

Politechnika Śląska  
Wydział Informatyki, Elektroniki i Informatyki

# **PODSTAWY PROGRAMOWANIA KOMPUTERÓW**

Dziekanat

---

autor	Kamil Sikora
prowadzący	Dr. inż. Adam Gudyś
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	poniedziałek, 12:00 – 13:30
sekcja	16
termin oddania sprawozdania	2019-18-01

---

# 1 Treść zadania

Napisać program, który na podstawie protokołów ocen, przekazanych do programu przez przełącznik „-i” wygeneruje dla każdego studenta plik zawierający uzyskane oceny, posortowane według nazwy przedmiotu. Nazwą pliku wyjściowego jest numer albumu studenta. Program uruchamiany jest z linii poleceń z wykorzystaniem następującego przełącznika:

-i nazwy plików z protokołami ocen, rozdzielone spacją

# 2 Analiza zadania

Zadanie polega na wstawianiu odpowiednich elementów do struktury danych, a następnie wygenerowaniu tylu plików, ilu studentów udało się do struktury wczytać. Pliki wyjściowe mają wyglądać jak na przykładzie:

Jan Jaworek

nr albumu: 14332

dr hab. inż. Mikołaj Kopernik Mechanika nieba 4.5 2012-01-14

dr inż. Jan Heweliusz Wstęp do sztucznej inteligencji 3.5 2012-02-29

## 2.1 Struktury danych

W programie wykorzystano drzewo binarne do przechowywania danych studentów. Drzewo binarne przechowuje dane w węzłach. Węzeł może mieć od 0 do 2 potomków, przy czym po lewej stronie węzła znajdują się potomki przechowujące wartości nie większe niż węzeł rodzicielski, po prawej zaś większe.

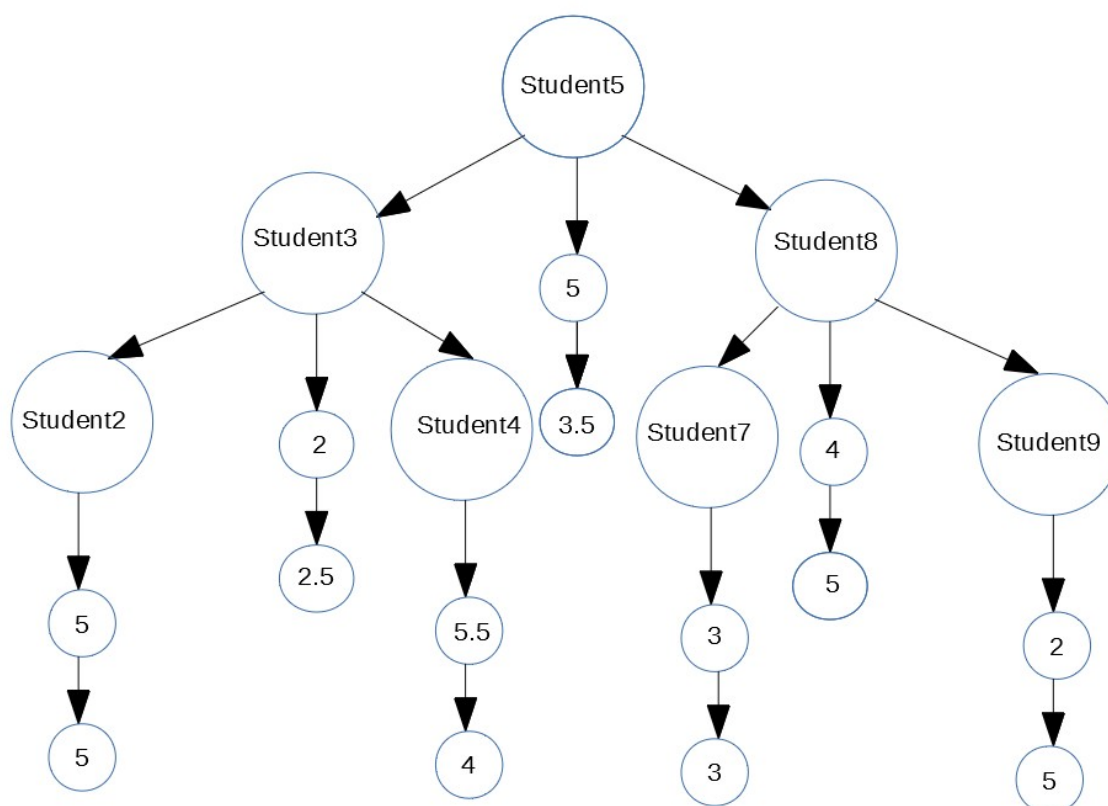
Każdy element drzewa binarnego posiada wskaźnik na listę jednokierunkową ocen danego studenta. Każda lista jest sortowana według nazw przedmiotów.

