Detección de errores en uso de memoria

La memoria que se obtiene dinámicamente (mediante `new`) debe ser liberada (con `delete`). Esto debe hacerse de manera sistemática: por cada instrucción que solicita memoria debe haber alguna o algunas correspondientes que la liberen, tal vez en otra función. No obstante, además de esta buena práctica de programación, se debe controlar que el programa hace un correcto manejo de le memoria. Una vez que se sabe que el programa cumple la especificación funcional se debe correr alguna herramienta de análisis. Un ejemplo de ello es valgrind.

El siguiente es el resultado de la ejecución de un programa que deja sin liberar 1 bloque de 8 bytes y otro de 184 bytes:

```
$ valgrind ./principal < test/esVaciaPila.in
==5305== Memcheck, a memory error detector
==5305== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5305== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5305== Command: ./principal
==5305==
1>La pila está vacia.
2>La pila está vacia.
3>Fin.
==5305==
==5305== HEAP SUMMARY:
==5305==     in use at exit: 192 bytes in 2 blocks
==5305==   total heap usage: 9 allocs, 7 frees, 78,400 bytes allocated
==5305==
==5305== LEAK SUMMARY:
==5305==    definitely lost: 8 bytes in 1 blocks
==5305==    indirectly lost: 184 bytes in 1 blocks
==5305==      possibly lost: 0 bytes in 0 blocks
==5305==    still reachable: 0 bytes in 0 blocks
==5305==         suppressed: 0 bytes in 0 blocks
==5305== Rerun with --leak-check=full to see details of leaked memory
==5305==
==5305== For counts of detected and suppressed errors, rerun with: -v
==5305== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Al comando de ejecución se le puede agregar la opción `--leak-check=full` para obtener más informacion:

```
$ valgrind --leak-check=full ./principal < test/esVaciaPila.in
==5324== Memcheck, a memory error detector
==5324== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5324== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5324== Command: ./principal
==5324==
1>La pila está vacia.
2>La pila está vacia.
3>Fin.
==5324==
==5324== HEAP SUMMARY:
==5324==     in use at exit: 192 bytes in 2 blocks
==5324==   total heap usage: 9 allocs, 7 frees, 78,400 bytes allocated
==5324==
==5324== 192 (8 direct, 184 indirect) bytes in 1 blocks are definitely lost in loss record 2 of 2
==5324==    at 0x4C3217F: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload_memcheck-amd64
```

```
==5324==    by 0x1092BD: crearPila() (pila.cpp:34)
==5324==    by 0x108A1B: main (principal.cpp:48)
==5324==
==5324== LEAK SUMMARY:
==5324==    definitely lost: 8 bytes in 1 blocks
==5324==    indirectly lost: 184 bytes in 1 blocks
==5324==      possibly lost: 0 bytes in 0 blocks
==5324==    still reachable: 0 bytes in 0 blocks
==5324==         suppressed: 0 bytes in 0 blocks
==5324==
==5324== For counts of detected and suppressed errors, rerun with: -v
==5324== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Se muestra que en la línea 48 de `principal.cpp` se llama a `crearPila`, de `pila.cpp` que en la línea 34 usa el operador `new` para obtener memoria. Ese segmento de memoria obtenido no fue liberado.

Luego de agregar la función que debe devolver ese bloque, `liberarPila(p);`, en la línea 157 de `principal.cpp` se puede ver que no hay errores ni memoria perdida.

```
$ valgrind --leak-check=full ./principal < test/esVaciaPila.in
==5387== Memcheck, a memory error detector
==5387== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5387== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5387== Command: ./principal
==5387==
1>La pila está vacia.
2>La pila está vacia.
3>Fin.
==5387==
==5387== HEAP SUMMARY:
==5387==     in use at exit: 0 bytes in 0 blocks
==5387==   total heap usage: 9 allocs, 9 frees, 78,400 bytes allocated
==5387==
==5387== All heap blocks were freed -- no leaks are possible
==5387==
==5387== For counts of detected and suppressed errors, rerun with: -v
==5387== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Por más información acerca de los mensajes de error ver Mensajes de error.