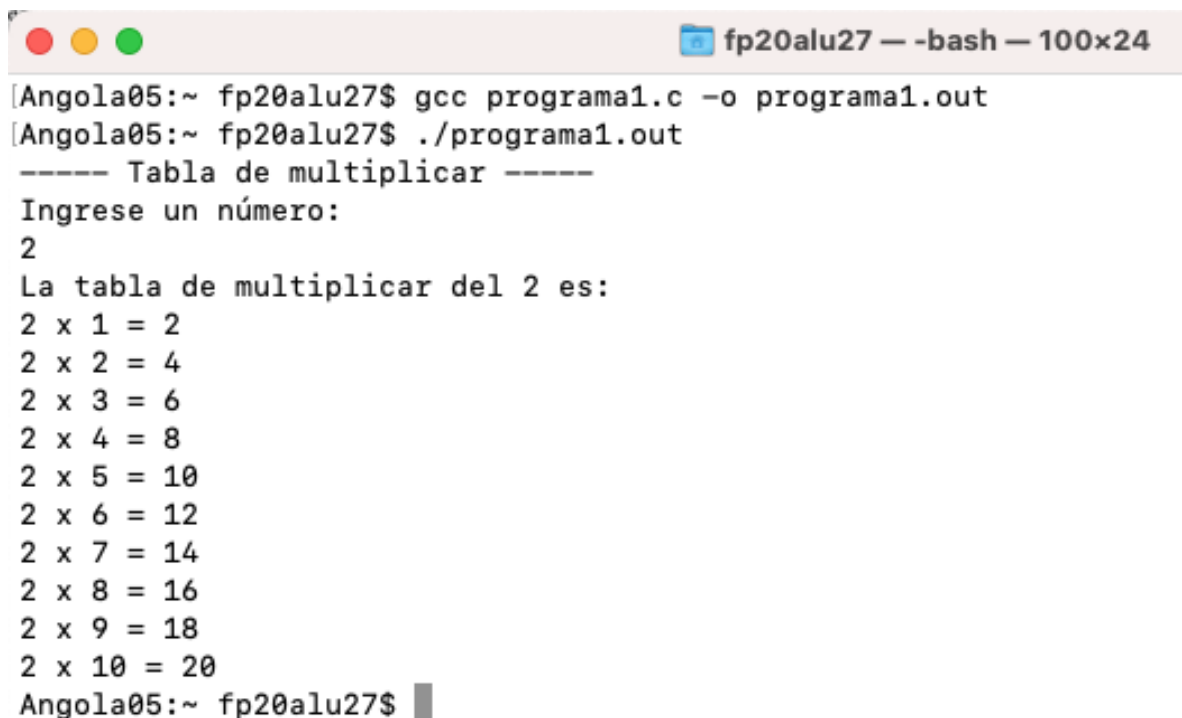
## Laboratorio

### Programa 1.c



```
fp20alu27 — vi programa1.c — 80x24
Área/Departamento: Laboratorio de computación salas A y B
#include <stdio.h>
int main() {
    int num, cont = 0;
    printf("\a----- Tabla de multiplicar -----\\n"); printf("Ingrese un número: \\n");
    scanf("%d", &num);
    printf("La tabla de multiplicar del %d es:\\n", num);
    while (++cont <= 10)
        printf("%d x %d = %d\\n", num, cont, num*cont);
    return 0;
}
```

