Podstawy Programowania Komputerów

Drzewo poszukiwań binarnych

6 grudnia 2018

W programie zadeklarowano następujące typy:

```
typedef int T;

/** węzeł drzewa poszukiwać binarnych */
struct wezel
{
   /** wartość przechowywana w węźle drzewa */
   T wartosc;
   /** wskaźnik na lewy potomek węzła */
   wezel * pLewy;
   /** wskaźnik na prawy potomek węzła */
   wezel * pPrawy;
};
```

Proszę zaimplementować następujące funkcje:

1. ```
   /** Funkcja dokonuje odbicia lustrzanego drzewa poszukiwać binarnych.
      Polega to na zmianie uporządkowania drzewa — wartości mniejsze są
      położone na prawo, a wartości większe na lewo.
   @param pRoot korzeń drzewa */
   void odbij (wezel * pRoot);
   ```

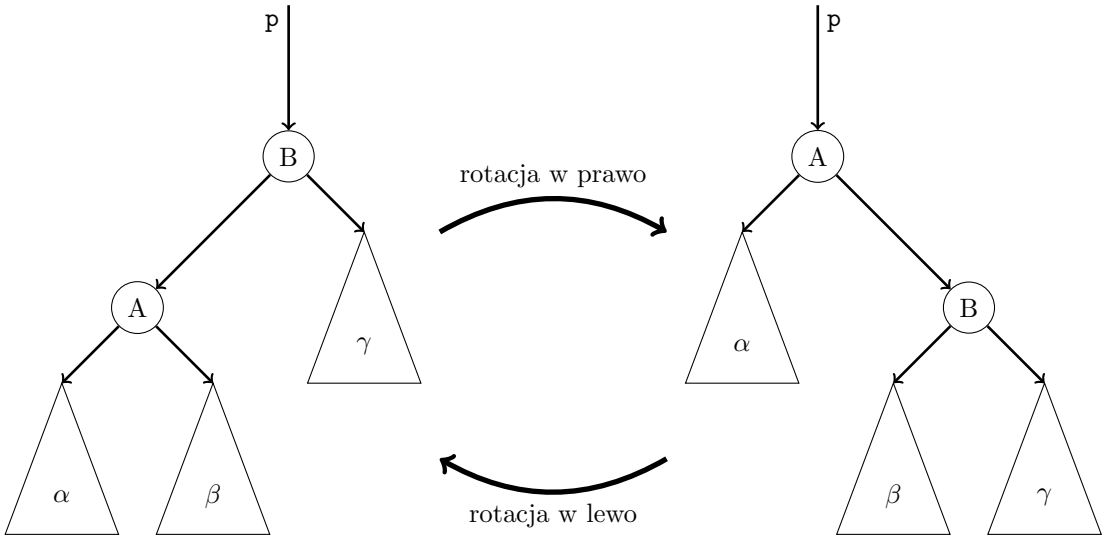
2. ```
   /** Funkcja wypisuje na standardowe wyjście wartości drzewa przy
      przechodzeniu wszerz. Jako kolejkę FIFO wykorzystana jest lista
      jedno- lub dwukierunkowa.
   @param pRoot korzeń drzewa
   */
   void wypiszWszerz (wezel * pRoot);
   ```

3. ```
   /** Funkcja wyszukuje nastepnik wezla. Następnikiem węzła jest kolejny
      węzeł pod względem wartości. Przykład: Jeżeli drzewo przechowuje
      wartości 1, 6, 9, 15, to następnikiem węzła przechowującego 6 jest
      węzeł przechowujący 9. Węzeł przechowujący 15 nie ma następnika.
   @param pRoot korzeń drzewa
   @param pPoprzednik wezel, ktorego nastepnik wyszukuje funkcja
   @return adres nastepnika (jeżeli istnieje)
    */
   wezel * znajdzNastepnik (wezel * pRoot, wezel * pPoprzednik);
   ```

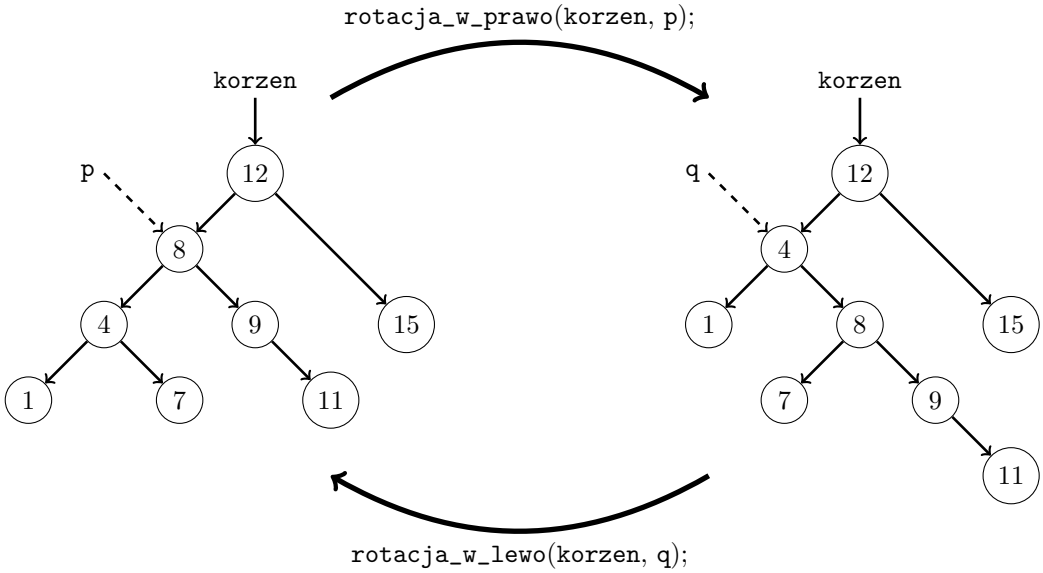
4. ```
   /** Funkcja usuwa węzeł z drzewa.
   @warning Należy rozpatrzyć cztery przypadki:
   (1) Węzeł do usunięcia nie ma potomków.
   ```

```
(2L) Węzeł do usunięcia ma tylko lewy potomek.
(2P) Węzeł do usunięcia ma tylko prawy potomek.
(3) Węzeł do usunięcia ma dwa przypadki. Wtedy należy znaleźć następnik
    węzła do usunięcia, przenieść wartość z następnika do węzła do
    usunięcia i usunąć następnik (który ma co najwyżej jeden potomek).
@param pRoot korzeń drzewa
@param pDoUsuniecia wezeł do usunięcia
*/
void usunWezel (wezel * & pRoot, wezel * pDoUsuniecia);
```

5. Proszę zaimplementować funkcje, które realizują operacje rotacji w drzewie binarnym.



Trójkąty (α, β, γ) symbolizują poddrzewa (które mogą być puste, tzn. równe **nullptr**).



(a)

```
/** Funkcja dokonuje rotacji w lewo w drzewie poszukiwań binarnych.
    Po operacji drzewo zachowuje własności drzewa poszukiwań
    binarnych.
```

```
@param pRoot korzeń drzewa
@param q wskaźnik na węzeł, względem którego dokonywana jest rotacja
    */
void rotacja_w_lewo (wezel * & pRoot, wezel * q);
```

(b)

```
/** Funkcja dokonuje rotacji w prawo w drzewie poszukiwań binarnych.
    Po operacji drzewo zachowuje własności drzewa poszukiwań
    binarnych.
@param pRoot korzeń drzewa
@param p wskaźnik na węzeł, względem którego dokonywana jest rotacja
    */
void rotacja_w_lewo (wezel * & pRoot, wezel * p);
```