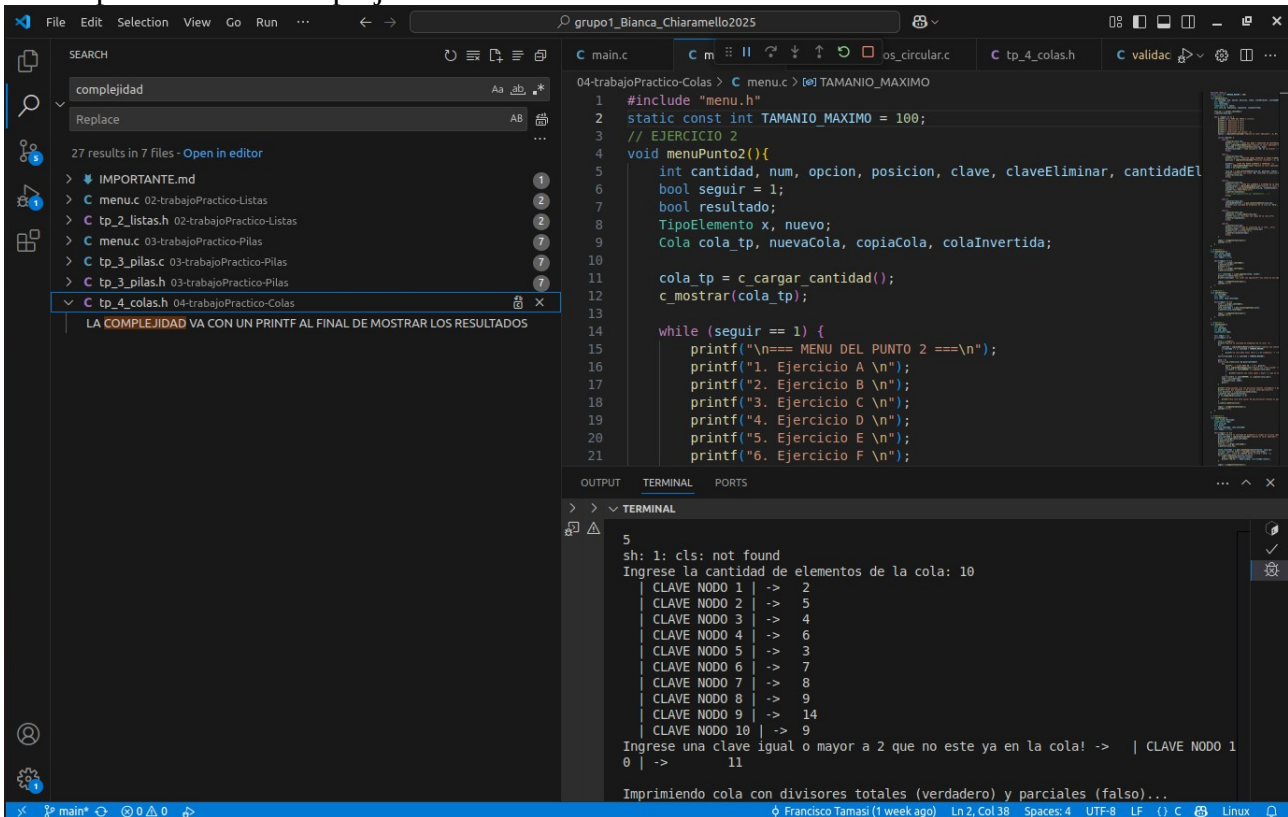


GRUPO 1 – Correcciones Trabajo Práctico: COLAS

RESULTADO DE LA CORRECCIÓN: **APROBADO-**

OBSERVACIONES

No responde sobre complejidad.