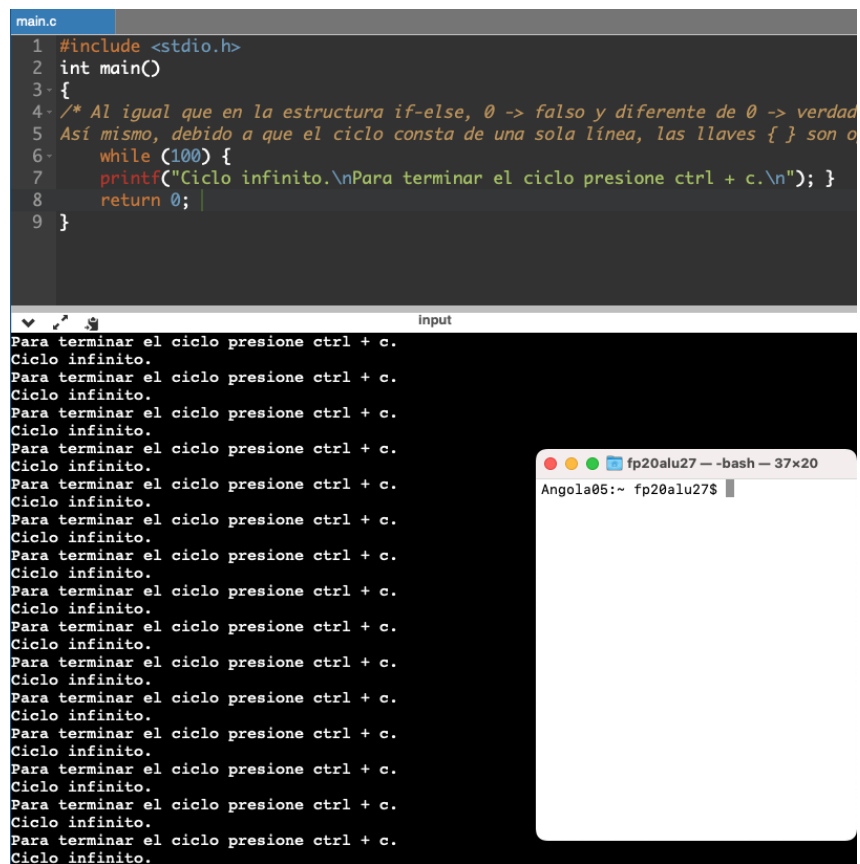
Figura 1.1 - Programa 1.c



```
fp20alu27 — -bash — 100x24
[Angola05:~ fp20alu27$ gcc programa1.c -o programa1.out
[Angola05:~ fp20alu27$ ./programa1.out
----- Tabla de multiplicar -----
Ingrese un número:
2
La tabla de multiplicar del 2 es:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
Angola05:~ fp20alu27$
```

Figura 1.2 - Salida 1.c

## Programa 2.c



The screenshot shows a C program named 'main.c' with the following code:

```
1 #include <stdio.h>
2 int main()
3 {
4     /* Al igual que en la estructura if-else, 0 -> falso y diferente de 0 -> verdad
5     Así mismo, debido a que el ciclo consta de una sola línea, las llaves {} son o
6     while (100) {
7         printf("Ciclo infinito.\nPara terminar el ciclo presione ctrl + c.\n"); }
8     return 0;
9 }
```

The program is running in a terminal window. The output shows a repeating pattern of the message "Ciclo infinito. Para terminar el ciclo presione ctrl + c." followed by a newline. The terminal window title is "fp20alu27 - bash - 37x20".

Figura 1.3 - Programa 2.c error de bucle.



The screenshot shows the same C program, but with the loop condition corrected from 'while (100)' to 'while (1)'. The code is as follows:

```
1 #include <stdio.h>
2 int main() {
3     /* Al igual que en la estructura if-else, 0 -> falso y diferente de 0 -> verdad
4     Así mismo, debido a que el ciclo consta de una sola línea, las llaves {} son o
5     while (1) {
6         printf("Ciclo infinito.\nPara terminar el ciclo presione ctrl + c.\n");
7         break;
8     }
9     return 0;
10 }
```

The program is running in a terminal window. The output shows the message "Ciclo infinito. Para terminar el ciclo presione ctrl + c." followed by a newline. The terminal window title is "fp20alu27 - bash - 37x20". The program has finished with exit code 0.

Figura 1.4 - Programa 2.c corrección.

