Evaluación de Código C

Evaluación de Código C

1. BLOQUE DLC: Reglas

Ejemplo 1

1. Define la regla que se incumple y proón una alternativa más adecuada según el
   SEI CERT C.
   La regla mas adecuada es :

DCL30-C. Declare objects with appropriate storage durations

```
(base) dimi@dimi:~/cpp$ gcc ejemplo3-1.c -o ejemplo3-1 -std=c11
(base) dimi@dimi:~/cpp$ ./ejemplo3-1
1 - Mi Cadena
2 - Todo va bien
3 - ----
4 - Mi Cadena
```

Ejemplo 2

1. ¿Qué hace el siguiente segmento de código?

This code discribes : a sctructure called **flexArrayStruct** with 2 variable an integer num, and an array data.
the function : func, does a dynamic allocation of memory initializing the arrays with 1

2. De haber algún problema ¿Podrías decir la línea en la que se encuentra?

```
struct flexArrayStruct *structP = (struct flexArrayStruct *) malloc(sizeof
(struct flexArrayStruct) + sizeof(int) * (array_size - 1) ) ;
```

3. Define la regla que se incumple y propón una alternativa correcta siguiendo el SEI CERT C.

- DCL30-C. Declare objects with appropriate storage durations
- DCL38-C. Use the correct syntax when declaring a flexible array member

Ejemplo 3

1. ¿Que hace el siguiente segmento de codigo si invocamos la funcion func con un 0?
   If we call the function with 0 it will :

    1. (i) will be initialized to (4)
    2. f(int i) will take the value of (4)
    3. (0) will be evaluate in the switch-case
    4. finallyy print -> (17)
2. De haber algun problema ¿Podrıas decir la lınea en la que se encuentra?

   noncompliant code example declares variables and contains executable statements before the first case label within the switch statement

```
      int i = 4;  statement will never be executed [-Wswitch-unreachable]
```

3. Crea un fichero con un main y ejecuta el segmento de codigo.
4. Propon una solucion al ejemplo que cumpla con las normal del CMU

DCL41-C. Do not declare variables inside a switch statement before the first case label

5. Realiza un analisis estatico del codigo erroneo y copia en tu solucion el resultado.
   Utilizalasherramientas:
   (a) rats
   (b) cppchecker
   (c) splint
   (d) vera++
   (e) valgrind

```
(base) dimi@dimi:~/cpp$ splint ejemplo3-3.c
Splint 3.1.2 --- 21 Feb 2021

ejemplo3-3.c: (in function func)
ejemplo3-3.c:45:12: Fall through case (no preceding break)
  Execution falls through from the previous case (use @fallthrough@ to mark
  fallthrough cases). (Use -casebreak to inhibit warning)
ejemplo3-3.c:48:12: Fall through case (no preceding break)
ejemplo3-3.c:43:17: Statement after switch is not a case: int i = 4
  The first statement after a switch is not a case. (Use -firstcase to
inhibit
  warning)

Finished checking --- 3 code warnings
```

```
(base) dimi@dimi:~/cpp$ splint ejemplo3-3.c
Splint 3.1.2 --- 21 Feb 2021

Finished checking --- no warnings
```

```
(base) dimi@dimi:~/cpp$ valgrind --leak-check=yes ./ejemplo3-3

==350572== Memcheck, a memory error detector
==350572== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==350572== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright
info
==350572== Command: ./ejemplo3-3
==350572==
==350572==
==350572== HEAP SUMMARY:
==350572==     in use at exit: 0 bytes in 0 blocks
==350572==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==350572==
==350572== All heap blocks were freed -- no leaks are possible
==350572==
==350572== For lists of detected and suppressed errors, rerun with: -s
==350572== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
(base) dimi@dimi:~/cpp$ vera++   ejemplo3-3.c
ejemplo3-3.c:1: leading empty line(s)
ejemplo3-3.c:1: no copyright notice found
ejemplo3-3.c:2: trailing whitespace
ejemplo3-3.c:8: trailing whitespace
ejemplo3-3.c:14: trailing whitespace
ejemplo3-3.c:17: trailing whitespace
ejemplo3-3.c:19: line is longer than 100 characters
ejemplo3-3.c:20: trailing whitespace
ejemplo3-3.c:21: trailing whitespace
ejemplo3-3.c:24: trailing whitespace
ejemplo3-3.c:26: trailing whitespace
ejemplo3-3.c:28: trailing whitespace
ejemplo3-3.c:30: trailing whitespace
ejemplo3-3.c:30: line is longer than 100 characters
ejemplo3-3.c:31: trailing whitespace
ejemplo3-3.c:32: trailing whitespace
ejemplo3-3.c:33: trailing whitespace
ejemplo3-3.c:34: trailing whitespace
ejemplo3-3.c:39: trailing whitespace
ejemplo3-3.c:41: trailing whitespace
ejemplo3-3.c:42: trailing whitespace
ejemplo3-3.c:43: trailing whitespace
ejemplo3-3.c:44: trailing whitespace
ejemplo3-3.c:45: trailing whitespace
ejemplo3-3.c:46: trailing whitespace
```

```
ejemplo3-3.c:47: trailing whitespace
ejemplo3-3.c:48: trailing whitespace
ejemplo3-3.c:49: trailing whitespace
ejemplo3-3.c:49: comma should not be preceded by whitespace
ejemplo3-3.c:49: closing curly bracket not in the same line or column
ejemplo3-3.c:50: trailing whitespace
ejemplo3-3.c:51: trailing whitespace
ejemplo3-3.c:51: closing curly bracket not in the same line or column
ejemplo3-3.c:53: no newline at end of file
ejemplo3-3.c:53: trailing whitespace
ejemplo3-3.c:53: trailing empty line(s)
```