The screenshot shows a code editor with a search bar at the top left containing the text 'complejidad'. Below the search bar, a list of 27 results is shown, with the first result being 'LA COMPLEJIDAD VA CON UN PRINTF AL FINAL DE MOSTRAR LOS RESULTADOS'. The main editor area displays the code for 'menu.c', which includes a menu function. The terminal output shows the program's execution, including a prompt for the number of elements in the queue (10) and a list of nodes with their keys and values. The program also prompts for a key to be added or removed, and finally prints the queue with divisors.

```
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadEl
6     bool seguir = 1;
7     bool resultado;
8     TipoElemento x, nuevo;
9     Cola cola_tp, nuevaCola, copiaCola, colaInvertida;
10
11     cola_tp = c_cargar.cantidad();
12     c_mostrar(cola_tp);
13
14     while (seguir == 1) {
15         printf("\n== MENU DEL PUNTO 2 ==\n");
16         printf("1. Ejercicio A \n");
17         printf("2. Ejercicio B \n");
18         printf("3. Ejercicio C \n");
19         printf("4. Ejercicio D \n");
20         printf("5. Ejercicio E \n");
21         printf("6. Ejercicio F \n");
```

```
5
sh: 1: cls: not found
Ingrese la cantidad de elementos de la cola: 10
CLAVE NODO 1 -> 2
CLAVE NODO 2 -> 5
CLAVE NODO 3 -> 4
CLAVE NODO 4 -> 6
CLAVE NODO 5 -> 3
CLAVE NODO 6 -> 7
CLAVE NODO 7 -> 8
CLAVE NODO 8 -> 9
CLAVE NODO 9 -> 14
CLAVE NODO 10 -> 9
Ingrese una clave igual o mayor a 2 que no este ya en la cola! -> CLAVE NODO 1
0 -> 11
Imprimiendo cola con divisores totales (verdadero) y parciales (falso)...
```

Ejercicio 5: inconsistencias entre los tamaños de la Cola que no son considerados, solo se controla el seteado en menu.c. En la implementación con cola circular, permite cargar los dos elementos configurados en menu.c. No muestra la cola original.

GRUPO 1 – Correcciones Trabajo Práctico: COLAS

VS Code interface showing the initial state of the project. The Explorer panel on the left shows the file structure. The main editor shows three files: menu.c, colas_arreglos_circular.c, and validaciones.c. In menu.c, the TAMANIO_MAXIMO is set to 100. In validaciones.c, it is set to 5. The terminal at the bottom shows the command 'sh: 1: cls: not found' and a list of nodes.

```
04-trabajoPractico-Colas > C menu.c > TAMANIO_MAXIMO
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
```

```
04-trabajoPractico-Colas > C menu.c > TAMANIO_MAXIMO
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
```

```
libs > validaciones > implementaciones > C validaciones.c > TAMANIO_MAXIMO
1 #include "../headers/validaciones.h"
2 static const int TAMANIO_MAXIMO = 5;
3 bool esDigito(const char *c){
4     if( strlen(c)>1 && (c[0] == '-' ) ) {
5         c++;
```

OUTPUT TERMINAL PORTS

sh: 1: cls: not found

Ingrese la cantidad de elementos de la cola: 10

CLAVE NODO	CLAVE NODO	CLAVE NODO
1	2	2
2	5	5
3	4	4
4	6	6
5	3	3
6	7	7

VS Code interface showing the project after modifications. The Explorer panel on the left shows the file structure. The main editor shows three files: menu.c, colas_arreglos_circular.c, and validaciones.c. In menu.c, the TAMANIO_MAXIMO is set to 100. In validaciones.c, it is set to 5. The terminal at the bottom shows the command 'sh: 1: cls: not found' and a list of nodes.

```
04-trabajoPractico-Colas > C menu.c > TAMANIO_MAXIMO
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
```

```
04-trabajoPractico-Colas > C menu.c > TAMANIO_MAXIMO
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
```

```
libs > validaciones > implementaciones > C validaciones.c > TAMANIO_MAXIMO
1 #include "../headers/validaciones.h"
2 static const int TAMANIO_MAXIMO = 5;
3 bool esDigito(const char *c){
4     if( strlen(c)>1 && (c[0] == '-' ) ) {
5         c++;
```

OUTPUT TERMINAL PORTS

sh: 1: cls: not found

Ingrese la cantidad de elementos de la cola: 10

CLAVE NODO	CLAVE NODO	CLAVE NODO
1	2	2
2	5	5
3	4	4
4	6	6
5	3	3
6	7	7
7	8	8
8	9	9
9	14	14
10	9	9

Ingrese una clave igual o mayor a 2 que no este ya en la cola! -> | CLAVE NODO 10 | -> 11

