ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине «Системы управления базами данных»

Выполнил: ст. гр. ТКИ – 441

Часников А. А.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2022

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc1)

[ЗАДАНИЕ 3](#_Toc2)

[СТРУКТУРА ПРОЕКТА 3](#_Toc3)

[1 ЛИСТИНГ ПРОГРАММЫ 4](#_Toc4)

[1.1 Файл deque.h 4](#_Toc5)

[1.2 Файл deque.cpp 5](#_Toc6)

[1.3 Файл Odeque.h 7](#_Toc7)

[1.4 Файл Odeque.cpp 8](#_Toc8)

[1.5 Файл test\_deque.cpp 11](#_Toc9)

[1.6 Файл test\_Odeque.cpp 13](#_Toc10)

[2 ДИАГРАММА КЛАССОВ 13](#_Toc11)

[3 РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЙ ТЕСТОВ 15](#_Toc12)

ЗАДАНИЕ

1. Для структуры «Deque» разработать API
2. На языке «C++» написать конструктор и деструктор класса
3. Разработать метод вывода структуры в строку
4. Реализовать основные функции для работы
5. Разработать тесты

СТРУКТУРА ПРОЕКТА

* Deque
  + deque.h
  + deque.cpp
  + Odeque.h
  + Odeque.cpp
* Tests
  + test\_deque.cpp
  + test\_Odeque.cpp

# ЛИСТИНГ ПРОГРАММЫ

## Файл **deque.h**

#include <initializer\_list>

#include <string>

#include <Odeque.cpp>

/\*\*

\* \brief .

\*/

class Deque

{

public:

/\*\*

\* \brief Хвост дека.

\*/

Odeque\* tail;

/\*\*

\* \brief Начало дека.

\*/

Odeque\* head;

/\*\*

\* \brief Размер массива.

\*/

size\_t size;

/\*\*

\* \brief Убирает элемент начиная с переднего фронта .

\* \return .

\*/

Odeque\* RemoveFront() noexcept;

// Убирате элемент начиная с заднего фронта

Odeque\* RemoveBack() noexcept;

/\*\*

\* \brief .

\*/

Deque() noexcept;

/\*\*

\* \brief .

\* \param Odeques .

\*/

Deque(const std::initializer\_list<int> odeques) noexcept;

/\*\*

\* \brief .

\*/

~Deque();

/\*\*

\* \brief .

\* \return .

\*/

bool PushFront(Odeque\* odeque) noexcept;

bool PushBack(Odeque\* odeque) noexcept;

/\*\*

\* \brief .

\* \return .

\*/

bool IsEmpty() const noexcept;

/\*\*

\* \brief .

\*/

void Clear() noexcept;

size\_t GetSize() const noexcept;

std::string ToString() const noexcept;

};

## **Файл deque.cpp**

#include <sstream>

#include "deque.h"

Deque::Deque() noexcept

: tail(nullptr), head(nullptr), size(0)

{

}

Deque::Deque(const std::initializer\_list<int> Odeques) noexcept

: Deque()

{

for (auto it = Odeques.begin(); it != Odeques.end(); it++)

{

Odeque \*el = new Odeque(\*it);

this->PushFront(el);

}

}

Deque::~Deque()

{

this->Clear();

}

bool Deque::PushFront(Odeque\* Odeque) noexcept

{

if (this->IsEmpty())

{

this->tail = this->head = Odeque;

this->size++;

return true;

}

this->tail->next = Odeque;

this->tail = this->tail->next;

this->size++;

return true;

}

bool Deque::PushBack(Odeque\* Odeque) noexcept

{

if (this->IsEmpty())

{

this->tail = this->head = Odeque;

this->size++;

return true;

}

this->head->next = Odeque;

this->head = this->head->next;

this->size++;

return true;

}

Odeque\* Deque::RemoveFront() noexcept

{

auto temp = this->head;

this->head = this->head->next;

this->size--;

return temp;

}

Odeque\* Deque::RemoveBack() noexcept

{

auto temp = this->tail;

this->tail = this->tail->next;

this->size--;

return temp;