==========

2. BLOQUE DLC: Recomendaciones**

Ejercicio 1

DCL10-C. Maintain the contract between the writer and caller of variadic functions

¿Qué hace el siguiente segmento de código?

The function average() takes a variable number of positive integer arguments and returns the average of those arguments.

¿Para qué se utiliza la variable va_eol?

The va_eol enumeration signals the end of the variable-length argument list to exit of the while loop, allowing the function to process all of the provided arguments.

•Incorpora el segmento de código en un programa .c de tal forma que no encontremos nigún warning cuando compilamos en gcc con los siguientes parámetros (std=c11). Dado que es C, elimina aquellos que no aplican. Escribe en la respuesta aquellos que se ven afectados y son eliminados.
los que no aplican:

```
(base) dimi@dimi:~/cpp$ gcc -Werror -Wall -Wextra -pedantic -Wcast-align -Wcast-qual -Wctor-dtor-privacy -Wdisabled-optimization -Wformat=2 -Winit-self -Wlogical-op -Wmissing-include-dirs -Wnoexcept -Wold-style-cast -Woverloaded-virtual -Wredundant-decls -Wshadow -Wsign-promo -Wstrict-null-sentinel -Wstrict-overflow=5 -Wundef -Wno-unused -Wno-variadic-macros -Wno-parentheses -fdiagnostics-show-option ejemplo4-1.c -std=c11
cc1: error: command-line option ‘-Wctor-dtor-privacy’ is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option ‘-Wnoexcept’ is valid for C++/ObjC++ but not
```

```
for C [-Werror]
cc1: error: command-line option ‘-Wold-style-cast’ is valid for C++/ObjC++
but not for C [-Werror]
cc1: error: command-line option ‘-Woverloaded-virtual’ is valid for
C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option ‘-Wsign-promo’ is valid for C++/ObjC++ but
not for C [-Werror]
cc1: error: command-line option ‘-Wstrict-null-sentinel’ is valid for
C++/ObjC++ but not for C [-Werror]
```

```
================================================================================
=========
```

```
(base) dimi@dimi:~/cpp$ gcc -Werror -Wall -Wextra -pedantic -Wcast-align -
Wcast-qual -Wdisabled-optimization -Wformat=2 -Winit-self -Wlogical-op -
Wmissing-include-dirs -Wredundant-decls -Wshadow -Wstrict-overflow=5 -Wundef
-Wno-unused -Wno-variadic-macros -Wno-parentheses -fdiagnostics-show-option
ejemplo4-1.c -std=c11
(base) dimi@dimi:~/cpp$ valgrind --leak-check=yes ./ejemplo4
==381298== Memcheck, a memory error detector
==381298== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==381298== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright
info
==381298== Command: ./ejemplo4
==381298==
The average is 3
==381298==
==381298== HEAP SUMMARY:
==381298==     in use at exit: 0 bytes in 0 blocks
==381298==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==381298==
==381298== All heap blocks were freed -- no leaks are possible
==381298==
==381298== For lists of detected and suppressed errors, rerun with: -s
==381298== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
(base) dimi@dimi:~/cpp$ gcc -Werror -Wall -Wextra -pedantic -Wcast-align -Wcast-qual -Wctor-dtor-privacy -Wdisabled-optimization -Wformat=2 -Winit-self -Wlogical-op -Wmissing-include-dirs -Wnoexcept -Wold
-style-cast -Woverloaded-virtual -Wredundant-decls -Wshadow -Wsign-promo -Wstrict-null-sentinel -Wstrict-overflow=5 -Wundef -Wno-unused -Wno-variadic-macros -Wno-parentheses -fdiagnostics-show-option ejem
plo4-1.c -std=c11
cc1: error: command-line option '-Wctor-dtor-privacy' is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option '-Wnoexcept' is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option '-Wold-style-cast' is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option '-Woverloaded-virtual' is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option '-Wsign-promo' is valid for C++/ObjC++ but not for C [-Werror]
cc1: error: command-line option '-Wstrict-null-sentinel' is valid for C++/ObjC++ but not for C [-Werror]
cc1: all warnings being treated as errors
(base) dimi@dimi:~/cpp$ gcc -Werror -Wall -Wextra -pedantic -Wcast-align -Wcast-qual -Wdisabled-optimization -Wformat=2 -Winit-self -Wlogical-op -Wmissing-include-dirs -Wredundant-decls -Wshadow -Wstrict-
overflow=5 -Wundef -Wno-unused -Wno-variadic-macros -Wno-parentheses -fdiagnostics-show-option ejemplo4-1.c -std=c11
(base) dimi@dimi:~/cpp$ valgrind --leak-check=yes ./ejemplo4
==381298== Memcheck, a memory error detector
==381298== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==381298== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==381298== Command: ./ejemplo4
==381298==
The average is 3
==381298==
==381298== HEAP SUMMARY:
==381298==     in use at exit: 0 bytes in 0 blocks
==381298==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==381298==
==381298== All heap blocks were freed -- no leaks are possible
==381298==
==381298== For lists of detected and suppressed errors, rerun with: -s
==381298== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) dimi@dimi:~/cpp$
```

