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 № 2**

по курсу «Объектно-ориентированное программирование»

**«Наследование в C++»**

**15 ВАРИАНТ**

Выполнил:

студенты гр. КТбо2-1

Самардак А.В.

Проверил:

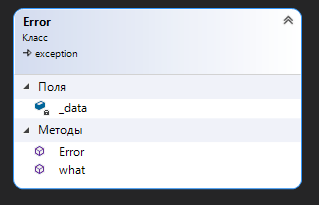
Тарасов С.А.

**Таганрог 2020**

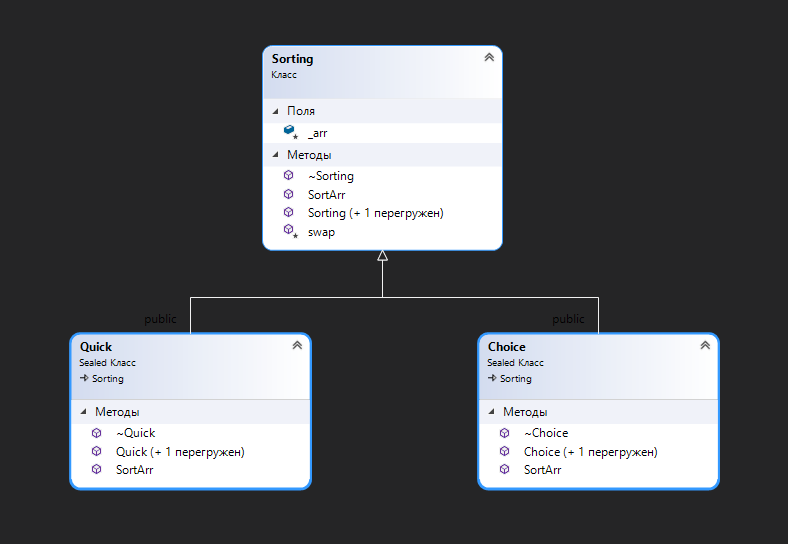
1. **Вариант задания №15**

Создать класс Sorting(сортировка), и на его основе классы Choice(сортировка выбором) и Quick(быстрая сорти-ровка). Размер сортируемых массивов задается при их создании, а элементы считываются из файла.

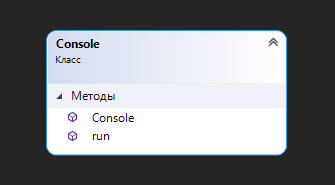
1. **UML-диаграмма класса Error**



1. **UML-диаграмма классов: Sorting, Quick, Choice**



1. **UML-диаграмма класса Console**



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();

}

**Console.h:**

#pragma once

#include <iostream>

#include <clocale>

#include "Sorting.h"

#include "Error.h"

class Console {

public:

Console();

void run();

};

**Console.cpp:**

#include "Console.h"

Console::Console() = default;

void Console::run() {

setlocale(LC\_ALL, "Russian");

std::string str;

int check, size;

do {

std::cout << "Какую сортировку выбрать?:\n 0. Choice\n 1. Quick" << std::endl;

std::cin >> check;

} while (check < 0 || check > 1);

std::cout << "Название файла: " << std::endl;

std::cin >> str;

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

std::cin >> size;

try {

Sorting\* sort = nullptr;

if (!check) {

sort = new Choice(size, str);

} else {

sort = new Quick(size, str);

}

sort->SortArr(size);

delete sort;

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

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

}

}

**Sorting.h:**

#pragma once

#include <fstream>

#include "Error.h"

class Sorting {

public:

Sorting() = default;

Sorting(const int size, const std::string& name\_file);

virtual ~Sorting();

virtual void SortArr(int n) = 0;

protected:

int\* \_arr;

void swap(int i, int j);

};

class Choice final: public Sorting {

public:

Choice() = default;

Choice(const int size, const std::string& name\_file);

~Choice();

void SortArr(int n);

};

class Quick final: public Sorting {

public:

Quick() = default;

Quick(const int size, const std::string& name\_file);

~Quick();

void SortArr(int n);

};

**Sorting.cpp:**

#include "Sorting.h"

Sorting::Sorting(const int size, const std::string& name\_file): \_arr(new int[size]) {

std::ifstream fin(name\_file + ".txt");

if (!fin.is\_open()) {

throw Error("Нет такого файла...");

} else {

for (int i = 0; i < size; ++i) {

fin >> \_arr[i];

}

}

fin.close();

}

Sorting::~Sorting() {

delete[] \_arr;

}

void Sorting::swap(int i, int j) {

int a = \_arr[i];

\_arr[i] = \_arr[j];

\_arr[j] = a;

}

//-------------

Choice::Choice(const int size, const std::string& name\_file): Sorting(size, name\_file) {}

Choice::~Choice() {}

void Choice::SortArr(int n) {

for (int i = 0; i < n - 1; ++i) {

int i\_min = i;

for (int j = i + 1; j < n; ++j) {

if (\_arr[j] < \_arr[i\_min]) {

i\_min = j;

}

}

swap(i, i\_min);

}

}

//-------------

Quick::Quick(const int size, const std::string& name\_file): Sorting(size, name\_file) {}

Quick::~Quick() {}

void Quick::SortArr(int n) {

int i = 0;

int j = n - 1;

int mid = \_arr[n / 2];

do {

while (\_arr[i] < mid) {

i++;

}

while (\_arr[j] > mid) {

j--;

}

if (i <= j) {

swap(i, j);

i++;

j--;

}

} while (i <= j);

if (j > 0) {

SortArr(j + 1);

}

if (i < n) {

SortArr(n - i);

}

}