

Universidad Autónoma De Baja California



Grupo-941

23/09/2025

Alumno:

Gastelum fierro Brian de Jesus

Ejercicio 1

Analiza el siguiente código en C y determina la salida del programa. Explica qué tipo de memoria (estática, dinámica o automática) se usa en cada caso.

```
ej1.cpp  U X
memoria > ej1.cpp > main()
1 #include <stdio.h>
2
3 int global_var = 10;
4
5 void func() {
6     int local_var=20;
7     static int static_var=30;
8     local_var++;
9     static_var++;
10    printf("local_var: %d, static_var: %d",local_var,static_var);
11 }
12 int main()
13 {
14     func();
15     func();
16     return 0;
17 }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
TERMINAL
PS C:\paradigmas\memoria> ./ej1
local_var: 21, static_var: 31local_var: 21, static_var: 32
PS C:\paradigmas\memoria>
```

Se utiliza

En la variable global_var y static_var se usa memoria estatica

En la variable local_var se usa memoria automatica

Ejercicio 2

Analiza el siguiente código y encuentra el error relacionado con la memoria dinámica.

```
ej1.cpp  U  ej2.cpp  U  X
memoria > ej2.cpp > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void allocate_memory()
5 {
6     int *arr=(int *)malloc(5 * sizeof(int));
7     for(int i=0; i<5; i++)
8     {
9         arr[i]=i * 2;
10    }
11    printf("arr[2]:%d\n",arr[2]);
12 }
13
14 int main()
15 {
16     allocate_memory();
17     return 0;
18 }
19
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\paradigmas\memoria> ./ej2
arr[2]:4
PS C:\paradigmas\memoria>
```

1. ¿Qué problema de manejo de memoria presenta este código?

Se reserva memoria dinámica con malloc dentro de allocate_memory().

Esa memoria nunca se libera con free.

2. ¿Cómo podríamos solucionarlo?

Liberando la memoria con free(arr) una vez que ya no la necesitamos

Ejercicio 3

El siguiente código usa pthread para crear un subprocesso que modifica una variable global. Analiza la ejecución y predice la salida.

The screenshot shows a code editor with two tabs: 'ej2.cpp' and 'ej3.cpp'. The 'ej3.cpp' tab is active, showing the following C++ code:

```
memoria > ej3.cpp > ...
1 #include <stdio.h>
2 #include <pthread.h>      #include errors detected. Please update
3
4 int shared_var = 5; // Variable compartida entre hilos
5
6 void *modify_variable(void *arg) {
7     shared_var += 10;
8     return NULL;
9 }
10
11 int main() {
12     pthread_t thread;
13     pthread_create(&thread, NULL, modify_variable, NULL);
14     pthread_join(thread, NULL);
15     printf("shared_var: %d\n", shared_var);
16     return 0;
17 }
18
```

The code includes a warning about '#include errors detected. Please update' for the pthread.h header. The code defines a global variable 'shared_var' and a function 'modify_variable' that increments it by 10. It then creates a thread to call 'modify_variable' and joins with it before printing the value of 'shared_var'.

The screenshot shows a terminal window with the following output:

```
PS C:\paradigmas\memoria> ./ej3
shared_var: 15
PS C:\paradigmas\memoria>
```

The terminal shows the command 'PS C:\paradigmas\memoria> ./ej3' being run, followed by the output 'shared_var: 15', and then the prompt 'PS C:\paradigmas\memoria>' again.

¿Cuál es la salida esperada del programa?

Shared_Var : 15

¿Qué pasaría si removemos pthread_join(thread,NULL)?

Me imprime 5