GRUPO 1 – Correcciones Trabajo Práctico: COLAS

```
04-trabajoPractico-Colas > C menu.c > [TAMANIO_MAXIMO]
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;

C menu.c x
04-trabajoPractico-Colas > C menu.c > [TAMANIO_MAXIMO]
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 100;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;

C validaciones.c M x
libs > validaciones > implementaciones > C validaciones.c > [TAMANIO_MAXIMO]
1 #include "../headers/validaciones.h"
2 static const int TAMANIO_MAXIMO = 5;
3 bool esDigito(const char *c){
4     if( strlen(c)>1 && (c[0] == '-') ) {
5         c++;

OUTPUT TERMINAL PORTS
sh: 1: cls: not found
Ingrese la cantidad de elementos de la cola: 10
| CLAVE NODO 1 | -> 2
| CLAVE NODO 2 | -> 5
| CLAVE NODO 3 | -> 4
| CLAVE NODO 4 | -> 6
| CLAVE NODO 5 | -> 3
| CLAVE NODO 6 | -> 7
| CLAVE NODO 7 | -> 8
| CLAVE NODO 8 | -> 9
| CLAVE NODO 9 | -> 14
| CLAVE NODO 10 | -> 9
Ingrese una clave igual o mayor a 2 que no este ya en la cola! -> | CLAVE NODO 10 | -> 11
Imprimiendo cola con divisores totales (verdadero) y parciales (falso)...
Valor del elemento // Es divisor total/parcial
Clave: 2 Valor: Falso
```

```
04-trabajoPractico-Colas > C menu.c > [TAMANIO_MAXIMO]
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 2;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
6     bool seguir = 1;
7     bool resultado;
8     TipoElemento x, nuevo;
9     Cola cola tp, nuevaCola, copiaCola, colaInvertida;

C colas_arreglos_circular.c M x
libs > colas > implementaciones > C colas_arreglos_circular.c > ...
1 #include "../headers/colas.h"
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 static const int TAMANIO_MAXIMO = 2; //Siempre 1 mas porque se pierde uno por la ranura
7
8 struct ColaRep {
9     TipoElemento *valores;

OUTPUT TERMINAL PORTS
CAMBIAR EL VALOR DE 'TAMANIO_MAXIMO' EN MENU.C, VALIDACIONES.C Y EN LA IMPLEMENTACION DE COLAS===
=== la variable se encuentra en la primer linea de codigo.===
sh: 1: cls: not found
Ingrese la cantidad de elementos de la cola: 2
| CLAVE NODO 1 | -> 2
Imprimiendo cola con divisores totales (verdadero) y parciales (falso)...
Valor del elemento // Es divisor total/parcial
Clave: 2 Valor: Verdadero
Ingresa 1 para SEGUIR, 0 para NO SEGUIR:
```


GRUPO 1 – Correcciones Trabajo Práctico: COLAS

```
1 #include "menu.h"
2 static const int TAMANIO_MAXIMO = 5;
3 // EJERCICIO 2
4 void menuPunto2(){
5     int cantidad, num, opcion, posicion, clave, claveEliminar, cantidadElementos;
6     bool seguir = 1;
7     bool resultado;
8     TipoElemento x, nuevo;
9     Cola cola_tp, nuevaCola, copiaCola, colaInvertida;
```

```
1 #include "../headers/colas.h"
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 static const int TAMANIO_MAXIMO = 2; //Siempre 1 mas porque se pierde uno por la ranura
7
8 struct ColaRep {
9     TipoElemento *valores;
```

OUTPUT TERMINAL PORTS

cpplib: main

CAMBIAR EL VALOR DE 'TAMANIO_MAXIMO' EN MENU.C, VALIDACIONES.C Y EN LA IMPLEMENTACION DE COLAS==

=== la variable se encuentra en la primer linea de codigo...===

5

sh: 1: cls: not found

Ingresar la cantidad de elementos de la cola: 5

| CLAVE NODO 1 | -> 2

Imprimiendo cola con divisores totales (verdadero) y parciales (falso)...

