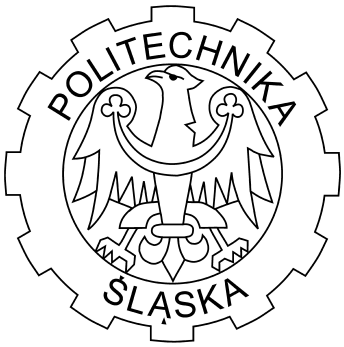
Politechnika Śląska
Wydział Automatyki Elektroniki i Informatyki

Implementacja kontenerów z biblioteki STL

Wojciech Janota

3 listopada 2020



1 Opis projektu

Głównym założeniem projetku było stworzenie odpowiedników kontenerów z biblioteki Standard Template Library. Założyłem, że zaimplementuję następujące kontenery:

- vector
- queue
- priority-queue
- stack
- list
- set
- binary tree

2 Opis budowy klas

2.1 Container

Zgodnie ze złożonymi wcześniej wstępnymi szkicami projektu główna klasa bazowa, `container`, odpowiada za zarządzanie pamięcią i podstawowe operacje na wewnętrznej tablicy dynamicznej. Po niej dziedziczą klasy `vector, queue, priority-queue, stack, set`. Jest w niej zaimplementowany konstruktor i destruktor, które są wspólne dla wszystkich kontenerów po niej dziedziczących, podstawowe "getery"i śetery"dla wewnętrznych zmiennych, operator[], podstawowe funkcje operujące na wewnętrznej tablicy dynamicznej (dodawanie, usuwanie elementów) oraz funkcja sortująca, jako metoda chroniona. Klasa jest abstrakcyjna, aby uniemożliwić stworzenie obiektu na jej podstawie.

2.2 Vector

`Vector` to kontener dziedziczący po `container` będący bardziej zaawansowaną tablicą dynamiczną. Zawiera w sobie mechanizmy obsługi pamięci, inteligentnie zarządza swoją wielkością. Dodawanie elementów na początek tego kontenera lub w środku jest bardzo kosztowne czasowo, ponieważ wymagane jest przesunięcie wszystkich elementów od indeksu na którym wstawiany jest nowy element o jedną pozycję w prawo. Jest za to bardzo łatwy w obsłudze (jak tablica dynamiczna), w miarę łatwy w implementacji i w większości zastosowań dość szybki.

2.3 Queue

`Queue`, czyli kolejka to kontener FIFO (First In First Out). Umożliwia dodawanie elementów na koniec, pobranie i wyrzucenie pierwszego elementu i podstawowe metody takie jak zwrócenie wielkości kontenera.

2.4 Priority queue

Jest to kolejka priorytetowa, na podstawie podanego priorytetu (MIN, MAX) na początku kolejki zawsze znajduje się największy lub najmniejszy element. Reszta działania jest identyczna jak w przypadku zwykłej kolejki.

2.5 Stack

`Stack`, czyli stos to kontener FILO (First In Last Out). Możliwe jest dodanie elementu na górę stosu i zdjęcie elementu ze stosu.

2.6 Set

`Set` to kontener samosortujący, poprzez wstawienie elementu w odpowiednie miejsce. Zaimplementowałem odpowiednik STL-owego multiseta sortującego tylko rosnąco, zwracającego także indeks pierwszego elementu nie mniejszego i nie większego od podanego.

2.7 List

Lista jest odpowiednikiem vectora, używającym listy jako wewnętrzego sposobu przechowywania danych. Umozliwia to dodawanie elementu na początku i na końcu w czasie stałym, jednak powoduje również, że operacje pobierania elementu z konkretnego indeksu są wykonywane w czasie liniowym.

2.8 Binary tree

Drzewo binarne jest strukturą sortująca, przechowującą kolejne elementy jako kolejne węzły w drzewie binarnym. Moja implementacja umożliwia wstawianie do drzewa, wypisanie drzewa w kolejności in-order, usuwanie poszczególnych węzłów oraz pobranie prawego i lewego syna podanego elementu.

3 Szczegółowy opis klas, zależności i metod

Szczegółowa dokumentacja została wygenerowana przez narzędzie doxygen.

4 Problemy

W związku z bardzo ograniczonym czasem na napisanie projektu, nie udało się zaimplementować mapy, a binary tree mogłoby zostać jeszcze udoskonalone. Używanie szablonów momentami bywa nieintuicyjne, jednak udało się zaimplementować powyższe kontenery w ich użyciem z czego jestem zadowolony.

5 Testowanie

Do testowania wycieków pamięci użyłem narzędzia valgrind. Projekt okazał się od nich wolny. Przetestowane zostały także wszystkie metody poszczególnych klas. Kod został sformatowany z użyciem clang-format w standardzie LLVM.

6 Wnioski

Po napisaniu w miarę funkcjonalnych odpowiedników kontenerów z STL mogę stwierdzić, że znacząco ułatwiają one pracę z użyciem języka C++. Pisząc je należy pamiętać o pewnych niuansach, które pominięte mogą okazać się katasrofalne, jak np. niepoprawne poszerzanie i zwężanie vectora może spowodować błąd SIGSEGV, ponieważ nagle program odwoła się do niepoprawnej pamięci. Muszę również stwierdzić, że pisanie projektu w tak krótkim czasie jest o wiele trudniejsze i ciężej jest rozłożyć efektywnie pracę. Rozumiem, że wynika to z obecnej sytuacji, jednak chcę zwrócić uwagę na problem, jaki tak krótki czas może sprawić.

7 Dokuemntacja wygenerowana przez doxygen