# 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, usuwanie i wyszukiwanie elementu w następujących strukturach danych:

a) Tablica dynamiczna,
b) Lista dwukierunkowa,
c) Kopiec binarny,
d) Drzewo poszukiwań binarnych (BST),
e) Drzewo czerwono-czarne.

**Należy przyjąć następujące założenia:**

- elementem wszystkich struktur jest 4 bajtowa liczba całkowita ze znakiem,
- wszystkie struktury danych powinny być alokowane dynamicznie i zajmować jak najmniej miejsca (relokacja przy dodawaniu/usuwaniu elementów). Kopiec powinien być zaimplementowany w wariancie z tablicą a nie jako drzewo ze wskaźnikami,
- dla tablicy i listy rozpatrzyć osobno operacje dodawania i usuwania elementu na następujących pozycjach:
    a. początek,
    b. koniec,
    c. inne wybrane miejsce.
- dla kopca wystarczy usuwanie elementu ze szczytu, elementy dodawane powinny oczywiście być automatycznie umieszczane we właściwych miejscach tak, aby zachowany był warunek kopca,
- należy zbadać zależność czasu wykonywania poszczególnych operacji od rozmiaru danej struktury (liczby przechowywanych elementów). Ponieważ wyniki niektórych operacji zależą od wartości elementów, pomiary dla konkretnego rozmiaru należy wykonać wielokrotnie (przynajmniej 10 razy lub więcej, za każdym razem generując nową losową populację), wynik powinien być uśredniony. Pomiary powinny być wykonane dla przynajmniej 5 reprezentatywnych rozmiarów (lepiej więcej) tak, aby odzwierciedlały one typ zależności (liniowy, logarytmiczny). Maksymalny rozmiar dobrać eksperymentalnie zależnie od możliwości sprzętu tak, aby nie przekroczyć pojemności pamięci a czas pomiarów był akceptowalny. Można przyjąć orientacyjnie, że dla prostej struktury typu tablica powinny być to rozmiary rzędu milionów,
- ważną funkcją programu powinna być możliwość sprawdzenia poprawności zbudowanej struktury i zaimplementowanych operacji dla przykładowych danych o niewielkim rozmiarze. Będzie to potrzebne przy oddawaniu projektu (szerzej na ten temat w dalszej części). Warto podkreślić, że takie sprawdzenie jest konieczne również przed wykonaniem testów czasowych (przy nieprawidłowo działających operacjach będą one bezwartościowe),

- sposoby dokładnego pomiaru czasu w systemie Windows podano na stronie www:

  http://staff.iiar.pwr.wroc.pl/antoni.sterna/sdizo/SDiZO_time.pdf

- dopuszczalnymi językami programowania są języki kompilowane do kodu natywnego (na przykład C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Phyton),
- używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- nie wolno korzystać z gotowych bibliotek takich jak STL, Boost lub innych przy implementacji głównych algorytmów. Można ich użyć tylko w operacjach pomocniczych takich jak przygotowanie (losowanie) lub wczytanie danych testowych, wyświetlanie kopca lub drzewa. Wszystkie algorytmy i struktury muszą być zaimplementowane samodzielnie (nie wolno kopiować gotowych rozwiązań),
- implementacja powinna być wykonana w postaci jednego programu,
- kod źródłowy powinien być odpowiednio komentowany,
- program musi być skompilowany do wersji EXE (w takiej wersji zostanie poddany testom),
- warto pamiętać o dużych różnicach w wynikach testów czasowych pomiędzy wersjami *Debug* i *Release* (testy trzeba koniecznie przeprowadzić w wersji *Release*).