Rysunek 1 przedstawia zaimplementowaną strukturę danych.

# 3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików wejściowych rozdzielonych białym znakiem, poprzedzone przełącznikiem „-i”, np.

Dzikanat -i protokol1.txt protokol2.txt protokol3.txt



Rysunek 1: Struktura zaimplementowana w programie.  
Każdy student(element drzewa binarnego) posiada listę jednokierunkową, przechowującą dane dotyczące otrzymanych ocen.

Uruchomienie programu bez podawania żadnych parametrów, bądź z parametrami innymi, niż podane wyżej skutkuje niewyświetleniem żadnej informacji na ekranie. W przypadku podania nazwy pliku, która nie jest prawidłowa, wyświetlana jest stosowna informacja na ekranie. W przypadku, gdy plik wejściowy ma nieprawidłowe formatowanie, również jest wyświetlana stosowna informacja.

## 4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalny. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji.

## 4.1 Ogólna struktura programu

W funkcji głównej sprawdzane jest, czy został użyty poprawny przełącznik. Jeśli tak, wywoływana jest funkcja **wczytaj**, odczytująca dane z pliku wejściowego, następnie funkcja ta wywołuje kolejne funkcje zajmujące się wstawianiem podanych wartości do struktury danych. Po wczytaniu danych do struktury, w funkcji głównej wywoływana jest funkcja **zapisz**, która z kolei wywołuje funkcje generującą pliki wyjściowe. Po wykonaniu tych operacji wywoływana jest funkcja **usuń**, która kasuje wszystkie zaalokowane dynamicznie elementy struktury.

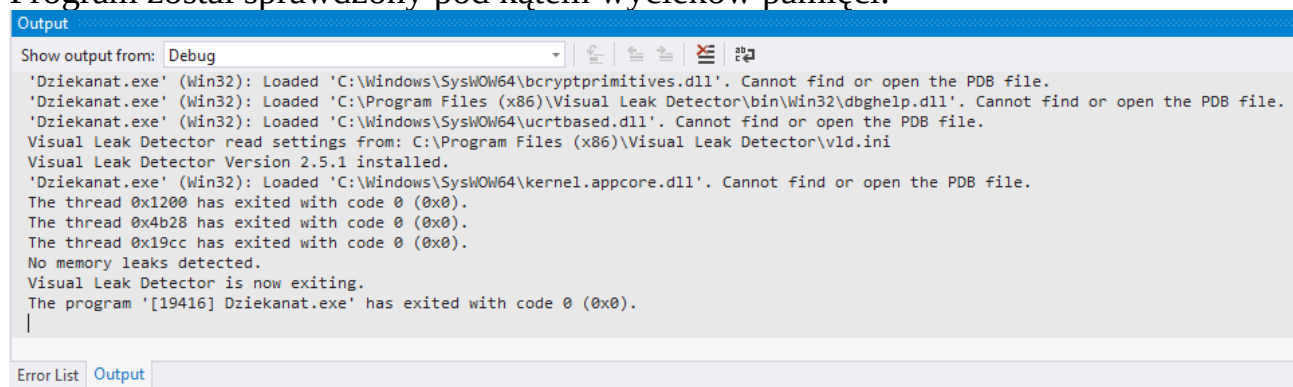
## 4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

## 5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki zawierające tylko nazwę przedmiotu, oraz prowadzącego nie zwracają błędów, natomiast pliki zawierające nazwę przedmiotu bądź prowadzącego zajęcia, w formacie innym niż taki, w którym 1 linijka pliku tekstowego zawiera nazwę przedmiotu, a druga dane prowadzącego, również zgłoszą informację o niepoprawnym formacie danych w danym pliku. Natomiast kolejne dane, czyli imię studenta, nazwisko studenta, jego numer albumu, uzyskana ocena, oraz data nie muszą być podawane w osobnych liniijkach, program wczyta dane poprawnie, jeśli wszystkie z oczekiwanych danych w pliku się tam znajdują. Jeśli w jakimkolwiek miejscu pliku znajdzie się puste pole zamiast odpowiedniej danej, program nie będzie działał poprawnie.

