Trabajo Práctico Nº 1

Materia: Programación Eficiente

Integrantes: Souza Rodrigo, Aimaro Franco

Listado de tests

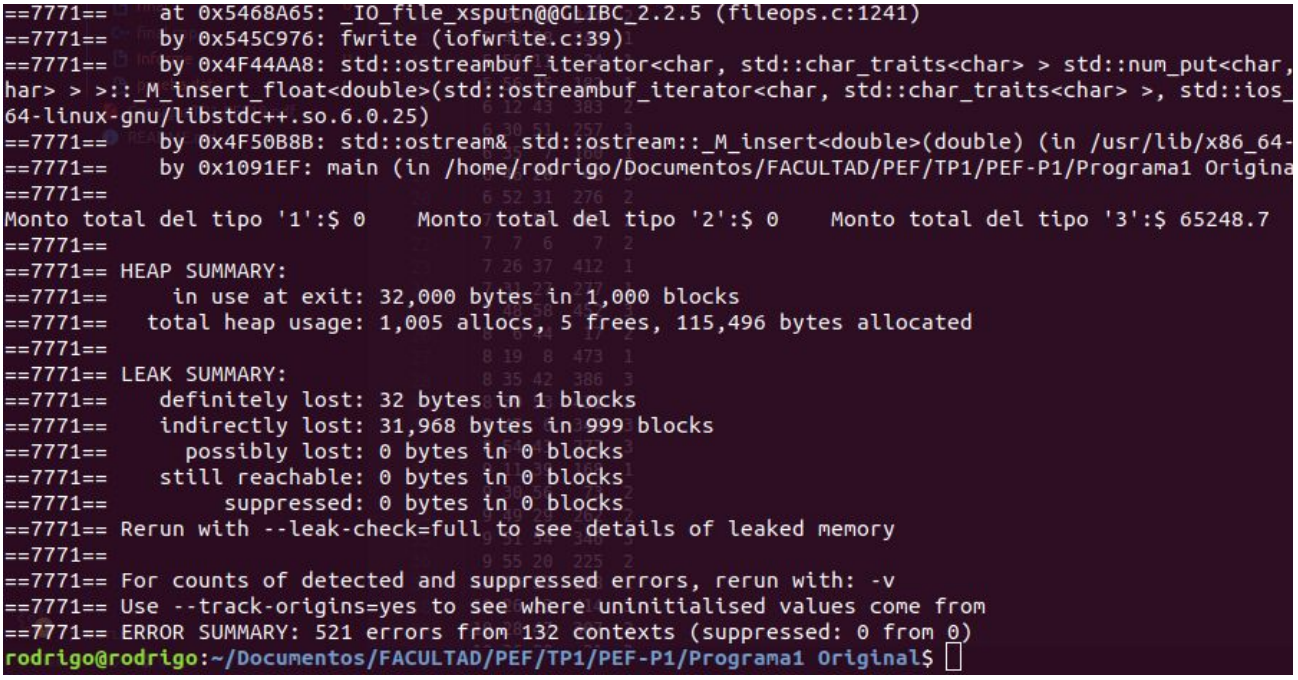
1. Verificar el uso de los métodos solicitados.
2. Verificar que la memoria sea liberada correctamente.
3. Verificar los cálculos del monto total realizados por cada código de tarifa.
4. Verificar que la función lea/abra un archivo x.
5. Verificar que una función x devuelva la cantidad de llamadas y tiempo acumulado total de cada código tarifa.
6. Corroborar la declaración de las bibliotecas, métodos, estructuras y verificación de Código Muerto.
7. Buenas Prácticas.

TEST 2 : Verificar que la memoria sea liberada correctamente.

Ejecutamos valgrind --tool=memcheck ./final para verificar si la memoria estaba siendo liberada correctamente

