ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М. А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «АЛГОРИТМЫ СОРТИРОВКИ» |
| по курсу: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Столяров Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

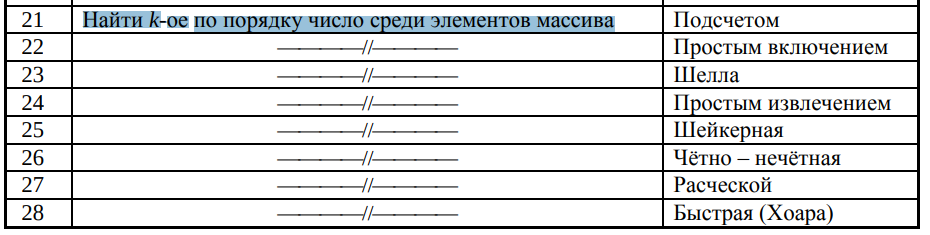
**1.1 Цель работы**

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 27**



**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 k-ое по порядку число среди элементов массива" << 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("Разсер массива: ", false, true, 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 << "Найденый элемент: " << array.find\_el(read\_value("какой элемент найти: ", false, true, false)) << 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);**

**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;**

**}**

**// 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 расчёской**

**int\* Array::sort\_arr() {**

**int if\_count = 0; // количество сравнений**

**int swap\_count = 0; // количество перестановок**

**const double factor = 1.247; // Фактор уменьшения**

**double step = size - 1;**

**while (step >= 1) {**

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

**if\_count++;**

**if (arr[i] > arr[(int)(i + step)]) {**

**swap\_count++;**

**swap(arr[i], arr[(int)(i + step)]);**

**}**

**}**

**step /= factor;**

**}**

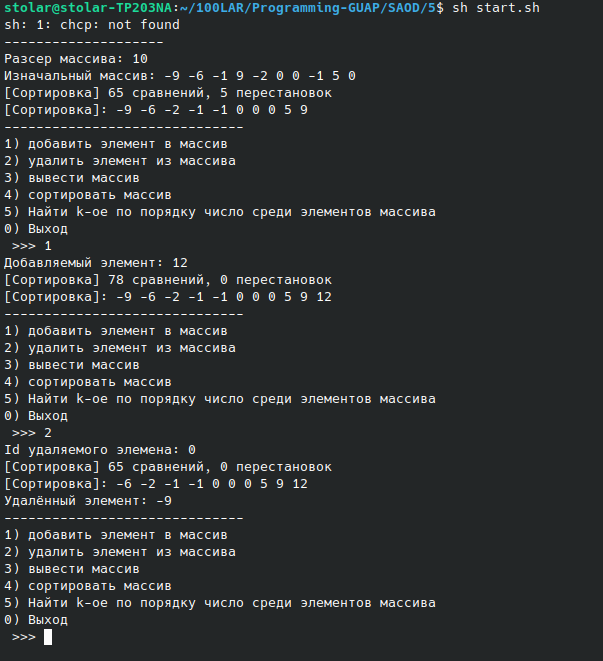
**cout << "[Сортировка] " << if\_count << " сравнений, " << swap\_count << " перестановок" << endl;**

**draw("[Сортировка]: ");**

**return arr;**

**}**

**Результат работы**



**Вывод**

Я освоил методы сортировки­.