}

bool Deque::IsEmpty() const noexcept

{

return this->head == nullptr;

}

void Deque::Clear() noexcept

{

while (!this->IsEmpty())

{

this->RemoveFront();

}

}

size\_t Deque::GetSize() const noexcept

{

return this->size;

}

std::string Deque::ToString() const noexcept

{

std::stringstream buffer;

buffer << "{ ";

auto temp = this->head;

while (temp->next != nullptr)

{

buffer << temp->ToString() << ", ";

temp = temp->next;

}

buffer << temp->ToString() << " }";

return buffer.str();

}

## Файл Odeque**.h**

include<string>

/\*\*

\* \brief

\*/

struct Odeque

{

// Значения элемента

int value;

/\*\*

\* \brief Консрукттор с параметрами.

\* \param value .

\*/

Odeque(const int value) noexcept;

/\*\*

\* \brief .

\*/

~Odeque();

// Указатель на следующий элемент

Odeque\* next;

std::string ToString() const noexcept;

bool operator ==(const Odeque& other);

bool operator !=(const Odeque& other);

};

## Файл Odeque.cpp

#include <sstream>

#include "deque.h"

Deque::Deque() noexcept

: tail(nullptr), head(nullptr), size(0)

{

}

Deque::Deque(const std::initializer\_list<int> Odeques) noexcept

: Deque()

{

for (auto it = Odeques.begin(); it != Odeques.end(); it++)

{

Odeque \*el = new Odeque(\*it);

this->PushFront(el);

}

}

Deque::~Deque()

{

this->Clear();

}

bool Deque::PushFront(Odeque\* Odeque) noexcept

{

if (this->IsEmpty())

{

this->tail = this->head = Odeque;

this->size++;

return true;

}

this->tail->next = Odeque;

this->tail = this->tail->next;

this->size++;

return true;

}

bool Deque::PushBack(Odeque\* Odeque) noexcept

{

if (this->IsEmpty())

{

this->tail = this->head = Odeque;

this->size++;

return true;

}

this->head->next = Odeque;

this->head = this->head->next;

this->size++;

return true;

}

Odeque\* Deque::RemoveFront() noexcept

{

auto temp = this->head;

this->head = this->head->next;

this->size--;

return temp;

}

Odeque\* Deque::RemoveBack() noexcept

{

auto temp = this->tail;

this->tail = this->tail->next;

this->size--;

return temp;

}

bool Deque::IsEmpty() const noexcept

{

return this->head == nullptr;

}

void Deque::Clear() noexcept

{

while (!this->IsEmpty())

{

this->RemoveFront();

}

}

size\_t Deque::GetSize() const noexcept

{

return this->size;

}

std::string Deque::ToString() const noexcept

{

std::stringstream buffer;

buffer << "{ ";

auto temp = this->head;

while (temp->next != nullptr)

{

buffer << temp->ToString() << ", ";

temp = temp->next;

}

buffer << temp->ToString() << " }";

return buffer.str();

}

## Файл test**\_deque.cpp**

#include <deque.cpp>

#include <iostream>

#include <gtest/gtest.h>

TEST(DT, deque\_test) {

// Arange && Act

Deque my\_deque;

// Assert

ASSERT\_TRUE(my\_deque.head == nullptr);

}

TEST(DT, deque\_pushfront) {

// Arange

Odeque\* my\_odeque1 = new Odeque(2);

Odeque\* my\_odeque2 = new Odeque(3);

Deque my\_deque;

// Act

my\_deque.PushFront(my\_odeque1);

my\_deque.PushFront(my\_odeque2);

// Assert

ASSERT\_TRUE(my\_deque.tail->value == 3);

}

TEST(DT, deque\_pushback) {

// Arange

Odeque\* my\_odeque1 = new Odeque(2);

Odeque\* my\_odeque2 = new Odeque(3);

Deque my\_deque;

// Act

my\_deque.PushBack(my\_odeque1);

my\_deque.PushBack(my\_odeque2);

// Assert

ASSERT\_TRUE(my\_deque.head->value == 3);

}

TEST(DT, deque\_removefront) {

// Arange

Odeque\* my\_odeque1 = new Odeque(2);