==========

Ejercicio 2.1

¿Qué hace el siguiente segmento de código?

The function calculates the factorial of an integer, and has two parameters: **n, and result**

MEM05-C. Avoid large stack allocations
Recursive functions must ensure that they do not exhaust the stack as a result of excessive recursions.

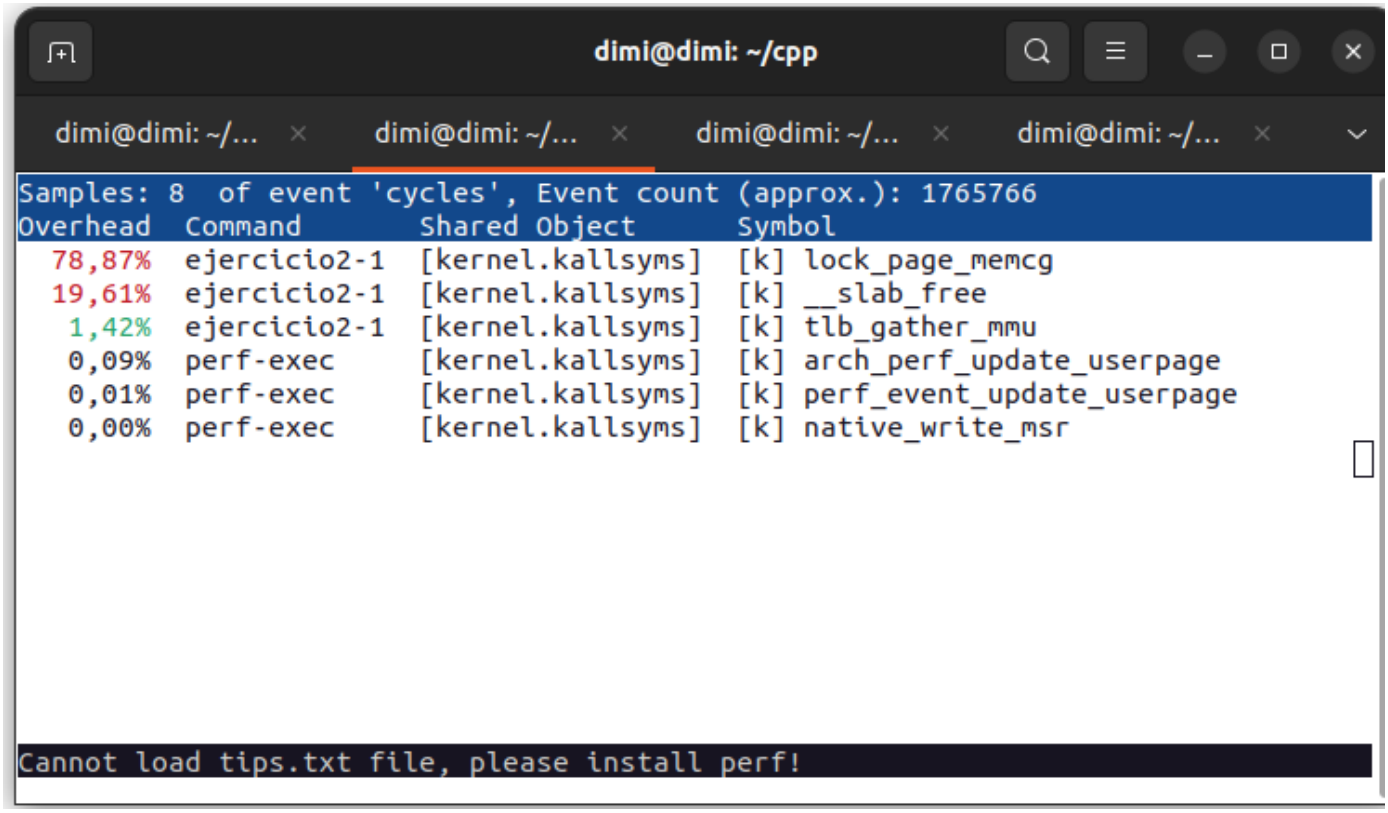
```
i f ( i <= 1 ) {
return 1;
}
r e t u r n i * f a c t o r i a l ( i – 1) ;
```

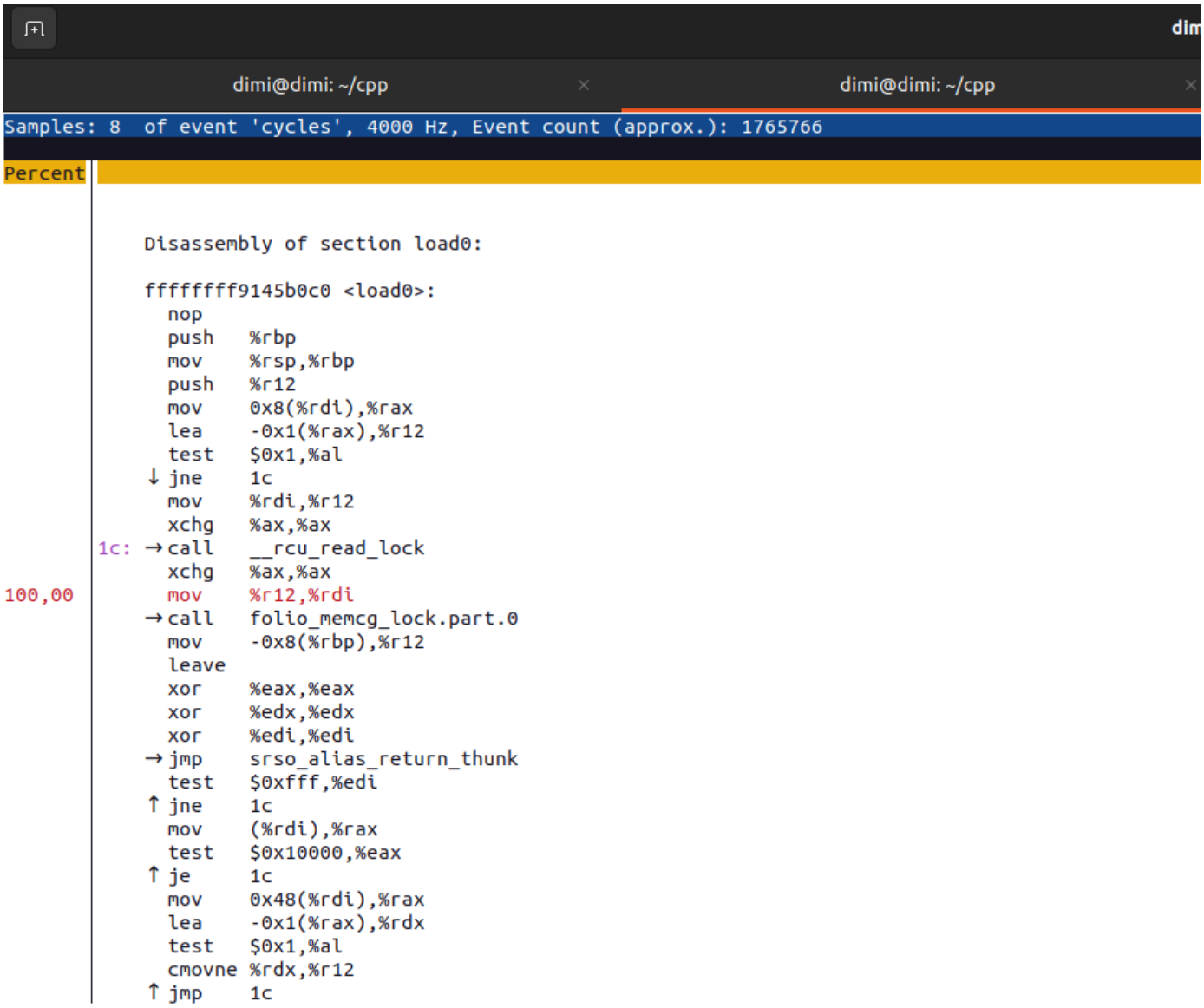
```
(base) dimi@dimi:~/cpp$ ./ejercicio2-1
Factorial of 12 is 479001600
(base) dimi@dimi:~/cpp$
```

