Instytut Informatyki i Matematyki Komputerowej UJ

Programowanie 2

Zadanie H – Szablonowe BST

Punktów do uzyskania: **10**

1. Generalia

- Zadanie polega na zaprogramowaniu szablonu klasy implementującej binarne drzewo przeszukiwań, noszącej nazwę `BST`.
- Każdy węzeł drzewa musi obejmować zawartość danych dwóch typów określonych w konkretyzacji szablonu:
  - Typ klucza (nazywany odtąd `KEY_TYPE`), odpowiedzialny za identyfikację i uporządkowanie zasadniczych danych.
  - Typ właściwej merytorycznej danej (nazywany odtąd `DATA_TYPE`).
- **Ważne założenie: w strukturze drzewa wartości typu `KEY_TYPE` nigdy się nie powtarzają.**

2. Warunki konkretyzacji

- Skonkretyzowane użycie szablonu zakłada postać:

```
BST < KEY_TYPE, DATA_TYPE >
```

- Stosowanym konkretyzacjami będą wyłącznie konkretyzacje typami statycznymi.
- W przypadku konkretyzacji typami bez zaimplementowanego domyślnie strumieniowego wyjścia należy założyć dostępność jego globalnego przeładowania.
- W przypadku konkretyzacji typu `KEY_TYPE` typem bez implementacji operatora mniejszości należy założyć dostępność jego globalnego przeładowania.
- **Powyżej opisane globalne przeładowania operatorów wejścia oraz mniejszości stanowią najobszerniejszy zbiór warunków koniecznych do użycia implementowanej klasy `BST`. Tym samym założenie czegokolwiek więcej jest bezpodstawne.**
- Jedynym dopuszczalnym konstruktorem obiektów klasy `BST` jest konstruktor bezparametrowy tworzący puste drzewo.
- Destruktor klasy `BST` musi poprawnie zwracać wszelką zaalokowaną w trakcie działania pamięć.

3. Wymagane metody publiczne

- `Insert`
  - Wywoływana wyłącznie z dwoma argumentami. Pierwszym w postaci wartości typu `KEY_TYPE` i drugim w postaci wartości typu `DATA_TYPE`.
  - Powoduje wstawienie do drzewa węzła o kluczu wartości pierwszego argumentu i zawartości merytorycznej drugiego argumentu.
  - W przypadku poprawnego wstawienia węzła zwraca wartość **`true`**, zaś w przypadku próby dodania węzła z istniejącym kluczem zwraca wartość **`false`**.
- `Search`
  - Wywoływana z argumentem wartości typu `KEY_TYPE`.
  - Zwraca adres wartości typu `DATA_TYPE` w przypadku znalezienie węzła o podanym kluczu lub wartość `NULL` w przypadku braku węzła o zadanym kluczu.
- `Delete`
  - Wywoływana z argumentem wartości typu `KEY_TYPE`.
  - Kasuje węzeł o kluczu danym argumentem.
  - W przypadku skasowania zwraca wartość **`true`**, zaś w przypadku braku w drzewie węzła o zadanym kluczu zwraca wartość **`false`**.
  - **Spośród dwóch możliwości zastąpienia kasowanego węzła węzłem o kluczu bezpośrednio poprzedzającym lub węzłem o kluczu bezpośrednio następującym, ZAWSZE wybiera węzeł o kluczu bezpośrednio następującym.**
- Metody `PreOrder`, `InOrder`, `PostOrder` oraz `LevelOrder`.
  - Są bezargumentowymi procedurami.
  - ***Wypisują węzły*** drzewa w porządkach zadanych nazwą (trzy pierwsze) lub poziomami (ostatnia).
  - ***Wypisanie węzła*** oznacza wypisanie na strumieniowe wyjście najpierw wartości typu `KEY_TYPE` z następującym bezpośrednio wypisaniem wartości typu `DATA_TYPE`. Zakładamy, że operatory wyjścia dla obu użytych typów są zewnętrznie zdefiniowane.
- `Height`
  - Bezargumentowa funkcja typu **`int`** zwracająca aktualną wysokość drzewa.
  - Za wysokość drzewa przyjmujemy maksymalną możliwą ilość krawędzi od korzenia do liścia.
  - Dla drzewa pustego funkcja zwraca wartość `-1`.

4. Uwarunkowania implementacyjne

- Treść rozwiązania musi być zawarta w pliku o nazwie `BST.h` i spakowana programem ZIP.
- Plik o nazwie `BST.h` jest włączany dla testowania rozwiązań.
- Należy założyć zewnętrzne włączenie pliku nagłówkowego `iostream` oraz użycie przestrzeni nazw `std`.
- Żadne własne włączenia plików nagłówkowych nie są dopuszczalne.
- Użyte algorytmy nie muszą być optymalne czasowo i pamięciowo, ale złożoności nie mogą przekraczać kwadratowych.