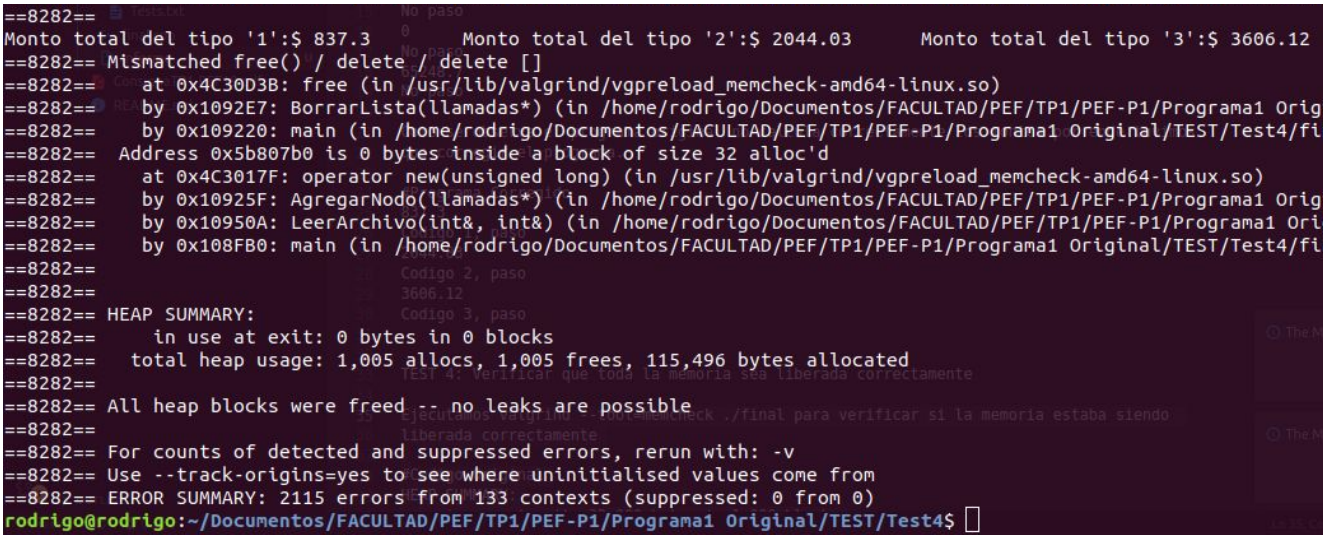
#Código Original
HEAP SUMMARY:
in use at exit: 32,000 bytes in 1,000 blocks
total heap usage: 1,005 allocs, 5 frees, 115,496 bytes allocated

LEAK SUMMARY:
definitely lost: 32 bytes in 1 blocks
indirectly lost: 31,968 bytes in 999 blocks
possibly lost: 0 bytes in 0 blocks
still reachable: 0 bytes in 0 blocks
suppressed: 0 bytes in 0 blocks



#Código Corregido
HEAP SUMMARY:
in use at exit: 0 bytes in 0 blocks
total heap usage: 1,005 allocs, 1,005 frees, 115,496 bytes allocated

All heap blocks were freed -- no leaks are possible



En esta prueba se puede observar que el código original no libera la memoria correctamente y deja mucha basura. En principio el método para liberar la memoria estaba pero no era llamado por el main, luego sucedió que el método no liberaba la memoria así que se corrigió; una vez corregido el programa libera correctamente la memoria.

TEST 3 : Verificar los cálculos del monto total realizados por cada código de tarifa.

Se hizo un archivo shell script (sh) para verificar si los montos que arrojaba el programa eran correcto.

Primero se corrió el programa original para probar la funcionalidad y tuvimos que corregir para hacer que calcule lo que se pretende, ya que al principio no lo hacía correctamente.

Cuando se ejecuta el sh el programa queda en espera, en ese momento se debe ingresar el nombre del archivo "prueba.dat". Por defecto se ejecuta el programa corregido, para ejecutar el no corregido descomentar la línea #9.

Lo que hace el programa es “printear” primero el resultado del monto y luego informa si el monto pasó o no la prueba realizada en el sh.

#Programa Original
4.58225e-42
No paso
0
No paso
65248.7
No paso

Como se observa el programa original no calcula correctamente los montos por ende tuvimos que corregir el programa.

#Programa Corregido
837.3
Codigo 1, paso
2044.03
Codigo 2, paso
3606.12
Codigo 3, paso

TEST 7: Buenas prácticas

Cuando empezamos a leer el programa nos costó entender el código a nivel sintáctico ya que no tenía las tabulaciones por ende tuvimos que aplicar el comando "indent -bl" para que el código fuera más legible.

TEST 6: verificación de código muerto

PAra poder controlar código muerto que nunca es ejecutado o funciones que jamás se llaman utilizamos la herramienta *gcov*, la misma genera recuentos exactos del número de veces que se ejecuta cada instrucción del programa y anota el código fuente con esa información.

A Continuación detalles de la prueba:
La versión utilizada es la *gcov 7.4.0.*

Comando utilizado:

g++ -fprofile-arcs -ftest-coverage final.cpp

luego ejecuto el programa.
luego ejecuto gcov final.cpp -> dependiendo de lo que quiera visualizar podemos agregarle parámetros, por ejemplo si necesitamos contabilizar la ejecución de bloques agregamos ***-a*** si necesitamos un informe de porcentajes, podemos hacer un ***-b.***
Las pruebas que obtuvimos fueron que en el programa original, habian 21 lineas muertas que no se ejecutaban, y la versión mejorada del programa (sin optimizar) detectamos solo 17.

Conclusiones finales:

Podemos decir que dedicamos mucho tiempo a intentar entender el funcionamiento del programa original, porque además de tener errores que saltaban a la vista, también no estábamos seguros de que cumpliera con las funciones correctas.
Por eso, antes de empezar a realizar pruebas, decidimos corregirlo para que funcione como corresponde.
Una vez, que el programa funciona, nos dedicamos a investigar shell script para poder dejar escritos los test necesarios. Algunos pudimos resolverlos, y otros no, por eso acudimos a probar el programa con las herramientas que vimos en la 1er parte de la materia.
Al final, pudimos testear todos casi todos los puntos, y obtuvimos un panorama mucho más claro de las pruebas que pasaba o no el programa.