• Instala la herramienta perf para realizar el profiling de la aplicación. Se puede instalar con apt.

```
(base) dimi@dimi:~/cpp$ sudo apt install linux-tools-$(uname -r) linux-tools-generic
```

• El programa permite mostrar el código desensamblado de la aplicación, adjunta alguna captura.


dimi@dimi: ~/cpp
dimi@dimi: ~/...
dimi@dimi: ~/...
dimi@dimi: ~/...
dimi@dimi: ~/...
Samples: 8 of event 'cycles', Event count (approx.): 1765766
Overhead Command Shared Object Symbol
78,87% ejercicio2-1 [kernel.kallsyms] [k] lock_page_memcg
19,61% ejercicio2-1 [kernel.kallsyms] [k] __slab_free
1,42% ejercicio2-1 [kernel.kallsyms] [k] tlb_gather_mmu
0,09% perf-exec [kernel.kallsyms] [k] arch_perf_update_userpage
0,01% perf-exec [kernel.kallsyms] [k] perf_event_update_userpage
0,00% perf-exec [kernel.kallsyms] [k] native_write_msr
Cannot load tips.txt file, please install perf!



Ejercicio 2.2

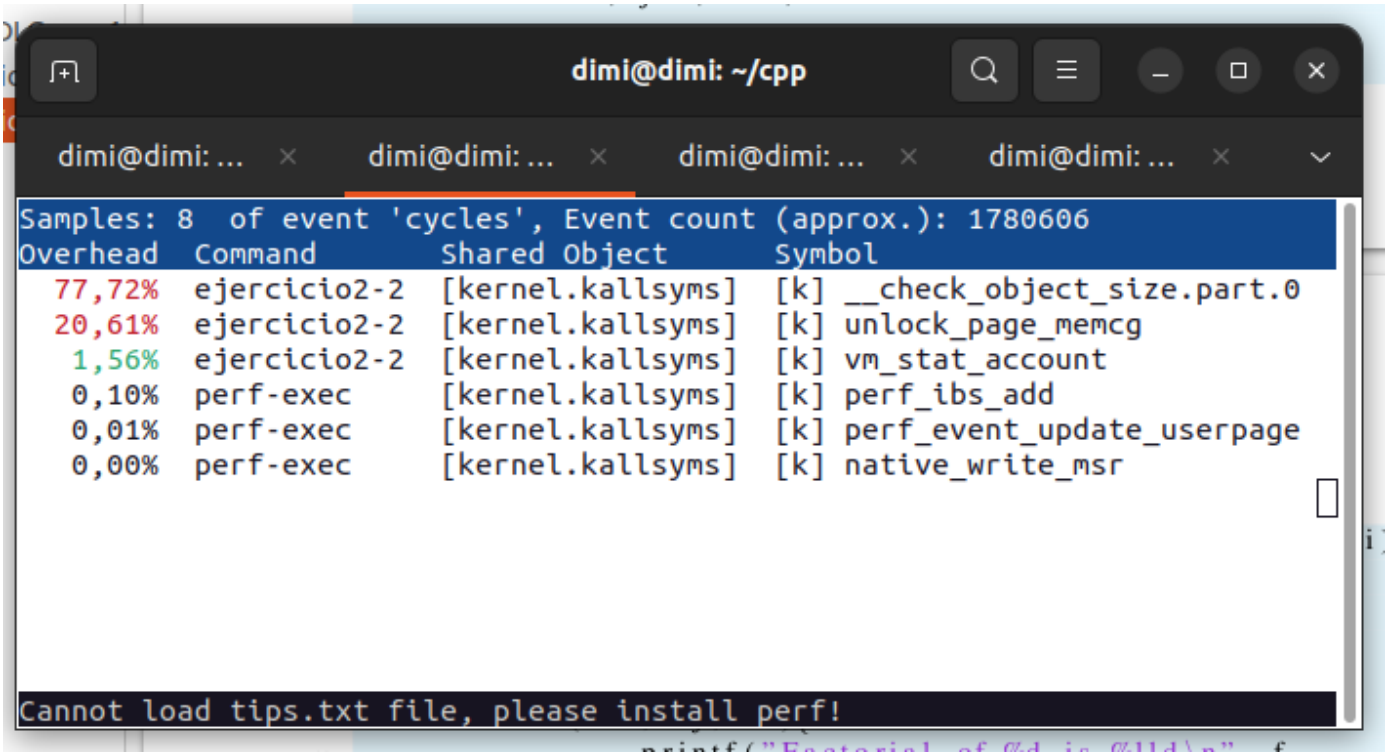
¿Qué hace el siguiente segmento de código?

