Daniel Glazer 252743 Grupa: Czwartek 17:05

Praktyczne Aspekty Rozwoju Oprogramowania

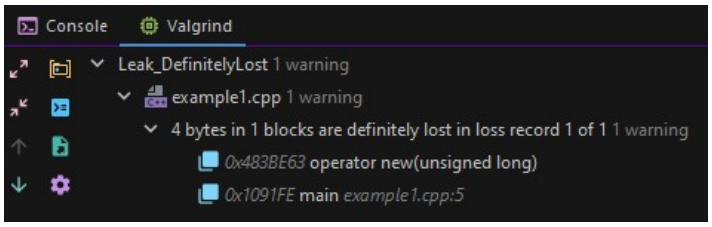
Zarządzanie pamięcią

1 Wstęp

Zadania zostały umieszczone na githubie `https://github.com/danielglazer26/paro_nokia` . Wykonano je w środowisku CLion, przy użyciu WSL do uruchomienia Valgrinda.

2 Zadanie 1

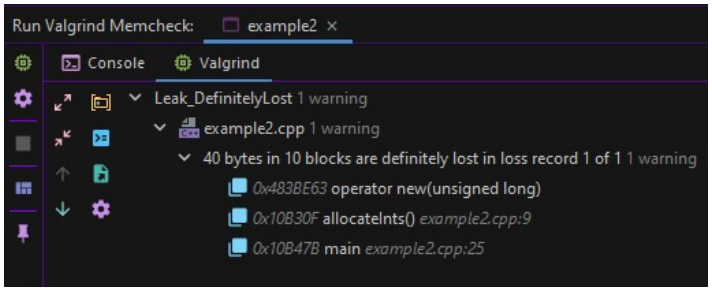
W zadaniu pierwszym Valgrind pokazywał, że tracone są 4 bajty ponieważ nie jest usuwany wskaźnik tworzony przez operator *new*. Aby zaradzić temu problemowi należało dodać operator *delete*, który usuwa wskaźnik na zmienną *num*.



Rysunek 1: Wyciek pamięci sygnalizowany przez Valgrinda w zadaniu 1

3 Zadanie 2

W zadaniu drugim Valgrind pokazywał, że traconych jest 40 bajtów na operatorze *new*, aby poradzić sobie z tym wyciekiem danych należało wywołać przed końcem programu metodę *deallocateInts(num)*, która usuwała zarezerwowany wskaźnik na wektor.



Rysunek 2: Wyciek pamięci sygnalizowany przez Valgrinda w zadaniu 2

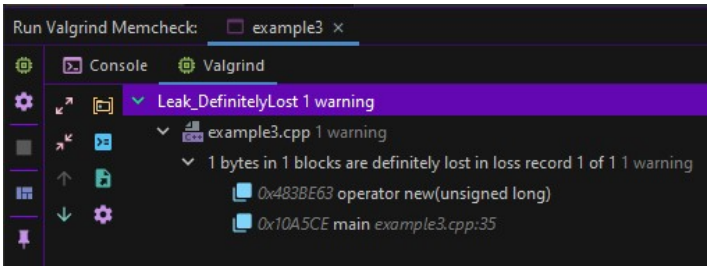
4 Zadanie 3

W zadaniu trzecim należało napisać swój własny wyjątek. Zrobiono więc klasę *CustomLogicError*, która dziedziczy po *std::logic_ error* wraz z konstruktorem w taki o to sposób:

```
class CustomLogicError : public std::logic_error {
public:
    explicit CustomLogicError(const std::string &arg) : logic_error(arg) {
    }
};
```

Rysunek 3: Klasa dziedzicząca po *std::logic_ error*

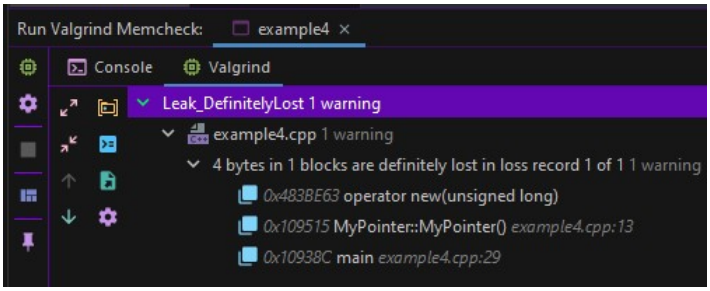
Następnie naprawiono wyciek pamięci, który polegał na tym, że w momencie, w którym podano jako parametr znak "d", został wyrzucony wyjątek w bloku *try catch*. Powodowało to, że wskaźnik na klasę *Resource* nie był usuwany, więc żeby temu zaradzić, usuwanie operatorem *delete* zostało przeniesione za blok.



Rysunek 4: Wyciek pamięci sygnalizowany przez Valgrinda w zadaniu 3

5 Zadanie 4

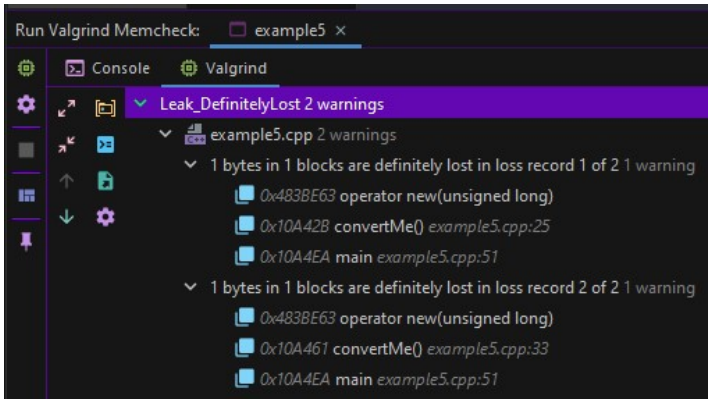
W zadaniu czwartym wyrzucany wyjątek powodował, że nie był wywoływany destruktor klasy *MyPointer*. W tym celu utworzono blok *try catch* wewnątrz konstruktora klasy *MyPointer*. Po wprowadzeniu tam bloku, destruktor spełniał swoje zadanie i usuwał za alokowaną pamięć.



Rysunek 5: Wyciek pamięci sygnalizowany przez Valgrinda w zadaniu 4

6 Zadanie 5

W zadaniu piątym przy kilku pierwszych uruchomieniach Valgrind nie pokazywał żadnych wycieków pamięci. Dopiero po którymś razie wskazał na wycieki w dwóch blokach danych. Było to spowodowane tym, że z pewnym prawdopodobieństwem pojawiały się wyjątki, które doprowadzały do zakończenia się programu w niekontrolowanym miejscu. Aby poradzić sobie z tym problemem zamieniono zwykłe wskaźniki na *unique_ptr* i usunięto wszystkie operatory *delete*.



Rysunek 6: Występujący po kilku próbach wyciek pamięci w zadaniu 5

7 Zadanie 6

W ostatnim zadaniu należało zdefiniować ciała funkcji *makeFile* i *addToFile*. Dla funkcji *makeFile* zdefiniowano *shared_ptr* dla strumienia pliku i co ważne należało też zdefiniować *deleter* obsługujący operacje zamknięcia strumienia pliku. Jeśli tego się nie zrobiło w momencie, w którym usuwany był współdzielony wskaźnik, strumień do pliku nie był zamykany. W przypadku funkcji *addToFile* dodano funkcję *fprintf*, do której przekazano wskaźnik na strumień oraz ciąg znaków. Po uruchomieniu programu Valgrind nie wskazywał na żadne wycieki pamięci.