МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Лабораторная работа № 1**

по курсу «Объектно-ориентированное программирование»

**«Классы и объекты в C++»**

**10 ВАРИАНТ**

Выполнил:

студенты гр. КТбо2-1

Самардак А.В.

Проверил:

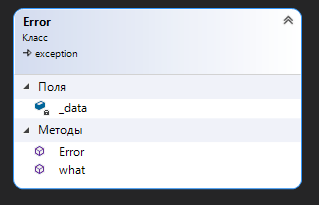
Тарасов С.А.

**Таганрог 2020**

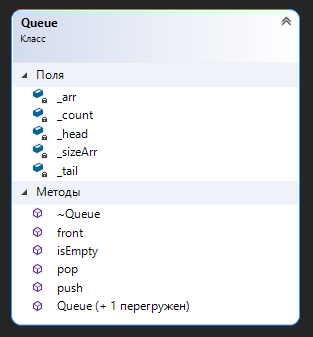
1. **Вариант задания №10**

Реализовать класс Очередь (Queue) для какого-либо типа данных с методами push, pop, isEmptyи front(показыва-ется первый элемент без его извлечения), работающими согласно соответствующей дисциплине обслуживания. Размер очереди задается при ее создании.

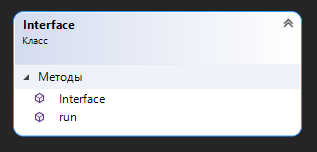
1. **UML-диаграмма класса Error**



1. **UML-диаграмма класса Queue**



1. **UML-диаграмма класса Interface**



1. **Листинг программы**

**main.cpp:**

#include "Interface.h"

#include <vld.h>

int main() {

Interface interface;

interface.run();

return 0;

}

**Error.h:**

#pragma once

#include <string>

class Error: public std::exception {

public:

Error(const std::string& value);

const char\* what() const override;

private:

std::string \_data;

};

**Error.cpp:**

#include "Error.h"

Error::Error(const std::string& value): \_data(value) {}

const char\* Error::what() const {

return \_data.c\_str();

}

**Interface.h:**

#pragma once

#include <iostream>

#include <clocale>

#include "Queue.h"

#include "Error.h"

class Interface {

public:

Interface();

void run();

};

**Interface.cpp:**

#include "Interface.h"

Interface::Interface() = default;

void Interface::run() {

setlocale(LC\_ALL, "Russian");

int sizeList;

do {

std::cout << "Сколько элементов в очереди?: ";

std::cin >> sizeList;

} while (sizeList <= 0);

Queue queue(sizeList);

while (true) {

int check;

std::cout << "1. Вывести последнее число\n2. Существует ли элемент в очереди" << std::endl;

std::cout << "3. Удалить первый элемент из очереди\n4. Добавить элемент\n5. Выход" << std::endl;

std::cin >> check;

try {

switch (check) {

case 1:

std::cout << "Ваше число: " << queue.front() << std::endl;

break;

case 2:

if (!queue.isEmpty()) {

std::cout << "Очередь не пустая" << std::endl;

} else {

std::cout << "Очередь пустая" << std::endl;

}

break;

case 3:

queue.pop();

break;

case 4:

int Element;

std::cin >> Element;

queue.push(Element);

break;

case 5:

return;

}

} catch (const std::exception& exept) {

std::cout << exept.what() << std::endl;

}

}

}

**Queue.h:**

#pragma once

class Queue {

public:

Queue();

Queue(const int& size);

~Queue();

void push(const int& data);

void pop();

bool isEmpty() const;

int front() const;

private:

int\* \_arr = nullptr;

int \_sizeArr;

int \_head = 0;

int \_tail = -1;

int \_count = 0;

};

**Queue.cpp:**

#include "Queue.h"

#include "Error.h"

Queue::Queue() = default;

Queue::Queue(const int& size): \_sizeArr(size), \_arr(new int[\_sizeArr]) {}

Queue::~Queue() {

delete[] \_arr; // Очистка памяти

}

void Queue::push(const int& data) {

if (\_count != \_sizeArr) { // Чтобы новый элемент не выходил за размеры очереди

if (\_tail == \_sizeArr - 1) { // Когда Хвост в конце очереди

\_tail = -1;

}

\*(\_arr + (++\_tail)) = data; // Добавление элемента в Хвост

\_count++; // Кол-во элементов в очереди

} else {

throw Error("Очередь переполнена"); // Ошибкинс

}

}

void Queue::pop() {

if (\_count > 0) { // Существует ли элемент в очереди

\_head++; // Сдвиг Головы ( типо удаление )

if (\_head == \_sizeArr) { // Когда Голова в конце массива

\_head = 0;

}

\_count--; // Уменьшение кол-ва элементов в очереди

} else {

throw Error("Нельзя удалить элемент - очередь пустая"); // Ошибкинс

}

}

bool Queue::isEmpty() const {

if (\_count > 0) { // Существует ли элемент в очереди

} else {

throw Error("Очередь пустая"); // Ошибкинс

}

}

int Queue::front() const {

if (\_count > 0) { // Существует ли элемент в очереди

return \*(\_arr + \_head); // Возращение Головы ( сам элемент )

} else {

throw Error("В очереди нет элементов"); // Ошибкинс

}

}