Valor del elemento // Es divisor total/parcial

Clave: 2 Valor: Verdadero

Ingresar 1 para SEGUIR, 0 para NO SEGUIR:

Ejercicio 6: no permite estructuras vacías.

Ejercicio 7: permite valores negativos en tiempo de espera. Permite cargar cero en tiempo de atención, con lo cual, entra en un loop infinito. Solamente funciona con el ejemplo de la guía

GRUPO 1 – Correcciones Trabajo Práctico: COLAS

The image shows a Visual Studio Code editor with a C project. The file explorer on the left shows the project structure. The main editor displays the source code for 'validaciones.c'. The code defines a linked list structure for a cola menu and implements functions to add and print items. The terminal window at the bottom shows the execution of the program, where the user enters '1' for the first cola, and the program prints 'Clave: -1' and 'Cola 1:'. The call stack on the left shows the function 'menuPunto7()' in 'menu.c'.

```

// validaciones.h
#ifndef VALIDACIONES_H
#define VALIDACIONES_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Cola {
    int clave;
    char *mensaje_error;
    struct Cola *siguiente;
} Cola;

Cola *ingresoDatosNumericos(char * mensaje_error);
Cola *ingresoDatosNumericosPositivos(char * mensaje_error);

#endif

// validaciones.c
#include "validaciones.h"

Cola *ingresoDatosNumericos(char * mensaje_error){
    while (!chequeo){
        printf("Ingrese la cantidad de elementos a cargar en la cola %d (max 95)\n", 1);
        int clave;
        scanf("%d", &clave);
        if (clave < 0 || clave > 95){
            printf("Error: la cantidad de elementos a cargar en la cola %d debe ser mayor o igual a 0 y menor o igual a 95.\n", 1);
            continue;
        }
        Cola *nuevoNodo = (Cola *) malloc(sizeof(Cola));
        nuevoNodo->clave = clave;
        nuevoNodo->mensaje_error = NULL;
        nuevoNodo->siguiente = NULL;
        Cola *actual = cola;
        while (actual != NULL){
            actual = actual->siguiente;
        }
        nuevoNodo->siguiente = actual;
        cola = nuevoNodo;
        printf("Cola %d:\n", 1);
        Cola *actual2 = cola;
        while (actual2 != NULL){
            printf("Clave: %d\n", actual2->clave);
            actual2 = actual2->siguiente;
        }
        printf("Cola %d:\n", 2);
        Cola *actual3 = cola;
        while (actual3 != NULL){
            printf("Clave: %d\n", actual3->clave);
            actual3 = actual3->siguiente;
        }
        printf("Cola %d:\n", 3);
        Cola *actual4 = cola;
        while (actual4 != NULL){
            printf("Clave: %d\n", actual4->clave);
            actual4 = actual4->siguiente;
        }
        printf("Cola %d:\n", 4);
        Cola *actual5 = cola;
        while (actual5 != NULL){
            printf("Clave: %d\n", actual5->clave);
            actual5 = actual5->siguiente;
        }
        printf("Cola %d:\n", 5);
        Cola *actual6 = cola;
        while (actual6 != NULL){
            printf("Clave: %d\n", actual6->clave);
            actual6 = actual6->siguiente;
        }
        printf("Cola %d:\n", 6);
        Cola *actual7 = cola;
        while (actual7 != NULL){
            printf("Clave: %d\n", actual7->clave);
            actual7 = actual7->siguiente;
        }
        printf("Cola %d:\n", 7);
        Cola *actual8 = cola;
        while (actual8 != NULL){
            printf("Clave: %d\n", actual8->clave);
            actual8 = actual8->siguiente;
        }
        printf("Cola %d:\n", 8);
        Cola *actual9 = cola;
        while (actual9 != NULL){
            printf("Clave: %d\n", actual9->clave);
            actual9 = actual9->siguiente;
        }
        printf("Cola %d:\n", 9);
        Cola *actual10 = cola;
        while (actual10 != NULL){
            printf("Clave: %d\n", actual10->clave);
            actual10 = actual10->siguiente;
        }
        printf("Cola %d:\n", 10);
        Cola *actual11 = cola;
        while (actual11 != NULL){
            printf("Clave: %d\n", actual11->clave);
            actual11 = actual11->siguiente;