Odeque\* my\_odeque2 = new Odeque(3);

Deque my\_deque;

// Act

my\_deque.PushFront(my\_odeque1);

my\_deque.PushFront(my\_odeque2);

my\_deque.RemoveFront();

// Assert

ASSERT\_TRUE(my\_deque.head->value == 3);

}

TEST(DT, deque\_removeback) {

// Arange

Odeque\* my\_odeque1 = new Odeque(5);

Odeque\* my\_odeque2 = new Odeque(3);

Deque my\_deque;

// Act

my\_deque.PushFront(my\_odeque1);

my\_deque.PushBack(my\_odeque2);

my\_deque.RemoveBack();

// Assert

ASSERT\_TRUE(my\_deque.head->value == 3);

}

TEST(DT, deque\_tostring) {

// Arange

const std::string expected = "{ 4, 5, 2 }";

Odeque\* my\_odeque1 = new Odeque(4);

Odeque\* my\_odeque2 = new Odeque(5);

Odeque\* my\_odeque3 = new Odeque(2);

Deque my\_deque;

// Act

my\_deque.PushFront(my\_odeque1);

my\_deque.PushFront(my\_odeque2);

my\_deque.PushFront(my\_odeque3);

// Assert

ASSERT\_TRUE(my\_deque.ToString() == expected);

}

int main(int argc, char \*\*argv)

{

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

## Файл test\_Odeque.cpp

#include <Odeque.cpp>

#include <iostream>

#include <gtest/gtest.h>

TEST(ODT, odq) {

// Arange

const int expected\_value = 2;

// Act

Odeque\* my\_deque = new Odeque(expected\_value);

// Assert

ASSERT\_TRUE(my\_deque->value == expected\_value);

}

int main(int argc, char \*\*argv)

{

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

# ДИАГРАММА КЛАССОВ

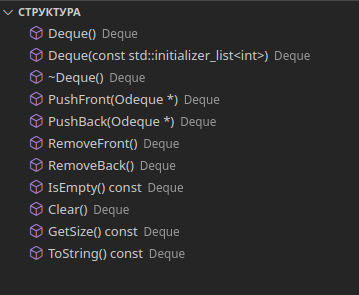


Рисунок 1. Диаграмма

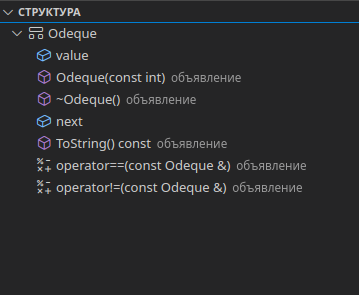


Рисунок 2. Диаграмма

# РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЙ ТЕСТОВ

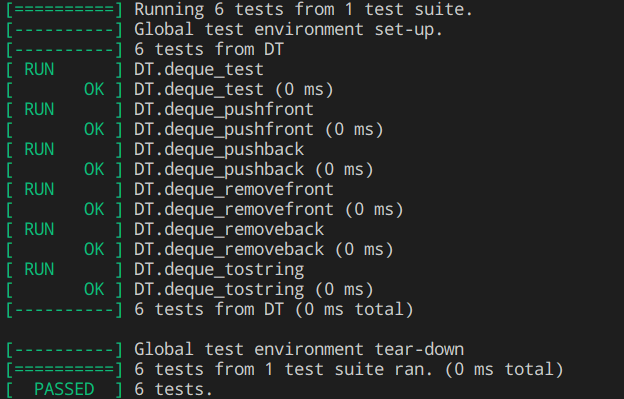


Рисунок 3. Результаты прохождения тестов

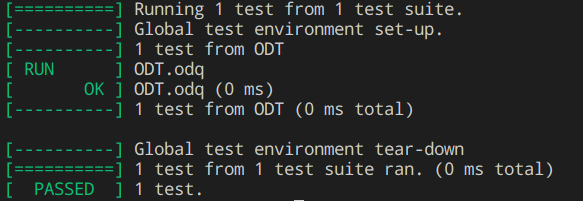


Рисунок 4. Результаты прохождения тестов