Podstawy Programowania Komputerów [PPK-SSI] **29 stycznia 2018**

Egzamin

dyn
13:00

imię i nazwisko _____

liczba punktów _____ / 10

dokument klasy **B1**, archiwizować do **2020-01-01**

W programie zdefiniowano następujące typy:

```
struct Student
{
    string nazwisko;
    Student * pNext;
    double ocena;
};
```

```
struct Egzaminator
{
    string nazwisko;
    Egzaminator * pNext;
    Student * pStudenci;
};
```

Korzystając z nich można utworzyć strukturę danych, której przykład jest przedstawiony na rys. 1. Egzaminatorzy tworzą listę jednokierunkową. Studenci przypisani do egzaminatorów tworzą listy jednokierunkowe uporządkowane w porządku leksykograficznym według nazwisk studentów. Egzaminator może nie mieć przypisanego żadnego studenta. Lista egzaminatorów może być pusta. Zakładamy, że nazwiska studentów nie powtarzają się ani nazwiska egzaminatorów nie powtarzają się.

W programie zostały zdefiniowane następujące funkcje:

- **int** `policzStudentow` (**Egzaminator** * `pEgz`)

  która zwraca liczbę studentów egzaminatora o adresie `pEgz`. Jeżeli egzaminator nie istnieje (`pEgz` == **nullptr**), funkcja zwraca 0.

- **Egzaminator** * `znajdzEgzaminatora` (**Egzaminator** * `pHead`, **const string** & `nazwisko`);

  która zwraca adres egzaminatora o zadanym nazwisku w liście o głowie `pHead`. Jeżeli lista jest pusta, funkcja zwraca **nullptr**. Jeżeli nie ma w liście szukanego egzaminatora, funkcja zwraca **nullptr**.

Zadanie

Uwaga: Niedopuszczalne jest modyfikowanie definicji struktur **Student** i **Egzaminator**.

1. (3 pkt) Proszę zdefiniować funkcję

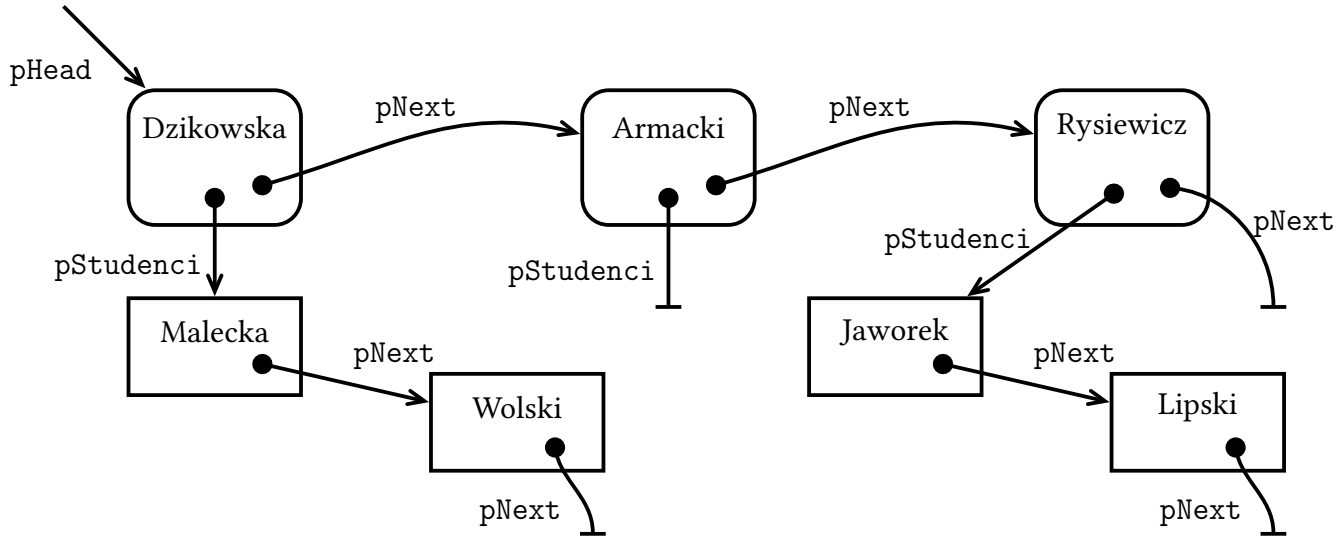
   Egzaminator * `znajdzNajmniejObciazonego` (**Egzaminator** * `pHead`);

   która zwraca adres egzaminatora z najmniejszą liczbą przypisanych studentów. Jeżeli lista egzaminatorów jest pusta, funkcja zwraca **nullptr** (lub 0). Jeżeli kilku egzaminatorów ma tę samą minimalną liczbę studentów, funkcja zwraca adres któregokolwiek egzaminatora.

2. (1 pkt) Proszę zdefiniować rekurencyjną funkcję

   void `dodajStudenta` (**Student** * & `pGlowa`, **const string** & `nazwisko`)

   która dodaje studenta o zadanym nazwisku do listy studentów o głowie `pGlowa`. Lista studentów jest posortowana wg nazwisk studentów. Po dodaniu kolejnego studenta lista nadal jest posortowana. Lista może być pusta.



Rysunek 1: Przykładowa struktura danych przechowująca egzaminatorów i przypisanych im studentów.

3. (1 pkt) Proszę zdefiniować funkcję

   void `dodajStudentaEgzaminatorowi` (**Egzaminator** * `pHead`, **const string** & `nazwisko`);

   która dodaje studenta o nazwisku `nazwisko` i ocenie `0.0` najmniej obciążonemu egzaminatorowi z listy egzaminatorów o głowie `pHead`. Jeżeli lista egzaminatorów jest pusta, funkcja nie robi nic. Funkcja wykorzystuje zdefiniowane wyżej funkcje.

4. (3 pkt) Proszę zdefiniować funkcję

   Student * `wytnijStudenta` (**Student** * & `pGlowa`, **const string** & `nazwisko`);

   która wycina (ale nie niszczy!) z listy studentów o głowie `pGlowa` studenta o zadanym nazwisku. Funkcja zwraca adres wyciętego studenta.

5. (2 pkt) Proszę zdefiniować funkcję

   void `wpiszOcene` (**Student** * & `pPrzeegzaminowani`, **Egzaminator** * `pEgzaminatorzy`, **const string** & `egzaminator`, **const string** & `student`, **double** `ocena`);

   która wykonuje następujące czynności (funkcja korzysta z wyżej zdefiniowanych funkcji):

   - wyszukuje egzaminatora o nazwisku `egzaminator`,
   - wycina studenta o nazwisku `student` z listy egzaminatora znalezionego w poprzednim punkcie,
   - wpisuje wyciętemu studentowi ocenę i wreszcie
   - przenosi wyciętego studenta do listy o głowie `pPrzeegzaminowani` – nie jest istotnie miejsce wstawienia studenta.

   Jeżeli nie ma egzaminatora o podanym nazwisku lub nie ma studenta o podanym nazwisku, funkcja nie robi nic.

Powodzenia ☺