## Programa 3.c

```
1  #include <stdio.h>
2  int main ()
3  {
4      char op = 'n';
5      double sum = 0, calif = 0; int veces = 0;
6      do
7      {
8          printf("\tSuma de calificaciones\n");
9          printf("Ingrese la calificación:\n");
10         scanf("%lf", &calif);
11         veces++;
12         sum = sum + calif;
13         printf("¿Desea sumar otra? S/N\n");
14         setbuf(stdin, NULL); // limpia el buffer del teclado scanf("%c",&op);
15         getchar();
16     }
17
18     while (op == 'S' || op == 's');
19     printf("El promedio de las calificaciones ingresadas es: %lf\n", sum/veces);
20     return 0;
21 }
```

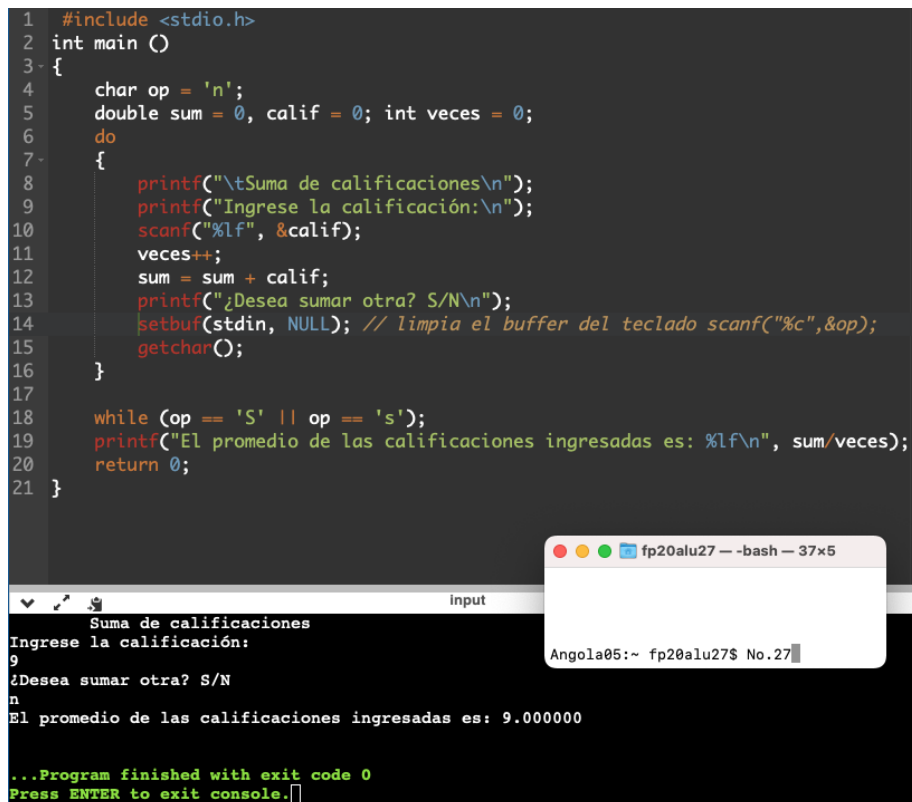


Figura 1.5 - Programa 3.c uso de setbuf

```
1  #include <stdio.h>
2  int main ()
3  {
4      char op = 'n';
5      double sum = 0, calif = 0; int veces = 0;
6      do
7      {
8          printf("\tSuma de calificaciones\n");
9          printf("Ingrese la calificación:\n");
10         scanf("%lf", &calif);
11         veces++;
12         sum = sum + calif;
13         printf("¿Desea sumar otra? S/N\n");
14         //setbuf(stdin, NULL); // limpia el buffer del teclado scanf("%c",&op);
15         getchar();
16     }
17
18     while (op == 'S' || op == 's');
19     printf("El promedio de las calificaciones ingresadas es: %lf\n", sum/veces);
20     return 0;
21 }
```

