


Sprawozdanie z programów z listy drugiej			
 <p>Wrocław University of Science and Technology</p>	<b>Student:</b>  <i>Jakub Król</i> 226269	<b>Data Laboratoriów:</b> 08.03.2017r. g.16:30	<b>Prowadzący:</b> <i>Dr inż. Krzysztof Halawa</i>
		<b>Wykonano:</b> 14.03.2017r.	
		<b>Grupa laboratoryjna:</b> E02-18p	<b>Ocena:</b>

## Wstęp:

Programy głównie opierały się na przypomnieniu **języka obiektowego oraz wskaźników**. 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 wielu komentarzy, jednak wszelakie **funkcje oraz zmienne są nazywane zgodnie z zastosowaniem** w języku angielskim, tak aby **kod był czytelny i przejrzysty**.

**Zgodnie z zaleceniami** wykładowcy: Dr inż. Łukasz Jeleń, **nie został wykorzystany STL**.

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

Programy nie miały udostępniać interfejsu użytkownika, jednak do testów został on zaprogramowany i w większości pozabezpieczony.

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

\*Z powodu tego, że wykład pojawił się b.późno (fizycznego na uczelni jeszcze nie było), a opis zadań nie był zbyt jasny (podpunkty mocno mieszały) do rozróżnienia kolejki od zadanej listy, autor programów posłużył się definicją zaczerpniętą ze strony:

<https://www.quora.com/What-is-the-difference-between-Queue-and-List>

## Program nr 1:

1. Należy napisać funkcję rekurencyjną, która będzie generowała wszystkie permutacje łańcucha znakowego przekazanego do tej funkcji.
  - (a) Korzystając z funkcji `jestPal` z poprzednich zajęć zmodyfikuj problem generowania permutacji w taki sposób, aby dla każdej wygenerowanej permutacji program sprawdzał, czy jest to palindrom. Jeśli tak, to należy ten łańcuch zapisać do tablicy zawierającej znalezione palindromy `palList` zdefiniowanej globalnie.
  - (b) Po stworzeniu tablicy z palindromami okaże się, że zawiera ona duplikaty. Należy stworzyć funkcję `usunDup()`, która usunie duplikaty z tablicy. Funkcja ta nie musi być rekurencyjna.

Kod programu znajduje się pod linkiem:

[https://github.com/Rexluu/PAMSI/blob/Final\\_versions/List\\_2/L2\\_Z1.cpp](https://github.com/Rexluu/PAMSI/blob/Final_versions/List_2/L2_Z1.cpp)

## Testy:

```
- - - - -
Program do generowania wszystkich mozliwych permutacji,
wypisuje wszystkie utworzone w ten sposob palindromy
```

Podaj slowo: kajak

```
0: kajak
1: akjka
```

```
- - - - -
==6508==
==6508== HEAP SUMMARY:
==6508==    in use at exit: 0 bytes in 0 blocks
==6508== total heap usage: 705 allocs, 705 frees, 21,271 bytes allocated
==6508==
==6508== All heap blocks were freed -- no leaks are possible
==6508==
==6508== For counts of detected and suppressed errors, rerun with: -v
==6508== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
- - - - -
```

```
Program do generowania wszystkich mozliwych permutacji,
wypisuje wszystkie utworzone w ten sposob palindromy
```

Podaj slowo: PAMSI

Brak palindromow

```
==6840==
==6840== HEAP SUMMARY:
==6840==    in use at exit: 0 bytes in 0 blocks
==6840== total heap usage: 571 allocs, 571 frees, 17,103 bytes allocated
==6840==
==6840== All heap blocks were freed -- no leaks are possible
==6840==
==6840== For counts of detected and suppressed errors, rerun with: -v
==6840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
.....  
Program do generowania wszystkich mozliwych permutacji,  
wypisuje wszystkie utworzone w ten sposob palindromy
```

