


Sprawozdanie z programów z listy czwartej			
 <p>Wrocław University of Science and Technology</p>	Student:	Data Laboratoriów:	Prowadzący:
		29.03.2017r. 05.04.2017r. g.16:30	Dr inż. Krzysztof Halawa
		Wykonano:	
		Grupa laboratoryjna: E02-18p	Ocena:

Wstęp:

Programy opierały się na tworzeniu abstrakcyjnych formatów (baz) danych. Autor w dalszej części dokumentu będzie prezentował swoje programy w **języku C/C++**. Do programów obiektowych został stworzony plik makefile do kompilacji.

Programy obiektowe zostały napisane zgodnie ze sztuką hermetyzacji danych oraz używają szablonów.

W kodzie nie ma wielu komentarzy, jednak wszelakie **funkcje** oraz **zmienne są nazywane zgodnie z zastosowaniem** w języku angielskim, tak aby **kod był czytelny i przejrzysty**.

Wnioski nie są zbierane pod koniec dokumentu, są one wypisywane przy konkretnym zagadnieniu.

W programach są używane wskaźniki oraz zmienne dynamiczne, aby sprawdzić czy nie następują wycieki pamięci posłużono się programem valgrind z opcją leak-check=full.

Programy nie korzystają z STL zgodnie z zaleceniami wykładowcy.

Program 1 & 4

1. Należy zaimplementować kolejkę priorytetową bazującą na sekwencji przechowującą elementy określonego typu (np.: int, float, itp.). Należy napisać funkcje wykonujące podstawowe operacje na tej kolejce (dodawanie i usuwanie elementów zgodnie z opisem na wykładzie).
4. Korzystając z kolejki priorytetowej z zadania 1 należy zaimplementować jedno z sortowań bazujących na kolejce priorytetowej (sortowanie przez wybór lub przez wstawianie). Proszę zaprezentować działanie dla 10 - 15 elementów.

Opis:

Do żadnego z programów nie należało tworzyć menu, program 4 wykorzystuje program 1, co powinno pokazać że oba są zrealizowane poprawnie, jednak aby pokazać w sprawozdaniu wszystkie funkcje kolejki priorytetowej, autor zdecydował się na menu do programu 1.

Kod programu:

Program 1:

https://github.com/Rexluu/PAMSI/tree/Final_versions/List_4/L4_Z1

Program 2:

https://github.com/Rexluu/PAMSI/tree/Final_versions/List_4/L4_Z4

Testy:

Podczas testów, w tle uruchomiony jest valgrind.

Test na pokazanie poprawnego dodawania i wyświetlania kolejki priorytetowej

- - - - -

Program realizujący kolejke priorytetowa
Typ danych ustalony w kodzie zrodlowym programu

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 1

[priorytet, dana]: 5 1

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 1

[priorytet, dana]: 4 2

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 1

[priorytet, dana]: 6 3

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 4

Nr 1 : 2

Nr 2 : 1

Nr 3 : 3

Usuwanie oraz zdublowany priorytet

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 1

[priorytet, dana]: 5 4

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 4

Nr 1 : 2
Nr 2 : 1
Nr 3 : 4
Nr 4 : 3

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 2

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 4

Nr 1 : 1
Nr 2 : 4
Nr 3 : 3

Mieszane dodawanie, usuwanie, wyswietlanie najmniejszego priorytetu, wyswietlanie calej kolejki.

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyswietl element z najmniejszym priorytetem
4. Wyswietl kolejke
5. Usun kolejke i wyjdz z programu

Opcja [1-4]: 3

- 1
1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyswietl element z najmniejszym priorytetem
4. Wyswietl kolejke
5. Usun kolejke i wyjdz z programu

Opcja [1-4]: 2

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyswietl element z najmniejszym priorytetem
4. Wyswietl kolejke
5. Usun kolejke i wyjdz z programu

Opcja [1-4]: 1

[priorytet, dana]: 8 5

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyswietl element z najmniejszym priorytetem
4. Wyswietl kolejke
5. Usun kolejke i wyjdz z programu

Opcja [1-4]: 1

[priorytet, dana]: 0 6

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyswietl element z najmniejszym priorytetem
4. Wyswietl kolejke
5. Usun kolejke i wyjdz z programu

Opcja [1-4]: 4

Nr 1 : 6
Nr 2 : 4
Nr 3 : 3
Nr 4 : 5

Komunikat valgrind o braku wycieku pamięci:

1. Dodaj element
2. Usun element z najmniejszym priorytetem
3. Wyświetl element z najmniejszym priorytetem
4. Wyświetl kolejke
5. Usun kolejke i wyjdź z programu

