**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**

Отчет по лабораторной работе №2

по курсу «Объектно ориентированное программирование»

«Наследование в С++»

Выполнил: студент гр. КТбо2-6

Калитин А.В.

Проверил:

Тарасов С.А.

Таганрог 2020

**Оглавление**

[Техническое задание 3](#_TOC_250003)

[Выполнение задания 3](#_TOC_250002)

1. [Спецификация классов 3](#_TOC_250001)
2. [Диаграмма классов 4](#_TOC_250000)
3. Листинг 5

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**Вариант № 15**

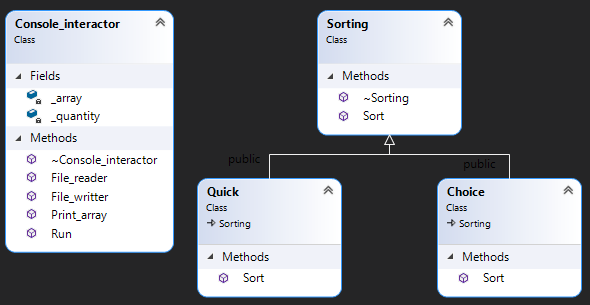
# Создать класс Sorting (сортировка), и на его основе классы Choice (сортировка выбором) и Quick (быстрая сортировка). Размер сортируемых массивов задается при их создании, а элементы считываются из файла.

# ВЫПОЛНЕНИЕ ЗАДАНИЯ

# Спецификация классов.

Класс **Console\_interactor** обеспечивает консольное взаимодействие с пользователем и хранит в себе массив для сортировки. Также он может считать массив из файла и записать отсортированный в другой файл с названием, которое задаст пользователь. Абстрактный класс **Sorting** необходим для наследования от него разных видов сортировок. В классах наследниках **Quick** и **Choice** переопределен метод **Sort** и реализованы разные виды сортировок, в первом – быстрая, во втором - сортировка выбором.

# Диаграмма классов.



* 1. **Листинг**

**Main.cpp**

#define \_CRTDBG\_MAP\_ALLOC

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <crtdbg.h>

#include <iostream>

#include "Console\_interactor.h"

#include <time.h>

//15. Создать класс Sorting(сортировка), и на его основе

//классы Choice(сортировка выбором) и Quick(быстрая сортировка).Размер сортируемых массивов задается при их создании,

//а элементы считываются из файла

int main()

{

\_CrtSetDbgFlag(\_CRTDBG\_ALLOC\_MEM\_DF | \_CRTDBG\_LEAK\_CHECK\_DF);

Console\_interactor console;

console.Run();

}

Choise.h

#pragma once

#include "Sorting.h"

class Choice:public Sorting

{

public:

int\* Sort(int n, int\* array) override;

};

**Console\_interactor.h**

#pragma once

#include <string>

#include <iostream>

#include <exception>

#include <fstream>

class Console\_interactor

{

public:

~Console\_interactor();

void Print\_array(int\* array);

void Run();

void File\_reader(const std::string& path);

void File\_writter(const std::string& path);

private:

long long int \_quantity;

int\* \_array=nullptr;

};

**Console\_interactor.cpp**

#include "Console\_interactor.h"

#include "Choice.h"

#include "Quick.h"

#include <exception>

#include <algorithm>

#include <iostream>

#include <fstream>

#include <string>

#include <exception>

Console\_interactor::~Console\_interactor()

{

if (\_array != nullptr)

{

delete[] \_array;

}

}

void Console\_interactor::Print\_array(int\* array)

{

for (long long int i = 0; i < \_quantity; i++)

{

std::cout << array[i] << '\n';

}

}

void Console\_interactor::File\_reader(const std::string& path)

{

std::ifstream file;

file.open(path);

long long int i;

if (file.is\_open())

{

file >> \_quantity;

\_array = new int[\_quantity];

for (i=0; i < \_quantity; i++)

{

file>>\_array[i];

}

}

else

{

file.close();

throw std::exception("File not found.");

}

}

void Console\_interactor::File\_writter(const std::string& path)

{

std::ofstream output\_file;

output\_file.open(path,std::ios::app);

if (output\_file.is\_open())

{

std::cout << "File " << path << " created\n";

for (long long int i = 0; i < \_quantity; i++)

{

output\_file << \_array[i] << '\n';

}

}

else

{

output\_file.close();

throw std::exception("File not created");

}

output\_file.close();

}

void Console\_interactor::Run()

{

std::cout << "enter the number of items(10 or 100 or 1000 or 1000000):\n";

std::string path;

std::cin >> path;

path += ".txt";

try

{

File\_reader(path);

}

catch (const std::exception& err)

{

std::cerr << "Error: " << err.what();

exit(1);

}

int command = 1;

std::cout << "1 - Choice sort\n2 - Qiuck sort\n3 - Create a file with a sorted array(the array will be sorted forcibly)\n4 - Print array\n5 - Shuffle array\n0 - Exit.\n";

while (command)

{

std::cout << "Enter a command: ";

std::cin >> command;

switch (command)

{

case 0:

break;

case 1:

{

Choice obj;

obj.Sort(\_quantity, \_array);

std::cout << "Success\n";

}

break;

case 2:

{

Quick obj;

obj.Sort(\_quantity, \_array);

std::cout << "Success\n";

}

break;

case 3:

{

Quick obj;

obj.Sort(\_quantity, \_array);

std::cout << "Enter a new file name:\n";

std::cin >> path;

path += ".txt";

try

{

File\_writter(path);

}

catch (const std::exception& err)

{

std::cerr << "Error: " << err.what();

exit(1);

}

}

break;

case 4:

Print\_array(\_array);

break;

case 5:

std::random\_shuffle(\_array, \_array + \_quantity);

break;

default:

std::cout << "Invalid command. Try again.\n";

break;

}

}

}

**Choise**.**cpp**

#include "Choice.h"

int\* Choice::Sort(int n, int\* array)

{

int i, j;

int count, key;

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

{

count = array[i]; key = i;

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

if (array[j] < array[key]) key = j;

if (key != i)

{

array[i] = array[key];

array[key] = count;

}

}

return array;

}

**Quick**.**h**

#pragma once

#include "Sorting.h"

class Quick:public Sorting

{

public:

int\* Sort(int n, int\* array) override;

};

**Quick**.**cpp**

#include "Quick.h"

int\* Quick::Sort(int size, int\* array)

{

int i = 0;

int j = size - 1;

int mid = array[size / 2];

do

{

while (array[i] < mid)

{

i++;

}

while (array[j] > mid)

{

j--;

}

if (i <= j)

{

int tmp = array[i];

array[i] = array[j];

array[j] = tmp;

i++;

j--;

}

} while (i <= j);

if (j > 0)

{

Quick::Sort(j + 1,array);

}

if (i < size)

{

Quick::Sort(size - i, &array[i]);

}

return array;

}

**Sorting**.**h**

#pragma once

class Sorting

{

public:

virtual ~Sorting() {};

virtual int\* Sort(int n, int\* array)=0;

};