Program został sprawdzony pod kątem wycieków pamięci.



```
Output
Show output from: Debug
'Dziekanat.exe' (Win32): Loaded 'C:\Windows\SysWOW64\bcryptprimitives.dll'. Cannot find or open the PDB file.
'Dziekanat.exe' (Win32): Loaded 'C:\Program Files (x86)\Visual Leak Detector\bin\Win32\dbghelp.dll'. Cannot find or open the PDB file.
'Dziekanat.exe' (Win32): Loaded 'C:\Windows\SysWOW64\ucrtbased.dll'. Cannot find or open the PDB file.
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
'Dziekanat.exe' (Win32): Loaded 'C:\Windows\SysWOW64\kernel.appcore.dll'. Cannot find or open the PDB file.
The thread 0x1200 has exited with code 0 (0x0).
The thread 0x4b28 has exited with code 0 (0x0).
The thread 0x19cc has exited with code 0 (0x0).
No memory leaks detected.
Visual Leak Detector is now exiting.
The program '[19416] Dziekanat.exe' has exited with code 0 (0x0).
|
```

## 6    **Wnioski**

Program do generowania plików na podstawie dostarczonych protokołów jest programem prostym, chociaż wymaga samodzielnego zarządzania pamięcią. Najbardziej wymagające okazało się sortowanie listy ocen dla każdego studenta. Wymagało to rozpatrzenia wielu przypadków. Problem również sprawiło mi zwolnienie zaalokowanej dynamicznie pamięci, aby pozbyć się wycieków pamięci.

# **Dodatek**

## **Szczegółowy opis typów i funkcji**

Dziekanat

Generated by Doxygen 1.8.15





# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">mark</a>	.....	??
<a href="#">student</a>	.....	??



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/ <a href="#">funkcje.h</a>	. . . . .	??
C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/ <a href="#">struktury.h</a>	. . . . .	??



## Chapter 3

# Class Documentation

### 3.1 mark Struct Reference

```
#include <struktury.h>
```

Collaboration diagram for mark:



#### Public Attributes

- float [ocena](#)  
*ocena studenta*
- std::string [subject](#)  
*przedmiot, z ktorego ocene przechowujemy*
- std::string [teacher](#)  
*prowadzacy dany przedmiot, z ktorego ocene przechowujemy*
- std::string [data](#)  
*data wpisu*
- [mark](#) \* [next](#)  
*wskaznik na nastepny element listy*

#### 3.1.1 Detailed Description

Struktura mark jest lista jednokierunkowa

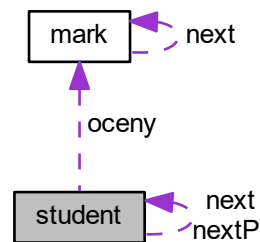
The documentation for this struct was generated from the following file:

- C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/[struktury.h](#)

## 3.2 student Struct Reference

```
#include <struktury.h>
```

Collaboration diagram for student:



### Public Attributes

- `std::string name`  
*imie studenta*
- `std::string surname`  
*nazwisko studenta*
- `int album_number`  
*numer albumu studenta*
- `student * next`  
*wskaznik na lewe dziecko*
- `student * nextP`  
*wskaznik na prawe dziecko*
- `mark * oceny`  
*wskaznik na liste jednokierunkowa ocen*

### 3.2.1 Detailed Description

Struktura student jest drzewem binarnym. Kazdy element drzewa binarnego zawiera w sobie liste jednokierunkowa mark

The documentation for this struct was generated from the following file:

- C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/[struktury.h](#)

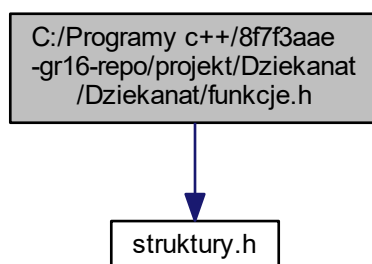
## Chapter 4

# File Documentation

### 4.1 C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/funkcje.h File Reference

```
#include "struktury.h"
```

Include dependency graph for funkcje.h:



### Functions

- void `dodajStudenta` (`student` \*`pRoot`, const std::string &`imie`, const std::string &`nazwisko`, int `nr_albumu`, float `ocena`, const std::string &`przedmiot`, const std::string &`prowadzacy`, const std::string &`data`)
- void `wczytaj` (const std::string &`nazwa`, `student` \*`pRoot`)
- `student` \* `znajdz` (`student` \*`pRoot`, int `nr_albumu`)
- void `dodajOcene` (`mark` \*`pMark`, float `ocena`, const std::string &`przedmiot`, const std::string &`prowadzacy`, const std::string &`dat`)
- void `dodajDoStruktury` (`student` \*`pRoot`, const std::string &`imie`, const std::string &`nazwisko`, int `nr_albumu`, float `ocena`, const std::string &`przedmiot`, const std::string &`prowadzacy`, const std::string &`data`)
- void `generujPlik` (const `student` \*`pRoot`)
- void `usunOcene` (`mark` \*`pOcena`)
- void `usun` (`student` \*`pRoot`)
- void `zapisz` (const `student` \*`pRoot`)

### 4.1.1 Function Documentation

#### 4.1.1.1 dodajDoStruktury()

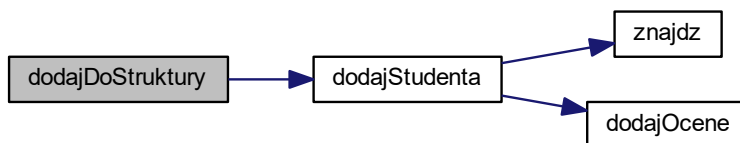
```
void dodajDoStruktury (
    student *& pRoot,
    const std::string & imie,
    const std::string & nazwisko,
    int nr_albumu,
    float ocena,
    const std::string & przedmiot,
    const std::string & prowadzacy,
    const std::string & data )
```

Funkcja dodaje odczytane dane do struktury. Jesli wskaznik na korzen drzewa wskazuje na null, w funkcji tworzony jest obiekt mark z ocenami studenta, oraz obiekt student z danymi studenta. W innym wypadku funkcja wywoluje funkcje dodajaca studenta, ktora zajmuje sie obsluga innych wypadkow

##### Parameters

in, out	<i>pRoot</i>	wskaznik na korzen drzewa.
in	<i>imie</i>	imie studenta.
in	<i>nazwisko</i>	nazwisko studenta.
in	<i>nr_albumu</i>	numer albumu studenta.
in	<i>ocena</i>	ocena uzyskana przez studenta.
in	<i>przedmiot</i>	nazwa przedmiotu z ktorego student otrzymal ocene.
in	<i>prowadzacy</i>	dane prowadzacego dany przedmiot, z ktorego student otrzymal ocene.
in	<i>data</i>	data wpisu oceny.

Here is the call graph for this function:



#### 4.1.1.2 dodajOcene()

```
void dodajOcene (
    mark *& pMark,
```



```
float ocena,
const std::string & przedmiot,
const std::string & prowadzacy,
const std::string & dat )
```

Kazdy student ma swoja liste ocen. Funkcja ta dodaje ocene do listy ocen studenta w porzadku alfabetycznym wedlug nazwy przedmiotu, z ktorego student otrzymal ocene

#### Parameters

in, out	<i>pMark</i>	wskaznik na poczatek listy ocen(danego studenta).
in	<i>ocena</i>	ocena uzyskana przez studenta.
in	<i>przedmiot</i>	nazwa przedmiotu z ktorego student otrzymal ocene.
in	<i>prowadzacy</i>	dane prowadzacego dany przedmiot, z ktorego student otrzymal ocene.
in	<i>dat</i>	data wpisu oceny.

#### 4.1.1.3 dodajStudenta()

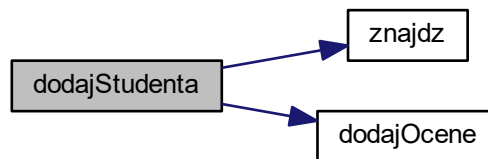
```
void dodajStudenta (
    student *& pRoot,
    const std::string & imie,
    const std::string & nazwisko,
    int nr_albumu,
    float ocena,
    const std::string & przedmiot,
    const std::string & prowadzacy,
    const std::string & data )
```

Funkcja dodaje studenta do struktury drzewa, oraz wywołuje funkcję dodajaca ocene dla danego studenta, dlatego jako parametry przyjmuje dane dotyczace ocen. Funkcja Przyjmuje 8 parametrow

#### Parameters

in, out	<i>pRoot</i>	wskaznik na korzen drzewa.
in	<i>imie</i>	imie studenta.
in	<i>nazwisko</i>	nazwisko studenta.
in	<i>nr_albumu</i>	numer albumu studenta.
in	<i>ocena</i>	ocena uzyskana przez studenta.
in	<i>przedmiot</i>	nazwa przedmiotu z ktorego student otrzymal ocene.
in	<i>prowadzacy</i>	dane prowadzacego dany przedmiot, z ktorego student otrzymal ocene.
in	<i>data</i>	data wpisu oceny.