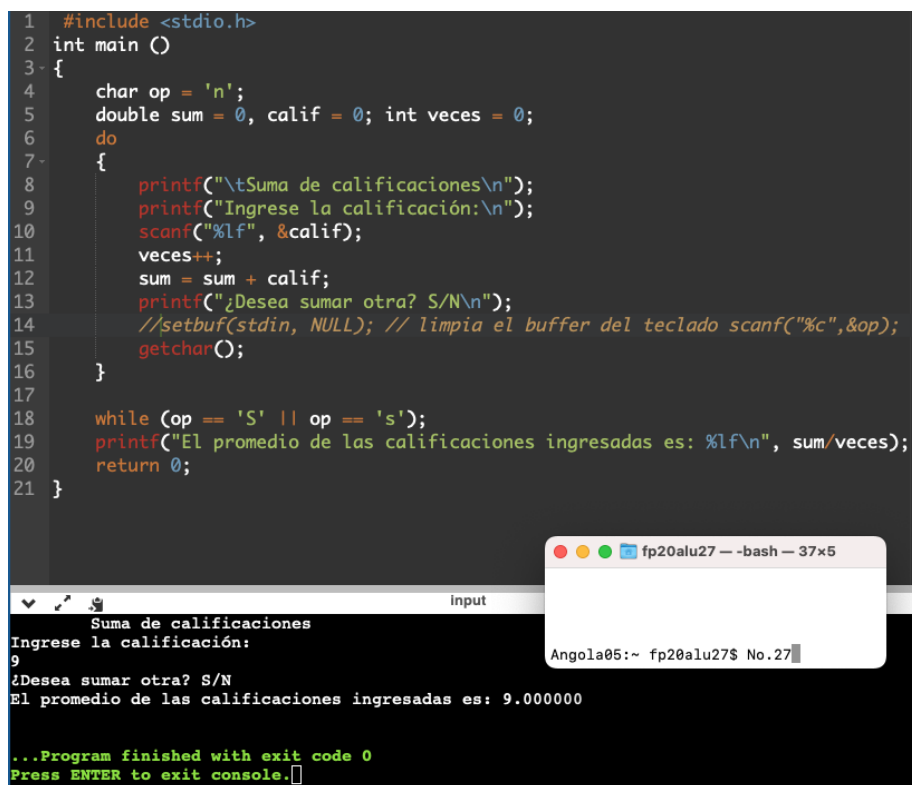


Figura 1.6 - Programa 3.c sin setbuf

## Programa 4.c

```
1 #include <stdio.h>
2 int main () {
3     int op, uno, dos;
4     do{
5         printf(" --- Calculadora ---\n"); printf("\n¿Qué desea hacer\n");
6         printf("1) Sumar\n");
7         printf("2) Restar\n");
8         printf("3) Multiplicar\n"); printf("4) Dividir\n"); printf("5) Salir\n");
9         scanf("%d",&op);
10        switch(op) {
11            case 1:
12                printf("\tSumar\n");
13                printf("Introduzca los números a sumar separados por comas\n");
14                scanf("%d, %d",&uno, &dos);
15                printf("%d + %d = %d\n", uno, dos, (uno + dos));
16                break;
17            case 2:
18                printf("\tRestar\n");
19                printf("Introduzca los números a restar separados por comas\n");
20                scanf("%d, %d",&uno, &dos);
21                printf("%d - %d = %d\n", uno, dos, (uno - dos));
22                break;
23            case 3:
24                printf("\tMultiplicar\n");
25                printf("Introduzca los números a multiplicar separados por comas\n");
26                scanf("%d, %d",&uno, &dos);
27                printf("%d x %d = %d\n", uno, dos, (uno * dos));
28                break;
29            case 4:
30                printf("\tDividir\n");
31                printf("Introduzca los números a dividir separados por comas\n");
32                scanf("%d, %d",&uno, &dos);
33                printf("%d / %d = %d\n", uno, dos, (uno / dos));
34                break;
35            case 5:
36                printf("¡Gracias!\n");
37                return 0;
38            default:
39                printf("Opción no válida\n");
40                break;
41        }
42    } while (op != 5);
43 }
```