Podaj slowo: 123123

```
0: 123321  
1: 132231  
2: 213312  
3: 231132  
4: 312213  
5: 321123
```

```
.....  
==6535==  
==6535== HEAP SUMMARY:  
==6535==    in use at exit: 0 bytes in 0 blocks  
==6535== total heap usage: 12,756 allocs, 12,756 frees, 411,166 bytes allocated  
==6535==  
==6535== All heap blocks were freed -- no leaks are possible  
==6535==  
==6535== For counts of detected and suppressed errors, rerun with: -v  
==6535== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

## Program nr 2:

2. Należy zaimplementować listę jednokierunkową przechowującą elementy określonego typu (np.: int, float, itp.). Należy napisać funkcje wykonujące następujące operacje:

- (a) dodawanie elementu do listy
- (b) usuwanie pojedynczego elementu z listy
- (c) wyświetlanie zawartości listy
- (d) usuwanie wszystkich elementów z listy

Kod programu znajduje się pod linkiem:

[https://github.com/Rexluu/PAMSI/tree/Final\\_versions/List\\_2/L2\\_Z2](https://github.com/Rexluu/PAMSI/tree/Final_versions/List_2/L2_Z2)

## Testy:

*W pliku źródłowym ustawiono typ „int”, następnie typ „char” w celu pokazania tylko, że „program działa na szablonach”, cała reszta testów została przeprowadzona w jednym ciągu z monitoringiem programu valgrind na typach int.*

- - - - -

Program realizujący listę jednokierunkową (FIFO)

Typ danych ustalony w kodzie źródłowym programu

- 1. Dodaj element
- 2. Usun element
- 3. Wyświetl listę
- 4. Usun listę i wyjdź z programu

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 1 c

- 1. Dodaj element
- 2. Usun element
- 3. Wyświetl listę
- 4. Usun listę i wyjdź z programu

Opcja [1-4]: 3

Nr 1 : c

Dodaj [nr\_elem][dana]: 1 1

- 1. Dodaj element
- 2. Usun element
- 3. Wyświetl listę
- 4. Usun listę i wyjdź z programu

Opcja [1-4]: 3

Nr 1 : 1

*Test na dodanie elementu, dodanie na miejsce pierwsze oraz usuniecie z miejsca pierwszego*

- - - - -  
Program realizujący liste jednokierunkowa (FIFO)  
Typ danych ustalony w kodzie zrodlowym programu

1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 1 1

1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 1 0

1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu

Opcja [1-4]: 3

Nr 1 : 0

Nr 2 : 1

1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu

Opcja [1-4]: 2

Podaj numer elementu [1-2]: 1

1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu

Opcja [1-4]: 3

Nr 1 : 1

*Test na dodanie na koniec elementow, na dodanie nowych oraz na dodanie w srodku*

```
1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu
```

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 2 2

```
1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu
```

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 3 3

```
1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu
```

Opcja [1-4]: 1

Dodaj [nr\_elem][dana]: 2 0

```
1. Dodaj element
2. Usun element
3. Wyszwietl liste
4. Usun liste i wyjdz z programu
```

Opcja [1-4]: 3

Nr 1 : 1

Nr 2 : 0

Nr 3 : 2

Nr 4 : 3

*Test na usunięcie danych ze środka oraz na usunięcie z końca*

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 2

Podaj numer elementu [1-4]: 3

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 2

Podaj numer elementu [1-3]: 4

Podano nieistniejący element: 4

Prawidłowa wartość [1-3]

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 2

Podaj numer elementu [1-3]: 3

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 3

Nr 1 : 1

Nr 2 : 0

*Test na usunięcie wszystkich istniejących oraz wykaz (braku) wycieku pamięci*

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 4

```
==6854==
==6854== HEAP SUMMARY:
==6854==    in use at exit: 0 bytes in 0 blocks
==6854==  total heap usage: 10 allocs, 10 frees, 210 bytes allocated
==6854==
==6854== All heap blocks were freed -- no leaks are possible
==6854==
==6854== For counts of detected and suppressed errors, rerun with: -v
==6854== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

*Dodatkowy test na pokazanie, że program zwraca informacje, jeżeli lista jest pusta*

```
- - - - -
Program realizujący listę jednokierunkową (FIFO)
Typ danych ustalony w kodzie źródłowym programu
```

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 3

Lista jest pusta

1. Dodaj element
2. Usun element
3. Wyświetl listę
4. Usun listę i wyjdź z programu

Opcja [1-4]: 2

Lista jest pusta



### **Program nr 3:**

3. Należy zaimplementować kolejkę przechowującą elementy określonego typu (np.: int, float, itp.). Należy napisać funkcje wykonujące następujące operacje:

- (a) dodawanie elementu do kolejki
- (b) usuwanie pojedynczego elementu z kolejki
- (c) wyświetlanie zawartości kolejki
- (d) usuwanie wszystkich elementów z kolejki

Kod programu znajduje się pod linkiem:

[https://github.com/Rexluu/PAMSI/tree/Final\\_versions/List\\_2/L2\\_Z3](https://github.com/Rexluu/PAMSI/tree/Final_versions/List_2/L2_Z3)

### **Testy:**

Cały test został przeprowadzony na typie „int” wprowadzonym w kodzie źródłowym programu - „Queue <int> queue”. Klasy oparte są na szablonach tak jak w programie nr 2.