**Sprawdzenie poprawności zbudowanej struktury/operacji:**

- utworzenie struktury z liczb zapisanych w pliku tekstowym w ustalonym formacie. Pierwsza liczba w pliku określa rozmiar struktury danych, kolejne są wartościami elementów. Liczby w pliku będą rozdzielone białymi znakami (spacja, tabulacja, przejście do nowej linii). Nie powinna mieć znaczenia liczba i rodzaj białych znaków (na przykład spacje mogą być pojedyncze lub wielokrotne, liczby mogą być dowolnie grupowane w liniach). Można skorzystać ze wskazówek podanych na stronie www:

  http://staff.iiar.pwr.wroc.pl/antoni.sterna/sdizo/SDiZO_file.pdf

- wyświetlenie struktury na ekranie. Można przyjąć, że w testach liczba elementów tablicy i listy będzie niewielka co pozwoli na wyświetlenie ich w jednej linii. Listę trzeba wyświetlić na dwa sposoby: od początku i od końca, aby zweryfikować poprawność operacji dodawania/usuwania elementów. Dla kopca i drzewa zaproponować możliwie przejrzystą formę prezentacji,
- możliwość wykonania elementarnych operacji na strukturze (dodawanie, usuwanie, wyszukiwanie). W każdym przypadku powinna być możliwość wprowadzenia odpowiednich danych (na przykład pozycja tablicy i wartość do wstawienia na tej pozycji). Opisane operacje najlepiej zrealizować w formie menu dla każdej struktury,
- zwrócić uwagę na efektywność wykonywania testów. Na przykład po wykonaniu operacji zmieniającej dane w strukturze trzeba automatycznie wyświetlić jej nową zawartość,
- program powinien być odporny, przynajmniej w podstawowym zakresie, na podanie niewłaściwych danych (na przykład ujemnego indeksu tablicy).

**Sprawozdanie powinno zawierać:**

- krótki wstęp, w którym powinny być przedstawione złożoności obliczeniowe operacji w implementowanych strukturach na podstawie literatury,
- plan eksperymentu, czyli założenia co do wielkości struktur, sposobu generowania ich elementów, sposobu pomiaru czasu,
- zestawienie wyników w formie tabel i wykresów (czas operacji w funkcji liczby elementów), wyniki powinny być przedstawione osobno dla poszczególnych operacji. Trzeba pamiętać o właściwym doborze jednostek czasu (np. czasy poniżej sekundy podawać jako 1,2 ms a nie 0,0012 s). Liczba cyfr po przecinku powinna być "rozsądna", uwzględniająca nieuniknione błędy pomiarów (np. 12,3 ms a nie 12,3456789 ms),
- wnioski dotyczące efektywności poszczególnych struktur w zależności od zastosowań, wielkości struktury itp. Wskazać przyczyny rozbieżności (jeśli występują) pomiędzy złożonościami uzyskanymi eksperymentalnie a teoretycznymi,
- kod źródłowy powinien być przekazany w formie elektronicznej (pliki źródłowe i plik wykonywalny). W sprawozdaniu nie zamieszczamy pełnych wydruków kodu, można ewentualnie zilustrować omawiane operacje niewielkimi jego fragmentami.

**Ocena projektu:**

Warianty na ocenę powyżej 3.0 powinny być programami w wersji obiektowej. Możliwe są pewne odstępstwa od tej zasady jeśli "zwykły" kod będzie dobrej jakości.

3.0 – eksperymenty na tablicy, liście i kopcu binarnym (może być wersja nieobiektowa)

3.5 – eksperymenty na tablicy, liście i kopcu binarnym

4.0 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie BST (tylko podstawowe operacje: dodawanie, usuwanie i wyszukiwanie elementów, nie jest wymagane równoważenie drzewa)

4.5 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie BST (dodawanie, usuwanie i wyszukiwanie elementów, równoważenie drzewa na żądanie z wykorzystaniem algorytmu DSW, operacje rotacji w prawo i w lewo dla wskazanego węzła)

5.0 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie czerwono-czarnym

Dodatkowe, nieobowiązkowe opcje dla lubiących wyzwania:

- Eksperymenty na drzewie AVL
- Porównanie czasów operacji dla własnej implementacji tablicy i listy z czasami dla wariantu z kontenerami z biblioteki STL (odpowiednio *vector* i *list*)