Here is the call graph for this function:



#### 4.1.1.4 generujPlik()

```
void generujPlik (
    const student * pRoot )
```

Funkcja generuje plik tekstowy o nazwie według schematu: numer albumu + ".txt". Do pliku tekstowego wypisuje dane z elementu drzewa na który wskazuje wskaźnik pRoot, oraz wszystkie elementy posortowanej listy, której wskaźnik na początek jest w pRoot->ocena.

##### Parameters

in	<i>pRoot</i>	wskaźnik na korzeń drzewa.
----	--------------	----------------------------

#### 4.1.1.5 usun()

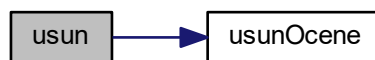
```
void usun (
    student *& pRoot )
```

Funkcja usuwa zaalokowane dynamicznie elementy struktury student. Przed usunięciem elementu struktury student, wywoływana jest funkcja usunOcene, z parametrem pRoot->oceny, kasująca wszystkie oceny danego studenta.

##### Parameters

in, out	<i>pRoot</i>	wskaźnik na korzeń drzewa.
---------	--------------	----------------------------

Here is the call graph for this function:



#### 4.1.1.6 usunOcene()

```
void usunOcene (
    mark *& pOcena )
```

Funkcja usuwa zaalokowane dynamicznie elementy struktury mark

##### Parameters

in, out	pOcena	wskaznik na poczatek listy ocen.
---------	--------	----------------------------------

#### 4.1.1.7 wczytaj()

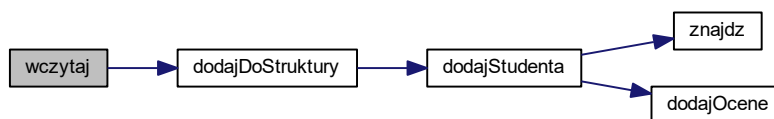
```
void wczytaj (
    const std::string & nazwa,
    student *& pRoot )
```

Funkcja otwiera plik wejsciowy, probuje otworzyc dany plik, jesli sie uda, oraz dane wejsciowe sa poprawne, to wywoluje funkcje dodajaca dane do struktury.

##### Parameters

in	nazwa	nazwa pliku wejsciowego.
in, out	pRoot	wskaznik na korzen drzewa

Here is the call graph for this function:



#### 4.1.1.8 zapisz()

```
void zapisz (
    const student * pRoot )
```

Funkcja iteruje po kolejnych elementach struktury drzewa, wywołując dla każdego elementu funkcję generującą plik - generujPlik.

##### Parameters

in	<i>pRoot</i>	wskaznik na korzen drzewa.
----	--------------	----------------------------

Here is the call graph for this function:



#### 4.1.1.9 znajdz()

```
student* znajdz (
    student * pRoot,
    int nr_albumu )
```

Funkcja sprawdza czy student o numerze albumu przyjmowanym jako parametr istnieje już w strukturze drzewa, której wskaznik na korzen jest przekazywany jako argument o zmiennej pRoot

## Parameters

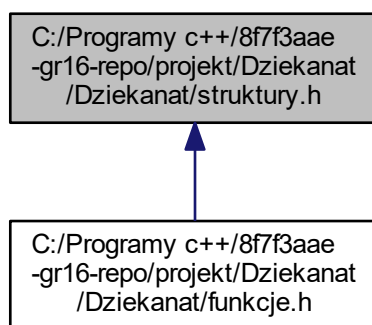
in	<i>pRoot</i>	wskaznik na korzen drzewa
in	<i>nr_albumu</i>	numer albumu studenta

## Returns

student \* znaleziony jesli istnieje element o podanym numerze albumu w strukturze, nullptr jeli nie odnaleziono takiego elementu.

## 4.2 C:/Programy c++/8f7f3aae-gr16-repo/projekt/Dziekanat/Dziekanat/struktury.h File Reference

This graph shows which files directly or indirectly include this file:



## Classes

- struct [mark](#)
- struct [student](#)