Figura 1.7 - Programa 4.c

## Corrección do-while programa 1.c

### -Opción verdadera

```
1 #include <stdio.h>
2 int main() {
3     int num, cont = 0, op;
4     do
5     {
6
7         printf("\n----- Tabla de multiplicar ----- \n");
8         printf("Ingrese un número: \n");
9         scanf("%d", &num);
10        printf("La tabla de multiplicar del %d es:\n", num);
11        while (++cont <= 10)
12            printf("%d x %d = %d\n", num, cont, num*cont);
13        printf("\n---Desea calcular otra multiplicacion 1 or 0: \n");
14        scanf("%d",&op);
15    }
16    while (op ==1 || op==1);
17    return 0;
18 }
```

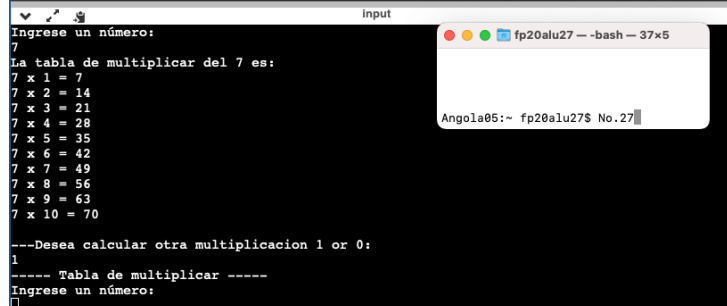


Figura 1.8 - Programa 1.c do-while verdadero.

-Opción falsa

```
1 #include <stdio.h>
2 int main() {
3     int num, cont = 0, op;
4     do
5
6     {
7         printf("\a----- Tabla de multiplicar ----- \n");
8         printf("Ingrese un número: \n");
9         scanf("%d", &num);
10        printf("La tabla de multiplicar del %d es:\n", num);
11        while (++cont <= 10)
12            printf("%d x %d = %d\n", num, cont, num*cont);
13        printf("\n---Desea calcular otra multiplicacion 1 or 0: \n");
14        scanf("%d",&op);
15    }
16    while (op ==1);
17    return 0;
18 }
```

input

```
----- Tabla de multiplicar -----
Ingrese un número:
9
La tabla de multiplicar del 9 es:
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

---Desea calcular otra multiplicacion 1 or 0:
0

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




Figura 1.9 - Programa 1.c do-while falso.



## Programa 5.C Define

```
1 #include <stdio.h>
2 #define MAX 5
3 int main ()
4 {
5     int arreglo[MAX], cont;
6     for (cont=0; cont<MAX; cont++)
7     {
8         printf("Ingrese el valor %d del arreglo: ", cont+1);
9         scanf("%i", &arreglo[cont]);
10    }
11    printf("El valor ingresado para cada elemento del arreglo es:\n");
12    for (cont=0; cont<MAX; cont++)
13    {
14        printf("%d\t", arreglo[cont]);
15    }
16    printf("\n");
17    return 0;
18 }
```

Angola05:~ fp20alu27\$ No.27

Input

Ingrese el valor 1 del arreglo: 7  
Ingrese el valor 2 del arreglo: 6  
Ingrese el valor 3 del arreglo: 8  
Ingrese el valor 4 del arreglo: 9  
Ingrese el valor 5 del arreglo: 4  
El valor ingresado para cada elemento del arreglo es:  
[7 6 8 9 4 ]  
...Program finished with exit code 0  
Press ENTER to exit console.

Figura 1.10 - Programa 5.c define.

## Programa 5.c Cambiar 1 while y 2 do-while

