Zadanie projektowe nr 1

Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych.

Należy zaimplementować oraz dokonać pomiaru czasu działania operacji takich jak dodawanie elementu, usunięcie elementu i wyszukanie elementu w następujących strukturach danych:

a) Tablica,
b) Lista dwukierunkowa,
c) Kopiec binarny (typu maksimum – element maksymalny w korzeniu) ,
d) Drzewo przeszukiwań binarnych BST (Binary Search Tree),
e) Drzewo czerwono-czarne (należy wcześniej zapoznać się z drzewem BST) .

Należy przyjąć następujące założenia:

- podstawowym elementem struktur jest 4 bajtowa liczba całkowita ze znakiem,
- wszystkie struktury danych powinny być alokowane dynamicznie (w przypadku tablic powinny zajmować jak najmniej miejsca – powinny być relokowane przy dodawaniu/usuwaniu elementów),
  - tablica, która służy do przechowywania kopca nie będzie relokowana po każdej operacji – przyjąć jakąś stałą wartość nadmiarową, aby można było wykonać na niej operacje dodawania,
- w przypadku tablicy i listy rozpatrzeć osobno operacje dodawania i usuwania elementu na następujących pozycjach:
  a. początek tablicy/listy,
  b. koniec tablicy/listy,
  c. Losowe miejsce tablicy/listy.
- w przypadku drzewa BST implementujemy algorytm równoważenia drzewa (algorytm DSW), który wywołujemy po automatycznie w przypadku usuwania lub dodania elementu (czas tych operacji mierzymy razem z operacją równoważenia),
- należy pomierzyć czasy wykonywania poszczególnych operacji w funkcji rozmiaru danej struktury (ilości elementów w niej przechowywanych). W przypadku zbyt krótkich czasów można albo zwiększyć ilość danych pomiarowych, albo powtórzyć dany pomiar np. 10 razy. Ponieważ wyniki zależą także od rozkładu danych, to pomiary dla konkretnego rozmiaru struktury należy wykonać wielokrotnie (np. 100 razy – za każdym razem generując nową populację) a wynik uśrednić. Ilość przechowywanych elementów należy dobrać

eksperymentalnie (np. może to być 1000, 2000, 5000, 10000, 20000) w zależności od wydajności sprzętu. Nie włączać do czasu pomiaru czasu generacji danych. Ilość różnych punktów pomiarowych (rozmiarów problemu) musi wynosić co najmniej 5. Podczas pomiaru czasu wykonywania wyłączyć zbędne aplikacje.

- należy mieć na uwadze, że czas wykonywania operacji może zależeć od wartości przechowywanych elementów, co należy uwzględnić w pomiarach i wnioskach,
- dodatkową funkcją programu musi być możliwość sprawdzenia poprawności zaimplementowanych operacji i zbudowanej struktury (szerzej w na ten temat w dalszej części dokumentu),
- do dokładnego pomiaru czasu w systemie Windows w c++ można skorzystać z funkcji *QueryPerformanceCounter* lub `std::chrono::high_resolution_clock` (opis na stronie http://cpp0x.pl/forum/temat/?id=21331)
- dopuszczalnymi językami programowanie są języki kompilowane do kodu natywnego (np. C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Phyton); dopuszczalne jest odstępstwo od tej reguły za zgodą prowadzącego
- używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- nie wolno korzystać z gotowych bibliotek np. STL, Boost lub innych – wszystkie algorytmy i struktury muszą być zaimplementowane przez studenta (nie kopiować gotowych rozwiązań),
- realizacja zadania powinna być wykonana w formie jednego programu,
- kod źródłowy powinien być komentowany,
- program musi skompilowany do wersji exe (i w takiej wersji zostanie poddany testom).

Sprawdzenie poprawności zbudowanej struktury/operacji obejmuje:

- utworzenie struktury z liczb zapisanych w pliku tekstowym .Każda liczba będzie w osobnej linii, natomiast pierwsza liczba określa ilość zapisanych liczb w pliku. Nazwa pliku nie może być na sztywno wpisana do programu – należy o nią zapytać przy wczytywaniu danych z pliku. Przy tworzeniu nowej struktury z pliku poprzednie dane muszą zostać usunięte. W przypadku tablicy i listy kolejność danych musi być taka sama jak w pliku. W przypadku drzew tworzymy je poprzez kolejne dodawanie elementów z pliku w kolejności jak są zapisane w pliku,
- wyświetlenie tej struktury na ekranie (w przypadku drzew wymyśleć jakąś przejrzystą formę – kopiec też jest drzewem – można korzystać z gotowych rozwiązań),
- możliwość wykonania operacji na strukturze z tym, że w przypadku:

a. Usuwania z tablicy - zostanie podana pozycja liczby(indeks), którą należy usunąć. Indeksujemy od zera,
b. wstawianie do tablicy - zostanie podana pozycja i wartość, którą należy wstawić na podana pozycję (tablice w razie konieczności rozsunąć),
c. Usuwania z listy - zostanie podana tylko wartość , którą należy usunąć ,
d. Dodawania do listy - podajemy wartość i numer pozycji (indeks) na który mamy wstawić wartość,
e. Usuwania (dodawania) dla drzew – zostanie podana liczba (klucz), którą należy usunąć (dodać). Z kopca usuwamy dowolny element poprzez podanie klucza.
f. Wyszukiwania (dla wszystkich struktur) – zostanie podana liczba, którą należy znaleźć – należy tylko wyświetlić, czy liczba jest w strukturze, czy nie.

Opisane operacje najlepiej zrealizować w formie menu dla każdej struktury, gdzie będzie możliwość wykonania tych operacji np.

1.Zbuduj z pliku (elementy powinny się pojawić w tablicy i liście w takiej kolejności jak w pliku, natomiast w przypadku drzew budowanie ma polegać na dodawaniu kolejnego wczytanego elementu do drzewa. Opcja „Zbuduj z pliku" powinna usunąć poprzednie dane)

2.Usuń

3.Dodaj

4.Znajdź

5. Utwórz losowo (losowe wygenerowanie struktury) – należy spytać o wielkość struktury.

6.Wyświetl (tablicę, listę, kopiec, drzewo). Tablicę oraz listę wyświetlamy w jednej linii (elementy rozdzielamy spacją). W przypadku listy lista jest wyświetlana dwukrotnie - od przodu w jednej linii i od tyłu w drugiej. Do wyświetlania drzew można skorzystać z gotowych rozwiązań.

7.Równoważenie drzewa (tylko dla drzewa BST – operacja: zbuduj z pliku, dodaj, usuń powinna być rozdzielona od operacji równoważenia drzewa dla celów sprawdzania poprawności programu) .

UWAGA ! Po każdej operacji należy z automatu wyświetlić strukturę

Sprawozdanie (w formie papierowej) powinno zawierać:

- krótki wstęp w którym zostaną przedstawione złożoności obliczeniowe operacji w implementowanych strukturach na podstawie literatury,
- plan eksperymentu czyli założenia co do wielkości struktur, sposobu generowania elementów tych struktur, sposobie pomiaru czasu, itp.

- zestawienie wyników w formie tabelarycznej i graficznej (czas wykonania operacji w funkcji ilości elementów – nie używać wykresu słupkowego) - i wyniki osobno dla poszczególnych operacji (zwracać uwagę na czytelność wykresów),
- wnioski dotyczące efektywności poszczególnych struktur w zależności od zastosowań, wielkości struktury itp., wskazać (jeśli są) przyczyny rozbieżności pomiędzy uzyskanymi eksperymentalnie złożonościami, a teoretycznymi.
- załączony kod źródłowy w formie elektronicznej (skopiować cały projekt oraz wersję skompilowaną programu)
- jednostki w sprawozdaniu powinny być mianowane i mieć zachowaną odpowiednią precyzję

Maksymalna ocena za zadanie:

3.0 – eksperymenty na tablicy, liście i kopcu binarnym

3.5 - eksperymenty na tablicy, liście i kopcu binarnym (program w wersji obiektowej)

4.0 - eksperymenty na tablicy, liście i kopcu binarnym i drzewie BST bez równoważenia drzewa (program w wersji obiektowej)

4.5 - eksperymenty na tablicy, liście i kopcu binarnym i drzewie BST z równoważenia drzewa algorytmem DSW (program w wersji obiektowej)

5.0 – eksperymenty na tablicy, liście i kopcu binarnym i drzewie czerwono- czarnym (program w wersji obiektowej)

5,5 - eksperymenty na tablicy, liście i kopcu binarnym i drzewie czerwono- czarnym, drzewie AVL (program w wersji obiektowej)

Tzw. „wysypywanie się” programu skutkuje obniżeniem oceny oraz koniecznością jego poprawienia.

Na stronie http://jaroslaw.mierzwa.staff.iiar.pwr.edu.pl/sdizo/menu.cpp podano przykładowe menu, które można wykorzystać przy pisaniu programu.