        }
        printf("Cola %d:\n", 11);
        Cola *actual12 = cola;
        while (actual12 != NULL){
            printf("Clave: %d\n", actual12->clave);
            actual12 = actual12->siguiente;
        }
        printf("Cola %d:\n", 12);
        Cola *actual13 = cola;
        while (actual13 != NULL){
            printf("Clave: %d\n", actual13->clave);
            actual13 = actual13->siguiente;
        }
        printf("Cola %d:\n", 13);
        Cola *actual14 = cola;
        while (actual14 != NULL){
            printf("Clave: %d\n", actual14->clave);
            actual14 = actual14->siguiente;
        }
        printf("Cola %d:\n", 14);
        Cola *actual15 = cola;
        while (actual15 != NULL){
            printf("Clave: %d\n", actual15->clave);
            actual15 = actual15->siguiente;
        }
        printf("Cola %d:\n", 15);
        Cola *actual16 = cola;
        while (actual16 != NULL){
            printf("Clave: %d\n", actual16->clave);
            actual16 = actual16->siguiente;
        }
        printf("Cola %d:\n", 16);
        Cola *actual17 = cola;
        while (actual17 != NULL){
            printf("Clave: %d\n", actual17->clave);
            actual17 = actual17->siguiente;
        }
        printf("Cola %d:\n", 17);
        Cola *actual18 = cola;
        while (actual18 != NULL){
            printf("Clave: %d\n", actual18->clave);
            actual18 = actual18->siguiente;
        }
        printf("Cola %d:\n", 18);
        Cola *actual19 = cola;
        while (actual19 != NULL){
            printf("Clave: %d\n", actual19->clave);
            actual19 = actual19->siguiente;
        }
        printf("Cola %d:\n", 19);
        Cola *actual20 = cola;
        while (actual20 != NULL){
            printf("Clave: %d\n", actual20->clave);
            actual20 = actual20->siguiente;
        }
        printf("Cola %d:\n", 20);
        Cola *actual21 = cola;
        while (actual21 != NULL){
            printf("Clave: %d\n", actual21->clave);
            actual21 = actual21->siguiente;
        }
        printf("Cola %d:\n", 21);
        Cola *actual22 = cola;
        while (actual22 != NULL){
            printf("Clave: %d\n", actual22->clave);
            actual22 = actual22->siguiente;
        }
        printf("Cola %d:\n", 22);
        Cola *actual23 = cola;
        while (actual23 != NULL){
            printf("Clave: %d\n", actual23->clave);
            actual23 = actual23->siguiente;
        }
        printf("Cola %d:\n", 23);
        Cola *actual24 = cola;
        while (actual24 != NULL){
            printf("Clave: %d\n", actual24->clave);
            actual24 = actual24->siguiente;
        }
        printf("Cola %d:\n", 24);
        Cola *actual25 = cola;
        while (actual25 != NULL){
            printf("Clave: %d\n", actual25->clave);
            actual25 = actual25->siguiente;
        }
        printf("Cola %d:\n", 25);
        Cola *actual26 = cola;
        while (actual26 != NULL){
            printf("Clave: %d\n", actual26->clave);
            actual26 = actual26->siguiente;
        }
        printf("Cola %d:\n", 26);
        Cola *actual27 = cola;
        while (actual27 != NULL){
            printf("Clave: %d\n", actual27->clave);
            actual27 = actual27->siguiente;
        }
        printf("Cola %d:\n", 27);
        Cola *actual28 = cola;
        while (actual28 != NULL){
            printf("Clave: %d\n", actual28->clave);
            actual28 = actual28->siguiente;
        }
        printf("Cola %d:\n", 28);
        Cola *actual29 = cola;
        while (actual29 != NULL){
            printf("Clave: %d\n", actual29->clave);
            actual29 = actual29->siguiente;
        }
        printf("Cola %d:\n", 29);
        Cola *actual30 = cola;
        while (actual30 != NULL){
            printf("Clave: %d\n", actual30->clave);
            actual30 = actual30->siguiente;
        }
        printf("Cola %d:\n", 30);
        Cola *actual31 = cola;
        while (actual31 != NULL){
            printf("Clave: %d\n", actual31->clave);