```
1 #include <stdio.h>
2 #define MAX 5
3 int main ()
4 {
5     int arreglo[MAX], cont=0;
6     //for (cont=0; cont<MAX; cont++)
7     while(cont<MAX)
8     {
9         printf("Ingrese el valor %d del arreglo: ", cont+1);
10        scanf("%i", &arreglo[cont]);
11        cont++;
12    }
13    printf("El valor ingresado para cada elemento del arreglo es:\n");
14    //for (cont=0; cont<MAX; cont++)
15    do
16    {
17        printf("%d\t", arreglo[cont]);
18        cont++;
19    }while(cont<MAX);
20    printf("\n");
21    return 0;
22 }
```

Angola05:~ fp20alu27\$ No.27

Input

Ingrese el valor 1 del arreglo: 0  
Ingrese el valor 2 del arreglo: 0  
Ingrese el valor 3 del arreglo: 0  
Ingrese el valor 4 del arreglo: 0  
Ingrese el valor 5 del arreglo: 0  
El valor ingresado para cada elemento del arreglo es:  
[0 ]  
...Program finished with exit code 0  
Press ENTER to exit console.

Figura 1.11 - Programa 5.c sustitución por while y do-while.

## Tarea

1- Codifica el ejercicio realizado en el desarrollo de la práctica 5 pseudocódigo.

-Diagrama de flujo:

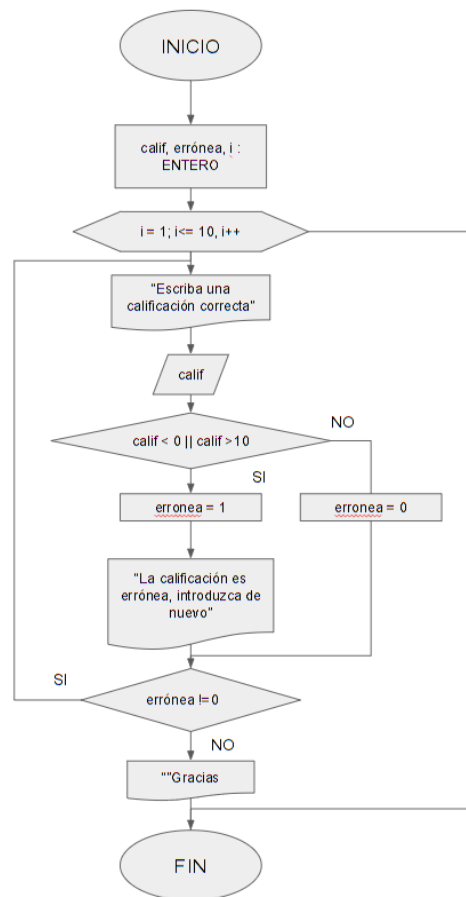


Figura 1.12 - Diagrama de flujo.

### -Pseudocódigo:

INICIO

Calif, errónea, i : ENTERO

PARA(i=1 ; i<=10 ; i++)

HACER

IMPRIMIR "Escriba una calificación correcta"

LEER Calif

SI Calif < 0 O Calif >10

errónea := 1

IMPRIMIR "La calificación es errónea introduzca de nuevo"

SI, NO

errónea := 0;

FIN SI

MIENTRAS errónea <> 0

ESCRIBIR "Gracias"

FIN PARA

FIN

### -Código C:

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int calif, error, i=0;
```

```
do{
```

```
printf("Escriba una calificación correcta: \n");
```

```
scanf("%d",&calif);
```

```
if (calif<0 || calif>10){
```

```
error = 1;
```

```
}else{
```

```
error = 0;
```

```
}
```

```
}while(error!=0);{
```

```
printf("Gracias");
```

```
i++;
```

```
}
```

```
return 0;
```

```
}
```

-Compilador:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int calif, error, i=0;
6
7      do{
8          printf("Escriba una calificación correcta: \n");
9          scanf("%d",&calif);
10
11          if (calif<0 || calif>10){
12              error = 1;
13          }else{
14              error = 0;
15          }
16
17      }while(error!=0);{
18          printf("Gracias");
19          i++;
20      }
21      return 0;
22  }
23
```

Figura 1.13 - Compilador en línea.

-Salida terminal:

1. Caso dentro del parámetro:

```
Escriba una calificación correcta:
6
Gracias

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

Figura 1.14 - Salida terminal verdadero.

