PARO 2022

Sprawozdanie - memory management

Hubert Droździak

Spis treści

1 Zadania

1.1 example1

W zadaniu tym alokowana pamięć nie była zwalniana, co powodowało za każdym uruchomieniem programu wyciekiem pamięci. Zadanie zostało rozwiązane poprzez zwolnienie pamięci operatorem **delete**.

1.2 example2

Zadanie to było bardzo podobne do poprzedniego. Pamięć była alokowana, ale nie była zwalniana. Rozwiązanie zadania polegało na użyciu funkcji **deallocateInts()** i zwolnieniu zajmowanej pamięci.

1.3 example3

W tym zadaniu z pomocą narzędzia valgrind zdiagnozowano wyciek pamięci. Dochodziło do niego, gdy użytkownik podawał jako argument funkcji znak '**d**'. Wyciek był spowodowany przez użycie ryzykownej funkcji wyrzucającej wyjątek. Zadanie rozwiązano zwalniając w bloku **catch** alokowaną wcześniej pamięć.

Jest to rozwiązanie wystarczające, aby program działał, lecz nie jest ono idealne. Lepszym sposobem byłoby pozbycie się operatorów **new** oraz **delete** z bloku **try** lub poprzez użycie inteligentnego wskaźnika.

Dodatkowo w ramach tego zadania stworzono klasę własnego wyjątku dziedziczącego po klasie **std::logic_error**.

1.4 example4

W zadaniu następował wyciek pamięci, spowodowany wywoływaniem w konstruktorze funkcji wyrzucającej wyjątek.

Zadanie zostało rozwiązane dzięki użyciu inteligentnych wskaźników, dzięki czemu uniknięto wycieku pamięci. W przypadku wystąpienia wyjątku inteligentny wskaźnik zadba o uwolnienie zablokowanej pamięci.

1.5 example5

W tym zadaniu przy spełnieniu się warunku w funkcji **process**() lub **processSecond()** następowało wyrzucenie wyjątku, przez co alokowana pamięć funkcją **convertMe()** nie została nigdy zwalniana, w skutek czego następował wyciek pamięci.

Zadanie rozwiązano z pomocą inteligentnych wskaźników, dzięki którym nie potrzeba samemu zwalniać pamięci, a zrobi to sam wskaźnik.

1.6 example6

W zadaniu, w funkcji **makeFile** zwraca współdzielony wskaźnik na plik. Następnie za pomocą funkcji **addToFile** są do pliku dodawane kolejne linie, dzięki otworzeniu do niego strumienia.

2 Wnioski

Nieprawidłowe używanie wskaźników może prowadzić wycieku pamięci, a w gorszym przypadku nawet do zaprzestania działania programu.

Najprostszym typem, a zarazem trudnym w użytkowaniu, jest zwykły wskaźnik. Ze względu na liczne wyjątki, może często dochodzić do wycieków pamięci.

Następcą zwykłego wskaźnika jest unikalny wskaźnik. Zapewnia on bezpieczne korzystanie z pamięci, ponieważ to nie na użytkowniku spoczywa uwolnienie pamięci.

Dodatkowym, przydatnym typem wskaźnika jest współdzielony wskaźnik. Zapewnia on odpowiednie działanie, kiedy istnieje potrzeba współdzielenia pamięci. Sam zaalokuje odpowiednią pamięć do obiektu oraz kiedy liczba referencji do zasobu spadnie do zera, zostaje on zwolniony.

3 Kod

Wykonane zadania zostały udostępnione na platformie GitHub pod linkiem: https://github.com/Warzkos/memory_management