            actual31 = actual31->siguiente;
        }
        printf("Cola %d:\n", 31);
        Cola *actual32 = cola;
        while (actual32 != NULL){
            printf("Clave: %d\n", actual32->clave);
            actual32 = actual32->siguiente;
        }
        printf("Cola %d:\n", 32);
        Cola *actual33 = cola;
        while (actual33 != NULL){
            printf("Clave: %d\n", actual33->clave);
            actual33 = actual33->siguiente;
        }
        printf("Cola %d:\n", 33);
        Cola *actual34 = cola;
        while (actual34 != NULL){
            printf("Clave: %d\n", actual34->clave);
            actual34 = actual34->siguiente;
        }
        printf("Cola %d:\n", 34);
        Cola *actual35 = cola;
        while (actual35 != NULL){
            printf("Clave: %d\n", actual35->clave);
            actual35 = actual35->siguiente;
        }
        printf("Cola %d:\n", 35);
        Cola *actual36 = cola;
        while (actual36 != NULL){
            printf("Clave: %d\n", actual36->clave);
            actual36 = actual36->siguiente;
        }
        printf("Cola %d:\n", 36);
        Cola *actual37 = cola;
        while (actual37 != NULL){
            printf("Clave: %d\n", actual37->clave);
            actual37 = actual37->siguiente;
        }
        printf("Cola %d:\n", 37);
        Cola *actual38 = cola;
        while (actual38 != NULL){
            printf("Clave: %d\n", actual38->clave);
            actual38 = actual38->siguiente;
        }
        printf("Cola %d:\n", 38);
        Cola *actual39 = cola;
        while (actual39 != NULL){
            printf("Clave: %d\n", actual39->clave);
            actual39 = actual39->siguiente;
        }
        printf("Cola %d:\n", 39);
        Cola *actual40 = cola;
        while (actual40 != NULL){
            printf("Clave: %d\n", actual40->clave);
            actual40 = actual40->siguiente;
        }
        printf("Cola %d:\n", 40);
        Cola *actual41 = cola;
        while (actual41 != NULL){
            printf("Clave: %d\n", actual41->clave);
            actual41 = actual41->siguiente;
        }
        printf("Cola %d:\n", 41);
        Cola *actual42 = cola;
        while (actual42 != NULL){
            printf("Clave: %d\n", actual42->clave);
            actual42 = actual42->siguiente;
        }
        printf("Cola %d:\n", 42);
        Cola *actual43 = cola;
        while (actual43 != NULL){
            printf("Clave: %d\n", actual43->clave);
            actual43 = actual43->siguiente;
        }
        printf("Cola %d:\n", 43);
        Cola *actual44 = cola;
        while (actual44 != NULL){
            printf("Clave: %d\n", actual44->clave);
            actual44 = actual44->siguiente;
        }
        printf("Cola %d:\n", 44);
        Cola *actual45 = cola;
        while (actual45 != NULL){
            printf("Clave: %d\n", actual45->clave);
            actual45 = actual45->siguiente;
        }
        printf("Cola %d:\n", 45);
        Cola *actual46 = cola;
        while (actual46 != NULL){
            printf("Clave: %d\n", actual46->clave);
            actual46 = actual46->siguiente;
        }
        printf("Cola %d:\n", 46);
        Cola *actual47 = cola;
        while (actual47 != NULL){
            printf("Clave: %d\n", actual47->clave);
            actual47 = actual47->siguiente;
        }
        printf("Cola %d:\n", 47);
        Cola *actual48 = cola;
        while (actual48 != NULL){
            printf("Clave: %d\n", actual48->clave);
            actual48 = actual48->siguiente;
        }
        printf("Cola %d:\n", 48);
        Cola *actual49 = cola;
        while (actual49 != NULL){
            printf("Clave: %d\n", actual49->clave);
            actual49 = actual49->siguiente;
        }
        printf("Cola %d:\n", 49);
        Cola *actual50 = cola;
        while (actual50 != NULL){
            printf("Clave: %d\n", actual50->clave);
            actual50 = actual50->siguiente;
        }
        printf("Cola %d:\n", 50);
        Cola *actual51 = cola;
        while (actual51 != NULL
```