2. Caso valor crítico:

```
Escriba una calificación correcta:
-15
Escriba una calificación correcta:
13
Escriba una calificación correcta:
9
Gracias

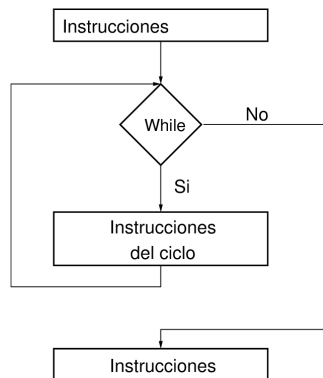
...Program finished with exit code 0
Press ENTER to exit console.
```

Figura 1.4 - Salida terminal falso-verdadero.

2 - Completa el cuadro con las estructuras iterativas (MIENTRAS, HACER MIENTRAS Y PARA).

- MIENTRAS (while).

-Diagrama-



-Pseudocódigo-

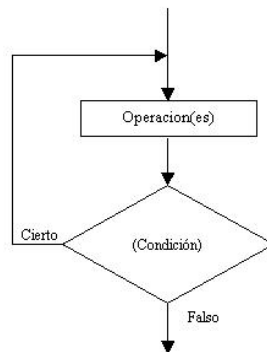
```
MIENTRAS
    bloque de instrucciones
FIN MIENTRAS
```

-Codificación-

```
while ( expresión lógica){
    Bloque de instrucciones;
    i++ //Opcional: se puede usar este operador para permitir que sea verdadero
    y repita el proceso.
}
```

- HACER MIENTRAS (do-while).

-Diagrama-



### -Pseudocódigo-

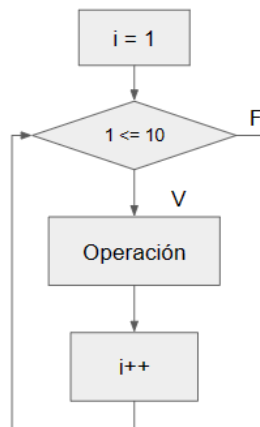
```
HACER
    Bloque de instrucciones
MIENTRAS
    Bloque de instrucciones
FIN MIENTRAS
```

### -Codificación-

```
do {
    instrucciones que se ejecuta una vez hasta pasar a while
} while (Expresión lógica);
```

- PARA (for).

### -Diagrama-



### -Pseudocódigo-

```
PARA (operación lógica) HASTA CON PASO 1 HACER
    Bloque de instrucciones
FIN PARA
```

### -Codificación-

```
for(valor inicial ; expresión lógica(condición) ; iteraciones (Incremento o decremento)
{
    Instrucciones a realizar.
}
```

## **Conclusión**

Con las estructuras de repetición for, while y do-while, ahora el alumno observa de forma práctica y lógica a través de la terminal como es la lectura del código por parte del compilador, junto a la lógica con la que lo analiza, por lo que ahora el alumno ya podrá implementar estas estructura a sus programas y con ello le permitirán crear programas eficaces y eficientes en líneas de código. Las estructuras se sintetizan de la siguiente forma: la estructura for permite en un sola línea de código definir la condición lógica, valor inicial con el cual se operarán las iteraciones y el decremento/incremento de la variable que entrará a la condición; la estructura while se ejecuta en casos verdaderos, con los cuales permite ser usado como una condición y permite repetir el caso de forma interna, mientras este sea verdadero; por último, la estructura do-while permite ejecutar un ciclo antes de la estructura while complementaria, con lo que permite hacer-romper el ciclo de forma sencilla y sin tener que repetir secciones del código.

## Referencias

- Templos, A.(2022).Manual de prácticas del laboratorio de Fundamentos de programación. México: UNAM.