Laboratorium Informatyki II
Ćwiczenie 13

Standardowa biblioteka szablonów STL

Biblioteka STL (**S**tandard **T**emplate **L**ibrary) – *Standardowa Biblioteka Szablonów* - została opracowana na wewnętrzne potrzeby HP Lab w roku 1994. Jest to biblioteka, w której zaimplementowano bardzo wydajne algorytmy, podstawowe struktury danych, kontenery oraz iteratory. Wszystkie te elementy utworzone są za pomocą szablonów w języku C++. Dzięki takiej konstrukcji możemy używać powyższych elementów wykorzystując dowolny typ danych wbudowany lub zdefiniowany w kodzie programu. Biblioteka STL dostarcza narzędzi dla programowania uogólnionego.

Biblioteka STL dostarcza trzy typ elementów:

Kontenery

Homogeniczne jednostki (klasy) o określonym sposobie przechowywania i dostępu do danych wybranego typu. Różnią się wewnętrzną organizacją danych, sposobem ich alokacji w pamięci, dostępu oraz wydajności operacji. Biblioteka STL dostarcza kilkanaście typów kontenerów:

- Lista `<list>` - dwukierunkowa lista elementów przechowujących wskaźniki na następny i poprzedzający element listy
- Tablica `<vector>` - jest tablicą dynamiczną wraz z narzędziami wspomagającymi manipulowanie zawartością i rozmiarem
- Tablica jednowymiarowa `<array>` - odpowiednik statycznej tablicy o stałej liczbie elementów, dostarczający narzędzi kontroli dostępu (pilnuje czy zlecone odwołanie jest możliwe do realizacji - indeks odwołania jest w zakresie rozmiaru tablicy.
- Stos `<stack>`- struktura danych, która działa na zasadzie **LIFO** (Last-In, First-Out), co oznacza, że ostatni element dodany do stosu jest pierwszym, który zostanie odczytany (usunięty).
- Kolejka `<queue>` - struktura danych, która działa na zasadzie **FIFO** (First-In, First-Out), co oznacza, że pierwszy element dodany do kolejki jest pierwszy, który zostanie odczytany (usunięty).
- Mapa poszukiwań `<map>` - struktura danych, która przechowuje je w parach klucz-wartość, gdzie klucze są unikalne i uporządkowane, a wartości są z nimi powiązane. Klucz i dane mogą być wartościami dowolnego typu (także złożonymi), w przypadku klucza typ musi posiadać zdefiniowany operator "`<`".

- Drzewo poszukiwań (zbiór) `<set>` - uporządkowana struktura danych pozwalająca na przechowywanie unikalnych elementów (elementy dublujące są kasowane)
- Tablica podwójnie kończona `<deque>` - jest struktura danych będącą połączeniem stosu i kolejki, możliwy jest dostęp do jej początku i końca dla dodawania, odczytywania i kasowanie
- Tablica bitowa `<bitset>` - struktura danych umożliwia zdefiniowanie słowa bitowego o żądanej długości i wykonywania operacji na nim
- Wielokrotne drzewo poszukiwań `<multiset>` - struktura danych będąca modyfikacją struktury `set` pozwalającej na przechowywanie wielu pozycji o danej wartości.
- Wielokrotna mapa poszukiwań `<multimap>` - jest strukturą danych będącą modyfikacją struktury `map` w której do klucza można przypisać więcej jak jedną wartość

Iteratory

Iterator w C++ to obiekt, który wskazuje na element w kontenerze lub sekwencji elementów. Działa jako mechanizm, który umożliwia iterację po elementach kontenera, odczytanie lub zmianę wartości tych elementów. Jest odpowiednikiem wskaźnika, który wskazuje na elementy kontenera. Istnieje wiele rodzajów iteratorów w C++, które różnią się swoimi właściwościami i możliwościami. Podstawowe typy iteratorów to:

- *Input Iterator*: Pozwala na odczytanie wartości z kontenera i poruszanie się do przodu. Może być używany w operacjach jednokierunkowej iteracji, takich jak odczytywanie wartości z kontenera za pomocą pętli `while` lub `for`.
- *Output Iterator*: Pozwala na zapis wartości do kontenera i poruszanie się do przodu. Może być używany do dodawania nowych elementów do kontenera.
- *Forward Iterator*: Kombinuje funkcje *Input Iterator* i *Output Iterator*, umożliwiając iterację do przodu, odczytanie i zapis wartości.
- *Bidirectional Iterator*: Rozszerza funkcjonalność *Forward Iteratora*, umożliwiając również iterację wsteczną, czyli poruszanie się do tyłu po elementach kontenera.
- *Random Access Iterator*: Zapewnia pełną funkcjonalność do poruszania się po elementach kontenera w dowolnej kolejności. Daje możliwość skokowego dostępu do elementów za pomocą operatora indeksowania [ ] oraz wykonywania operacji arytmetycznych na iteratorach, takich jak inkrementacja, dekrementacja, dodawanie i odejmowanie.

Każdy z kontenerów ma przypisany typ iteratora którego wybór dokonywany jest automatycznie.

Algorytmy

Algorytmy są funkcjami dostarczonymi wraz z biblioteką **STL** umożliwiającymi wykonywanie szeregu operacji na kontenerach. Korzystanie z nich jest możliwe po podpięciu biblioteki `<algorithm>`. Niektóre funkcje algorytmów zostały umieszczone w wydzielonych bibliotekach tematycznych `<numeric>` i `<functional>`. Kilka najpopularniejszych zestawiono poniżej:

- `std::sort`: Sortuje elementy w określonym zakresie. Może być stosowany do posortowania wektorów, list, tablic i innych kontenerów posiadających iterator.

- `std::find`: Wyszukuje pierwsze wystąpienie określonej wartości w określonym zakresie.
- `std::reverse`: Odwraca kolejność elementów w określonym zakresie.
- `std::copy`: Kopiuje elementy z jednego zakresu do innego.
- `std::transform`: Wykonuje operację transformacji na elementach z określonego zakresu i zapisuje wynik do innego zakresu.
- `std::accumulate`: Oblicza sumę wartości w określonym zakresie.
- `std::count`: Liczy ilość wystąpień określonej wartości w określonym zakresie.
- `std::remove`: Usuwa wszystkie wystąpienia określonej wartości z określonego zakresu.
- `std::unique`: Usuwa duplikaty z posortowanego zakresu.
- `std::find_if`: Wyszukuje pierwsze wystąpienie elementu spełniającego określony warunek.
- `std::binary_search`: Sprawdza, czy określona wartość znajduje się w posortowanym zakresie.
- `std::min_element` i `std::max_element`: Znajdują najmniejszy i największy element w określonym zakresie.

Zadania

Zaprojektować klasę z wykorzystaniem biblioteki STL pozwalającej na przechowywanie dowolnej liczby zestawów liczb losowych z podanego zakresu. Program pozwala na:

- utworzenie pustego obiektu
- utworzenie obiektu zawierającego jeden zestaw liczb o zadanych parametrach
- określonej liczby zestawów liczb o określonych parametrach
- metody wyświetlenia
  - określonego zestawu liczb,
  - wskazanych zestawów liczb,
  - nieokreślonego przedziału zestawów
  - wszystkich zestawów
- wyszukanie określonej liczby we obiekcie i zwrócenie informacji o liczbie jej wystąpień i numerach zestawów w których występuje
- dodanie zestawów (określona liczba zestawów)
- zwiększenie liczby wartości w wskazanym zestawie
- wyświetlenie zawartości obiektu
- wyświetlenie zawartości wskazanego zestawu
- porównanie zestawów w obiekcie, porównanie obiektów