The screenshot shows a Visual Studio Code editor with a C program for a queue simulation. The program is named 'tp_4_colas.c' and is part of a project '04-trabajoPractico-Colas'. The code uses a linked list to represent the queue. The program prompts the user to enter the time of service for each client. The output shows the queue being processed and the service times for three clients: 1, 2, and 3. The program is running in a terminal window.

Code Snippet:

```

406 // EJERCICIO 7
407
408 // */
409 // retornar la cola con los resultados de la atenciones de los clientes. Ver el ejemplo en el PDF del TP.
410 // retornar en la clave el nro de cola que lo atendio y en el void* de cada elemento los textos "Cliente 1 C
411 // Segun el ejemplo el primer item de la cola seria clave: 3, void* "Cliente 1 Cola 3".
412 // Si las colas estan vacias, retornar cola vacia.
413 Cola c_ej7_atenderclientes(Cola c1, Cola c2, Cola c3, int tiempoatencion){
414     TipoElemento nodo = te_crear(0), nodo2 = te_crear(0), nodo3 = te_crear(0), nodoaux; //crea la clave en 0
415     Cola resultado = c_crear();
416     int cont = 1, cont2 = 1, cont3 = 1;
417     Cola c1_aux = c_crear(); Cola c2_aux = c_crear(); Cola c3_aux = c_crear(); //se crean las colas auxiliares
418     c1_aux = c_ej2_copiar(c1); c2_aux = c_ej2_copiar(c2); c3_aux = c_ej2_copiar(c3); //se COPIAN las colas or
419     bool termino1 = false, termino2 = false, termino3 = false;
420

```

Terminal Output:

```

Cola 1:
-----
Imprimiendo las Claves de la Cola
Clave: -1
Clave: 6

Cola 2:
-----
Imprimiendo las Claves de la Cola
Clave: 1
Clave: 2
Clave: 8

Cola 3:
-----
Imprimiendo las Claves de la Cola
Clave: -9

```

Shell Prompt:

```

sh: 1: pause: not found
ingrese el tiempo de atencion:
0

```

GRUPO 1 – Correcciones Trabajo Práctico: COLAS

```
04-trabajoPractico-Colas > C main.c > main()
3 void main() {
6     do {
21         switch (opcion) {
36             break;
37         case 7:
38             menuPunto7();
39             break;
40         case 0:
41             printf("Saliendo...\n");
42             system("pause");
43             break;
44         default:
45             printf("AVISO: Ingrese un numero parte de las opciones.\n");
46             break;
```

== la variable se encuentra en la primer linea de codigo...==

```
7
sh: 1: cls: not found
Ingrese la cantidad de elementos a cargar en la cola 1 (max 98)
-> 2
| CLAVE NODO 1 | -> 3
| CLAVE NODO 2 | -> 2
sh: 1: cls: not found
Ingrese la cantidad de elementos a cargar en la cola 2 (max 97)
-> 2
| CLAVE NODO 1 | -> 5
| CLAVE NODO 2 | -> 3
sh: 1: cls: not found
Ingrese la cantidad de elementos a cargar en la cola 3 (max 96)
-> 1
| CLAVE NODO 1 | -> 10
sh: 1: cls: not found
Cola 1:
-----
Imprimiendo las Claves de la Cola
-----
Clave: 3
Clave: 2
```

```
04-trabajoPractico-Colas > C main.c > main()
3 void main() {
6     do {
21         switch (opcion) {
36             break;
37         case 7:
38             menuPunto7();
39             break;
40         case 0:
41             printf("Saliendo...\n");
42             system("pause");
43             break;
44         default:
45             printf("AVISO: Ingrese un numero parte de las opciones.\n");
46             break;
```

5

```
CLIENTES
Cliente 1 Cola 1
Cliente 1 Cola 2
Cliente 2 Cola 1
Cliente 2 Cola 2
Cliente 1 Cola 3
-----
Imprimiendo las Claves de la Cola
-----
Clave: 3
Clave: 2

-----
Imprimiendo las Claves de la Cola
-----
Clave: 5
Clave: 3

-----
Imprimiendo las Claves de la Cola
-----
Clave: 10
```