Opcja [1-4]: 5

```
==3298==
==3298== HEAP SUMMARY:
==3298==      in use at exit: 0 bytes in 0 blocks
==3298==    total heap usage: 6 allocs, 6 frees, 192 bytes allocated
==3298==
==3298== All heap blocks were freed -- no leaks are possible
==3298==
==3298== For counts of detected and suppressed errors, rerun with: -v
==3298== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Test programu nr 4:

```
text@debian Laptop: ~/src1/40/L4_Z4$ valgrind --leak-check=full ./L4_Z4
==3357== Memcheck, a memory error detector
==3357== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3357== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==3357== Command: ./L4_Z4
==3357==
```

Liczby do posortowania:

2 18 19 14 33 26 19 35 11 12 37 26 25 42 41 2 9 34 12 4

Nr 1 : 2
Nr 2 : 2
Nr 3 : 4
Nr 4 : 9
Nr 5 : 11
Nr 6 : 12
Nr 7 : 12
Nr 8 : 14
Nr 9 : 18
Nr 10 : 19
Nr 11 : 19
Nr 12 : 25
Nr 13 : 26
Nr 14 : 26
Nr 15 : 33
Nr 16 : 34
Nr 17 : 35
Nr 18 : 37
Nr 19 : 41
Nr 20 : 42

```
==3357==
==3357== HEAP SUMMARY:
==3357==      in use at exit: 0 bytes in 0 blocks
==3357==    total heap usage: 20 allocs, 20 frees, 640 bytes allocated
==3357==
==3357== All heap blocks were freed -- no leaks are possible
==3357==
==3357== For counts of detected and suppressed errors, rerun with: -v
==3357== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Program 2 & 5

2. Należy zaimplementować ogólny przypadek drzewa przechowującą elementy określonego typu (np.: int, float, itp.). Należy napisać funkcje wykonujące podstawowe operacje na drzewie (dodawanie i usuwanie elementów, wyświetlanie elementu w korzeniu drzewa, wyświetlanie wysokości drzewa).
5. Korzystając z drzew z zadań 2 i 3 należy zaimplementować przejścia pre-order, in-order i post-order. Należy zaprezentować działanie algorytmów osobno dla drzewa w postaci ogólnej i binarnej.

Opis:

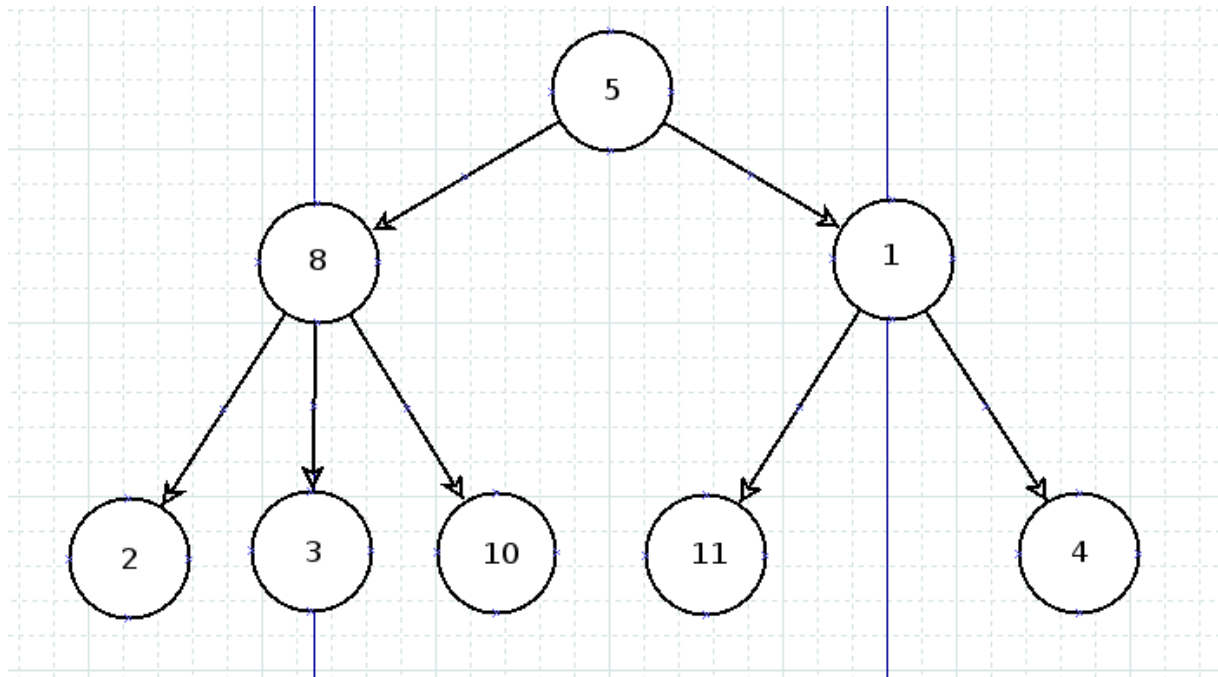
Program 5, a właściwie to zadanie zawiera się w funkcjonalności programu 2, dlatego też zostały one połączone w jeden podpunkt.

