Rok akademicki 2019/20 **Informatyka - S1 - semestr 3**

Algorytmy 2

Laboratorium 3 – Lista z przeskokami

Celem zadania jest implementacja słownika w postaci listy z przeskokami.
Implementacja nie może odwoływać się do typu tablicowego.
Każdy z węzłów listy posiada następujące składowe:

- pole składowej kluczowej typu **int**;
- pole typu **double**;
- pole typu **char**;
- tablica wskaźników do następnych węzłów (jeśli ze względu na logikę implementacji będzie to niezbędne przed wczytaniem wartości z pliku `inlab03.txt` to założyć, że maksymalna zadana wysokość węzłów nie przekroczy **10**).

UWAGA: Przyjąć „domyślne" indeksowanie tablicy wskaźników do następników, tzn. „numer" (indeks) najniższego poziomu niech wynosi **0**.
Węzły listy są uporządkowane rosnąco (posortowane) ze względu na wartość składowej kluczowej typu **int**.

Klucze (składowe kluczowe) są **unikalne** w ramach listy.
Należy stworzyć funkcję realizujące:

1) wstawienie do listy nowego elementu, funkcja ustawia pole typu **double** na losowe wartości, zaś pole typu **char** na wartość **'T'**; (uwzględnić przypadek, w którym węzeł o zadanej wartości składowej kluczowej już znajduje się w liście, należy wtedy zastąpić istniejącą wartość składowej typu **char** wartością **'D'**).
2) wstawienie do listy ***X*** nowych elementów o wygenerowanych losowo i różnych wartościach składowych kluczowych z zakresu 99 do 99999 (wartość ***X*** podana jako argument funkcji); funkcja ta powinna uwzględnić możliwość wylosowania „duplikatów" składowych kluczowych i odpowiednio korygować kontrolę liczby już wstawionych węzłów tak, by po zakończeniu jej wykonania liczba nowo wstawionych węzłów wynosiła ***X***;
3) wyszukanie w liście elementu o podanej jako argument wartości składowej kluczowej (wraz z obsługą przypadku, w którym element nie istnieje);
4) usunięcie z listy elementu o podanej jako argument wartości składowej kluczowej (uwzględniające obsługę przypadku, w którym taki element nie istnieje);
5) prezentacja wartości składowych kluczowych pierwszych ***Y*** węzłów znajdujących się w liście i osiągających co najmniej wskazaną jako argument tej funkcji wysokość ***N*** (wartości ***Y*** i ***N*** są argumentami funkcji prezentacji) ;
6) wypisywanie liczby węzłów znajdujących się w liście i „widocznych" na wskazanym poziomie (czyli węzłów o wysokości co najmniej równej wskazanej jako argument tej funkcji);
7) usuwanie wszystkich elementów listy.

Listę można zaimplementować strukturalnie (C) lub obiektowo (C++);
Na liście argumentów mogą się pojawić wyłącznie:

- referencja (wskaźnik) do pierwszego węzła (czoła) listy każdej z wymienionych funkcji *(tylko w przypadku implementacji strukturalnej);*
- wartość składowej kluczowej dla funkcji wstawiania, usuwania i wyszukiwania węzła o wskazanej wartości składowej kluczowej;
- liczba wstawianych węzłów dla funkcji wstawiania **X** elementów;
- numer poziomu dla funkcji prezentacji węzłów i funkcji wypisywania liczby węzłów.

Wszystkie funkcje powinny obsługiwać błędy (np. funkcja prezentacji powinna wypisywać odpowiedni komunikat, gdy lista jest pusta lub nie istnieje).

Program po uruchomieniu wczytuje plik wejściowy `inlab03.txt`**.**
Plik `inlab03.txt` zawiera w pierwszej linii kolejno liczbę elementów do wylosowania **X**, maksymalną wysokość **LMAX**, a następnie wartości pięciu kluczy **k1**, **k2**, **k3, k4, k5.** Należy przyjąć, że prawdopodobieństwo „obecności" węzła na kolejnym poziomie **PROB=0,5.**
Następnie wywoływana jest sekwencja funkcji (dalej w funkcji main()):

- czas start;
- zainicjowanie „pustej" listy (ale z głową i ogonem);
- wyszukanie klucza **k1**;
- wstawienie **X** elementów do listy;
- wypisanie liczby węzłów w liście (oprócz głowy i ogona) kolejno na poziomach od **0** do **LMAX-1**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" kolejno na poziomach od **0** do **LMAX-1**;
- wstaw element o wartości klucza **k2**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" na najniższym poziomie;
- wstaw element o wartości klucza **k3**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" na najniższym poziomie;
- wstaw element o wartości klucza **k4**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" na najniższym poziomie;
- wstaw element o wartości klucza **k5**;
- wypisanie liczby węzłów w liście (oprócz głowy i ogona) kolejno na poziomach od **0** do **LMAX-1**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" kolejno na poziomach od **0** do **LMAX-1**;
- usuń element o wartości klucza **k3**;
- usuń element o wartości klucza **k2**;
- usuń element o wartości klucza **k5**;
- wypisanie liczby węzłów w liście (oprócz głowy i ogona) kolejno na poziomach od **0** do **LMAX-1**;
- prezentacja wartości kluczowych pierwszych 20 węzłów (oprócz głowy i ogona) „widocznych" kolejno na poziomach od **0** do **LMAX-1**;
- czas stop;
- wypisz czas wykonania.

Przygotowanie e-maila do wysłania:

Uwaga! Kod źródłowy programu (1 plik) po oddaniu prowadzącemu zajęcia laboratoryjne musi zostać przesłany na adres `algo2@zut.edu.pl` :

- plik z kodem źródłowym musi mieć nazwę: `nr_albumu.algo2.lab03.main.c` (np. `123456.algo2.lab03.main.c`); jeśli kod źródłowy programu składa się z wielu plików, to należy stworzyć jeden plik, umieszczając w nim kody wszystkich plików składowych; (plik może mieć rozszerzenie `.c` lub `.cpp`)
- plik musi zostać wysłany z poczty ZUT (zut.edu.pl);
- nagłówek maila (temat) musi mieć postać: `ALGO2 IS1 XXXY LAB03`, gdzie `XXXY` to numer grupy (np. `ALGO2 IS1 210C LAB03`) ;

- w pierwszych trzech liniach pliku z kodem źródłowym w komentarzach muszą znaleźć się:
    - linia 1: informacja identyczna z zamieszczoną w nagłówku maila
    - linia 2: imię i nazwisko
    - linia 3: adres email
- email **nie** powinien zawierać żadnej treści (tylko załącznik).