КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

АЛГОРИТМЫ СОРТИРОВКИ

по курсу: Структуры и алгоритмы обработки данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

1.1 Цель работы

Целью работы является изучение алгоритмов внутренней сортировки и получение практических навыков их использования, и анализа их сложности.

1.2 Задание на лабораторную работу

Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем

Вариант 4

4	————//————	Простым извлечением

Найти количество различных чисел среди элементов массива

пространственая сложность

v=n*Cint+2*Cdouble+Cint

теоретическая пространственая сложность

V(n)=O(n)=O(max(O(n*Cint),O(2*Cdouble),O(Cint)))=O(max(O(n),O(1),O(1)))=O(n)

Временная сложность.

Tsort=O(max(O(K1),K(n^2*K2)))=O(max(O(1),O(n^2)))=O(n^2)

Листинг

main.cpp

```
#include <iostream>
using namespace std;

#include "libs/lib.h"
#include <cmath>
#include <time.h>
#include <iomanip>

// проверка ввода
#include "libs/simple_char.h"
#include "libs/input_validation.h"

#include "libs/array.h"

#include "sort.h"

int menu() {
```

```
    int id;
    while (true) {
        draw_line(30);
        cout << "1) добавить элемент в массив" << endl;
        cout << "2) удалить элемент из массива" << endl;
        cout << "3) вывести массив" << endl;
        cout << "4) сортировать массив" << endl;
        cout << "5) Найти количество различных элементов" << endl;
        cout << "0) Выход" << endl;
        id = read_value(" >>> ", false, false, false);
        if (id >= 0 && id <= 5) {
            return id;
        } else {
            cout << "Этого нет в меню" << endl;
        }
    }
}

int main() {
    // смена кодировки
    system("chcp 65001");
    draw_line();

    int size = read_value("Размер массива: ", false, false, false);

    Array array(size, true, true);

    int menu_i;
    while (true) {
        menu_i = menu();

        switch (menu_i) {
            case (0):
                return 0;
                break;

            case (1):
                array.add(read_value("Добавляемый элемент: ", false, false, false));
                break;

            case (2): {
                int returned = array.pop(read_value("Id удаляемого элемена: ", false, false,
false));
                cout << "Удалённый элемент: " << returned << endl;
                break;
            }

            case (3):
                array.draw("Массив: ");
                break;

            case (4):
```

```
            array.sort_arr();
            break;

        case (5):
            cout << "Количество различных элементов: " << array.dif_els() << endl;
            break;
      }

   }
   return 0;
}
```

sort.cpp

```
#include <iostream>
using namespace std;

#define rand_min -10
#define rand_max 10

class Array {
public:
   Array(int, bool, bool);
   ~Array();

   int* sort_arr();
   int* generator();
   void add(int);
   int pop(int);
   void draw(const char* promt);

   int find_el(int);
   int dif_els();
private:
   int* arr;

   int size;
   bool auto_sort;
   bool random;

};

// конструктор
Array::Array(int Size, bool Auto_sort = true, bool Random = true) {
   size = Size;
   auto_sort = Auto_sort;
   random = Random;

   arr = (int*)malloc(size * sizeof(int));
   if (random) {
      generator();
   } else {
```

```
        for (int i = 0; i < size; i++) {
            cout << "Array[" << i << "] = ";
            arr[i] = read_value("", false, false, false);
        }
    }

    draw("Изначальный массив: ");

    if (auto_sort) sort_arr();
}

// деструктор
Array::~Array() {
    free(arr);
}

// вывод массива
void Array::draw(const char* promt = "") {
    cout << promt;
    for (int i = 0; i < size; i++) cout << arr[i] << " ";
    cout << endl;
}

// добавление элемента в массив
void Array::add(int x) {
    arr = (int*)realloc(arr, ++size * sizeof(int));

    arr[size - 1] = x;

    if (auto_sort) sort_arr();
}

int Array::find_el(int x) {
    // for (int i = 0; i < size; i++) {
    //   if (arr[i] == x) return i;
    // }
    // return -1;
    if (x > 0 && x < size)
        return arr[x];
    return 0;
}

int Array::dif_els(){
    int count = 1;
    int tmp = 0;
    for (int j = 1; j < size; j++) {
        for (int k = 0; k < j; k++) {
            if (arr[j] != arr[k]) {
                tmp = 1;
            } else {
                tmp = 0;
                break;
```

```
            }

        }
        count += tmp;
    }
    return count;
}

// удаление элемента в массиве
int Array::pop(int x) {

    int returned = arr[x];
    int* old_arr = (int*)malloc(size * sizeof(int));
    old_arr = arr;
    arr = (int*)malloc((size - 1) * sizeof(int));

    int i, j = 0;
    for (i = 0; i < size; i++) {
        if (i != x) arr[j++] = old_arr[i];
    }

    size--;
    free(old_arr);

    if (auto_sort) sort_arr();

    return returned;
}

// заполнение массива случайными числами
int* Array::generator() {
    for (int i = 0; i < size; i++) {
        arr[i] = random_int(rand_min, rand_max);
    }
    return arr;
}

// сортировка Простым извлечением
int* Array::sort_arr() {
    int i,y;
    for(i=size-1; i>0; i--)
    {
        int max=0;
        for (int y=1; y<=i; y++)
        {
            if(arr[y]>arr[max]) max=y;
        }
        int temp = arr[i];
        arr[i] = arr[max];
        arr[max] = temp;
    }
    return arr;
```

```
}
```

Результат работы

Вывод

Я освоил алгоритмы внутренней сортировки и получил практические навыки их использования, и анализа их сложности.