15 czerwca 2018

Zadanie domowe – projekt 3

14 pkt.

Drzewa binarne to struktury, w których można w łatwy sposób uzyskać dostęp do danych, a także dodawać dane lub je usuwać. Takie drzewo składa się z węzłów, a każdy z nich może mieć co najwyżej dwoje dzieci, czyli węzły leżące na poziomie o jeden niższym – lewe i prawe dziecko. Każdemu węzłowi odpowiada wartość klucza (klucz może być zmienną dowolnego typu). Klucz lewego dziecka jest mniejszy od klucza węzła, a klucz prawego dziecka jest większy lub równy kluczowi węzła. Podstawę drzewa stanowi korzeń, czyli pierwszy węzeł, który nie ma rodzica.

1. Stwórz strukturę `struct Wezel` reprezentującą liść drzewa. Struktura powinna się składać z klucza typu `double` oraz wskaźnika na lewe dziecko i wskaźnika na prawe dziecko (będą to wskaźniki na tę strukturę). Zadeklaruj korzeń jako wskaźnik na strukturę. Nadaj wartości korzenia (wskaźnikom na dzieci nadaj wartości `NULL`).
2. Napisz funkcję `void wstaw(double k,struct Wezel **lisc)`, która wstawi podaną wartość w odpowiednim miejscu drzewa. Funkcja powinna sprawdzić, czy klucz liścia jest większy lub mniejszy niż wstawiana wartość i odpowiednio wstawić wartość jako lewego lub prawego syna. Jeśli dany węzeł ma już dwoje dzieci, to funkcja powinna rekurencyjnie wywołać samą siebie, aby zejść o poziom niżej: `wstaw(k,&(*lisc)->lewy);`. Jeśli wskaźnik na liść jest pusty, to funkcja powinna tworzyć nowy liść (czyli zaalokować dla niego pamięć, nadać mu wartość klucza oraz ustawić wskaźniki dzieci na `NULL`) i podpiąć go pod ostatnio sprawdzony węzeł (czyli ten, z którego nastąpiło zejście do niższego poziomu).
3. Napisz funkcję usuwającą drzewo `void usun_drzewo(struct Wezel *lisc)`, która będzie zwalniała pamięć zajmowaną przez drzewo. Wskazówka: sprawdzaj, czy wskaźnik na liść nie jest pusty. Jeśli nie jest, to wywołaj funkcję dla lewego i prawego syna, a następnie usuń liść.
4. Napisz funkcję `void wypisz_drzewo(struct Wezel *lisc,int p)` wypisującą kolejne elementy w drzewie (węzeł oraz jego dzieci w jednej linijce). Parametr `p` niech wskazuje na poziom, przy czym `p=0` dla korzenia. Jeśli któreś z dzieci nie ma nadanej wartości, to taka informacja powinna zostać również zamieszczona. Wypisywanie zaczynaj od lewego dziecka. Przykładowy wynik działania funkcji: drzewo.
5. Utwórz drzewo wypełnione 100. liczbami pseudolosowymi. Wypisz je na ekran.

Oceniane będą:

→ Poprawne zastosowanie poznanych procedur języka C (funkcji, tablic, wskaźników, itp.).

→ Poprawność działania kodu.

→ Czytelność kodu (stosowanie tabulacji dla bloków programu itp.).

→ Stosowanie komentarzy.

→ Optymalizacja kodu, czyli: stosowanie jak najmniejszej liczby zmiennych (w tym też operacje na wskaźnikach tam, gdzie będzie to optymalne) oraz wykonanie jak najmniejszej liczby operacji (pętli, watunków itp.), stosowanie operatorów „skrótowych", np. `++x` zamiast `x=x+1`, zminimalizowanie długości kodu (liczby linii oraz znaków – oczywiście z dokładnością do „elegancji" kodu i stosowania tabulacji itp.).

→ Poprawna implementacja drzewa i funkcji je obsługujących.

Termin: 28 czerwca godz. 23^{59}.