Kod programu:

https://github.com/Rexluu/PAMSI/tree/Final_versions/List_4/L4_Z2

Testy:

Graficzna interpretacja dodawanych liczb wykonana w programie Dia oraz testy programu wraz z prezentacją pre,post-order. **Rozmiar jest wysokością, a nie głębokością.**



```
==5741== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.  
==5741== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info  
==5741== Command: ./L4_Z2  
==5741==
```

Liczby dodawane na drzewo:
5 8 1 2 3 10 11 4

Root data: 5
Pre-order: 5 | 8 | 2 | 3 | 10 | 1 | 11 | 4 |
Post-order: 2 | 3 | 10 | 8 | 11 | 4 | 1 | 5 |
Rozmiar: 3

Po usunięciu danej numer 8

Root data: 5
Pre-order: 5 | 1 | 11 | 4 |
Post-order: 11 | 4 | 1 | 5 |
Rozmiar: 3

```
==5741==  
==5741== HEAP SUMMARY:  
==5741==    in use at exit: 0 bytes in 0 blocks  
==5741== total heap usage: 15 allocs, 15 frees, 432 bytes allocated  
==5741==  
==5741== All heap blocks were freed -- no leaks are possible  
==5741==  
==5741== For counts of detected and suppressed errors, rerun with: -v  
==5741== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```


Program 3 & 5

3. Należy zaimplementować drzewo binarne przechowujące elementy określonego typu (np.: int, float, itp.). Należy napisać funkcje wykonujące podstawowe operacje na tym drzewie (dodawanie i usuwanie elementów, wyświetlanie elementu w korzeniu drzewa, wyświetlanie wysokości drzewa).

5. Korzystając z drzew z zadań 2 i 3 należy zaimplementować przejścia pre-order, in-order i post-order. Należy zaprezentować działanie algorytmów osobno dla drzewa w postaci ogólnej i binarnej.

Opis:

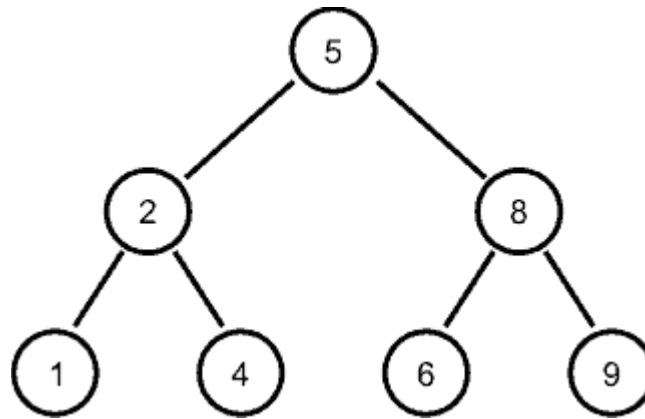
Program 5, a właściwie to zadanie zawiera się w funkcjonalności programu 3, dlatego też zostały one połączone w jeden podpunkt.

Kod programu:

https://github.com/Rexluu/PAMSI/tree/Final_versions/List_4/L4_Z3

Testy:

Graficzna interpretacja wpisywanego drzewa



W tle chodził valgrind monitorujący wycieki danych, poniżej test dodawania i usuwania dowolnego elementu (w tym przypadku elementu nr 2). Zaprezentowane są również przejścia pre,post,in order. **Rozmiar jest wysokością drzewa a nie jego głębokością.**

```
====6862==== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.  
====6862==== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info  
====6862==== Command: ./L4_Z3  
====6862====
```

Elementy do dodania na drzewo:

5 2 8 1 4 6 9

Pre-order: 5 | 2 | 1 | 4 | 8 | 6 | 9 |

Post-order: 1 | 4 | 2 | 6 | 9 | 8 | 5 |

In-order: 1 | 2 | 4 | 5 | 6 | 8 | 9 |

Rozmiar: 3

Dana root: 5

Post-order po usunięciu numeru 2: 6 | 9 | 8 | 5 |

Rozmiar: 3

```
====6862====
```

```
====6862==== HEAP SUMMARY:
```

```
====6862==== in use at exit: 0 bytes in 0 blocks
```

```
====6862==== total heap usage: 7 allocs, 7 frees, 224 bytes allocated
```

```
====6862====
```

```
====6862==== All heap blocks were freed -- no leaks are possible
```

```
====6862====
```

```
====6862==== For counts of detected and suppressed errors, rerun with: -v
```

```
====6862==== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```