Cały test został przeprowadzony w jednym ciągu z monitoringiem programu valgrind

#### **Test na dodanie oraz wyświetlenie kolejki**

- - - - -  
Program realizujący kolejkę (FIFO)  
Typ danych ustalony w kodzie źródłowym programu

- 1. Dodaj element
- 2. Usun element
- 3. Wyświetl kolejkę
- 4. Usun kolejkę i wyjdź z programu

Opcja [1-4]: 1

Dodaj [dana]: 1

- 1. Dodaj element
- 2. Usun element
- 3. Wyświetl kolejkę
- 4. Usun kolejkę i wyjdź z programu

Opcja [1-4]: 3

Nr 1 : 1

## Test na dodanie nowych elementow, oraz usuniecie

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

Opcja [1-4]: 1

Dodaj [dana]: 2

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

Opcja [1-4]: 1

Dodaj [dana]: 3

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

Opcja [1-4]: 3

Nr 1 : 1

Nr 2 : 2

Nr 3 : 3

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

Opcja [1-4]: 2

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

Opcja [1-4]: 3

Nr 1 : 2

Nr 2 : 3

Test na komunikaty o pustej kolejce oraz usuwanie do konca i dodanie elementu ponownie

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

Opcja [1-4]: 2

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

Opcja [1-4]: 2

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

Opcja [1-4]: 2

Kolejka jest pusta

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

Opcja [1-4]: 3

Kolejka jest pusta

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

Opcja [1-4]: 1

Dodaj [dana]: 1

Test na usunięcie wszystkich elementów naraz, oraz zwrócenie komunikatu o braku wycieku pamięci.

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

Opcja [1-4]: 1

Dodaj [dana]: 2

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

Opcja [1-4]: 1

Dodaj [dana]: 3

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

Opcja [1-4]: 3

Nr 1 : 1

Nr 2 : 2

Nr 3 : 3

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

Opcja [1-4]: 4

==5205==

==5205== HEAP SUMMARY:

==5205== in use at exit: 0 bytes in 0 blocks

==5205== total heap usage: 6 allocs, 6 frees, 96 bytes allocated

==5205==

==5205== All heap blocks were freed -- no leaks are possible

==5205==

==5205== For counts of detected and suppressed errors, rerun with: -v

==5205== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

rexlu@Debian-Laptop:~/PAMSI-lab/List\_2/L2\_Z3\$ █

## Program nr 4:

5. Należy zaimplementować tzw. kolejkę z dwoma końcami -> deque. Korzystając z implementacji deque należy stworzyć program, który będzie sprawdzał, czy podany łańcuch znakowy jest palindromem. Na wykładzie widzieli Państwo implementację ADT stosu i kolejki. Deque ADT łączy ze sobą funkcjonalności obu tych implementacji. Będziemy mogli zatem dodawać i usuwać elementy z obu stron deque'a. Poniżej zamieszczono przykładowy interfejs Deque ADT:

Kod programu znajduje się pod linkiem:

[https://github.com/Rexluu/PAMSI/tree/Final\\_versions/List\\_2/L2\\_Z4](https://github.com/Rexluu/PAMSI/tree/Final_versions/List_2/L2_Z4)

## Testy:

Program nie jest sprawdzany pod względem poprawności działania deque, samo sprawdzenie czy podany wyraz jest palindromem potwierdzi poprawność działania deque (deque zostało zaimplementowane)

### Test na wyraz literowy

```
==5398== Memcheck, a memory error detector
==5398== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==5398== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==5398== Command: ./L2_Z4
==5398==

- - - - -
Program sprawdza palindrom w oparciu o Deque

Podaj slowo: kajak

Podane slowo jest palindromem
==5398==
==5398== HEAP SUMMARY:
==5398==    in use at exit: 0 bytes in 0 blocks
==5398==   total heap usage: 5 allocs, 5 frees, 80 bytes allocated
==5398==
==5398== All heap blocks were freed -- no leaks are possible
==5398==
==5398== For counts of detected and suppressed errors, rerun with: -v
==5398== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

### Test na rozna wielkosc liter

```
- - - - -
Program sprawdza palindrom w oparciu o Deque

Podaj slowo: KaJaK

Podane slowo jest palindromem
==5402==
==5402== HEAP SUMMARY:
==5402==    in use at exit: 0 bytes in 0 blocks
==5402==  total heap usage: 5 allocs, 5 frees, 80 bytes allocated
==5402==
==5402== All heap blocks were freed -- no leaks are possible
==5402==
==5402== For counts of detected and suppressed errors, rerun with: -v
==5402== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

### Test na cyfry oraz rozpoznanie braku palindromu

```
- - - - -
Program sprawdza palindrom w oparciu o Deque

Podaj slowo: 123123

Podano slowo nie jest palindromem
==5434==
==5434== HEAP SUMMARY:
==5434==    in use at exit: 0 bytes in 0 blocks
==5434==  total heap usage: 6 allocs, 6 frees, 96 bytes allocated
==5434==
==5434== All heap blocks were freed -- no leaks are possible
==5434==
==5434== For counts of detected and suppressed errors, rerun with: -v
==5434== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```