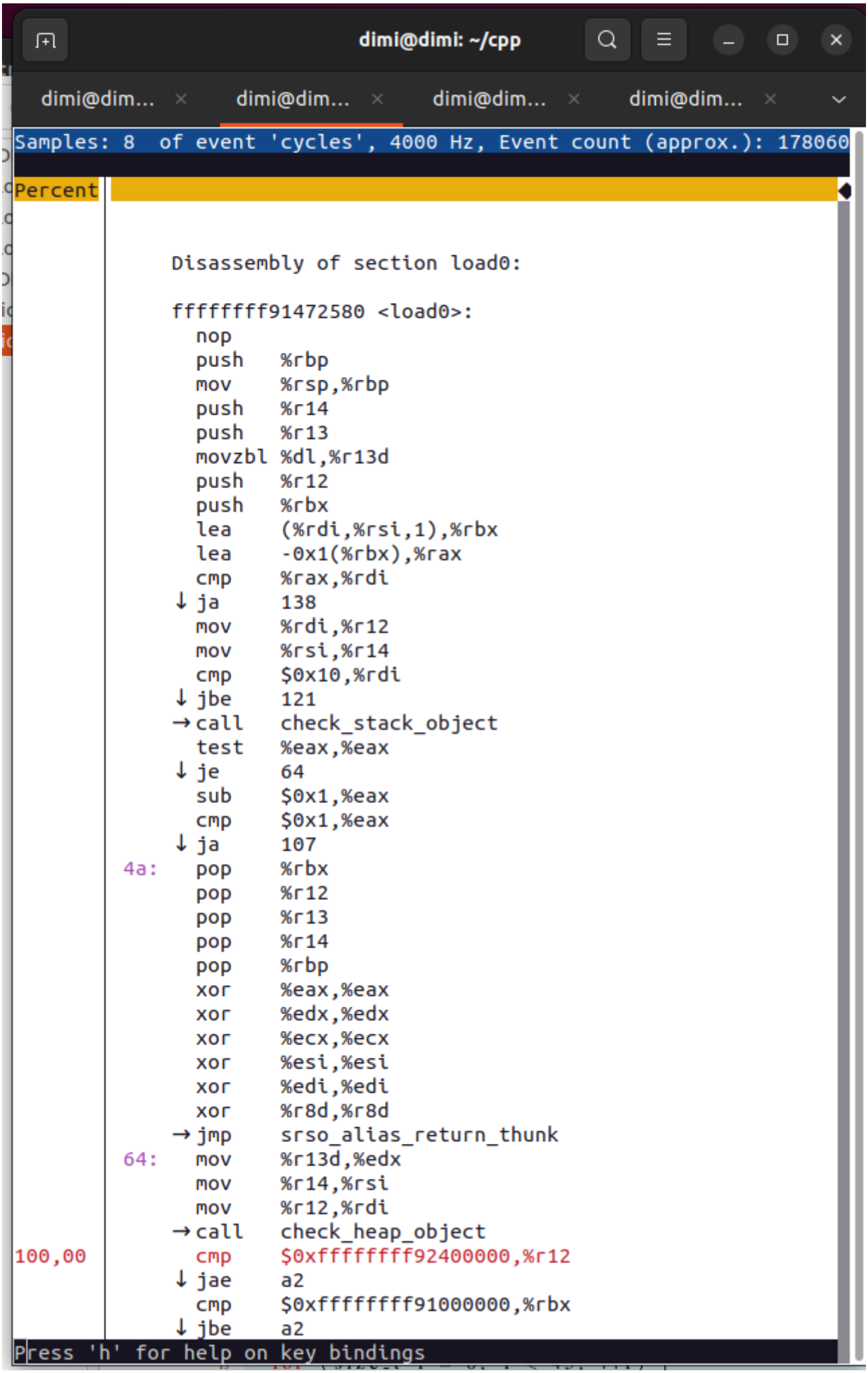
MEM05-C. Avoid large stack allocations
Recursive functions must ensure that they do not exhaust the stack as a result of excessive recursions.

```
unsigned long fib1(unsigned int n) {
  if (n == 0) {
    return 0;
```

```
  }
  else if (n == 1 || n == 2) {
    return 1;
  }
  else {
    return fib1(n-1) + fib1(n-2);
  }
}
```

• El programa permite mostrar el código desensamblado de la aplicación, adjunta alguna captura.




dimi@dimi: ~/cpp
dimi@dim...
dimi@dim...
dimi@dim...
dimi@dim...
Samples: 8 of event 'cycles', 4000 Hz, Event count (approx.): 178060
Percent
Disassembly of section load0:
ffffffff91472580 <load0>:
nop
push %rbp
mov %rsp,%rbp
push %r14
push %r13
movzbl %dl,%r13d
push %r12
push %rbx
lea (%rdi,%rsi,1),%rbx
lea -0x1(%rbx),%rax
cmp %rax,%rdi
↓ ja 138
mov %rdi,%r12
mov %rsi,%r14
cmp $0x10,%rdi
↓ jbe 121
→ call check_stack_object
test %eax,%eax
↓ je 64
sub $0x1,%eax
cmp $0x1,%eax
↓ ja 107
4a: pop %rbx
pop %r12
pop %r13
pop %r14
pop %rbp
xor %eax,%eax
xor %edx,%edx
xor %ecx,%ecx
xor %esi,%esi
xor %edi,%edi
xor %r8d,%r8d
→ jmp srso_alias_return_thunk
64: mov %r13d,%edx
mov %r14,%rsi
mov %r12,%rdi
→ call check_heap_object
100,00 cmp $0xffffffff92400000,%r12
↓ jae a2
cmp $0xffffffff91000000,%rbx
↓ jbe a2
Press 'h' for help on key bindings

==========
• ¿Podrías decir cual es la instrucción que más tiempo de CPU requiere? Adjunta
uns captura y describe la razón.
###Factorial

```
(base) dimi@dimi:~/cpp$ time ./ejercicio2-1
Factorial of 12 is 479001600

real    0m0,001s
user    0m0,000s
sys     0m0,001s
```

###Fobonacci

```
(base) dimi@dimi:~/cpp$ time ./ejercicio2-2
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
121393
196418
```

```
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733

real    0m0,001s
user    0m0,001s
sys     0m0,000s
(base) dimi@dimi:~/cpp$
```