Nokia PARO Laboratorium 3

Mikołaj Szymczyk

Marzec 2022

Spis treści

1	$\mathbf{W}\mathbf{step}$	2
2	Zadanie 1	2
3	Zadanie 2	2
4	Zadanie 3	3
5	Zadanie 4	3
6	Zadanie 5	3
7	Zadanie 6	4

1 Wstęp

Celem laboratorium było zapoznanie się z:

- zarządzaniem pamięcią aplikacji napisanych w języku C++,
- narzędziem Valgrid,
- smart pointerami m.in. shared_ptr oraz unique_ptr
- zastosowaniem wyjątków

2 Zadanie 1

Raport narzędzia Valgrid (rys. 1) wskazał na nie zwolnienie pamięci wskaźnika num. Rozwiązaniem tego problemu było zastosowanie operatora delete na zmiennej num.

```
Leak_DefinitelyLost 1 warning

✓ ﷺ example1.cpp 1 warning

✓ 4 bytes in 1 blocks are definitely lost in loss record 1 of 1 1 warning

© 0x4838E63 operator new(unsigned long)

© 0x1091FE main example1.cpp:5
```

Rysunek 1: Raport narzędzia Valgrid dla zadania 1

3 Zadanie 2

Raport narzędzia Valgrid (rys. 2) wskazał na nie zwolnienie pamięci wektora wskaźników num. Rozwiązaniem tego problemu było zastosowanie zaimplementowanej funkcji deallocateInts z parametrem num.

```
Leak_DefinitelyLost 1 warning

Leak_DefinitelyLost 1 warning

40 bytes in 10 blocks are definitely lost in loss record 1 of 1 1 warning

0x4838E63 operator new(unsigned long)

0x10B30F allocateInts() example2.cpp:9

0x10B47B main example2.cpp:25
```

Rysunek 2: Raport narzędzia Valgrid dla zadania 2

4 Zadanie 3

Raport narzędzia Valgrid (rys. 3) wskazał na nie zwalnianie pamięci wskaźnika rsc. Rozwiązaniem tego problemu było przeniesienie wywołania destruktora poza klamry try catch.

```
Leak_DefinitelyLost 1 warning

Leak_DefinitelyLost 1 warning

1 bytes in 1 blocks are definitely lost in loss record 1 of 1 1 warning

0x483BE63 operator new(unsigned long)

0x10938E main example3.cpp:35
```

Rysunek 3: Raport narzędzia Valgrid dla zadania 3

5 Zadanie 4

Raport narzędzia Valgrid (rys. 4) wskazał na nie zwalnianie pamięci w klasie *My-Pointer*. Problemem okazało się pole klasy, które było wskaźnikiem. Rozwiązaniem tego problemu było umieszczenie funkcji *foo* w klamrach try oraz umieszczenie w klamrach catch operatora delete dla pola klasy

```
Leak_DefinitelyLost 1 warning

Leak_DefinitelyLost 1 warning

4 example4.cpp 1 warning

4 bytes in 1 blocks are definitely lost in loss record 1 of 1 1 warning

6 0x4838E63 operator new(unsigned long)

6 0x109515 MyPointer::MyPointer() example4.cpp:13
```

Rysunek 4: Raport narzędzia Valgrid dla zadania 4

6 Zadanie 5

Raporty narzędzia Valgrid (rys. 5 i6), po małej modyfikacji kodu, wskazały na nie zwalnianie pamięci wskaźników pFirst oraz pSecond. Rozwiązaniem tego problemu było zastosowanie unique_ptr z biblioteki memory.

```
ole  

Valgrind

Valgrind

Valaction Leak_DefinitelyLost 1 warning

Value example5.cpp 1 warning

Value 1 bytes in 1 blocks are definitely lost in loss record 1 of 1 1 warning

Conduction Control of the control of t
```

Rysunek 5: Raport narzędzia Valgrid dla zadania 5



Rysunek 6: Raport narzędzia Valgrid dla zadania 5

7 Zadanie 6

Po zaimplementowaniu ciała funkcji makeFile oraz metody addToFile z wykorzystaniem shared_ptr z biblioteki memory. Raporty narzędzia Valgrid nie wykazywał wycieków pamięci