ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

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

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  |  |  | В.А. Алексеев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

* 1. Цель работы

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

* 1. Задание на лабораторную работу

Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем. Дополнительно в программе должны быть реализованы следующие функции:

1) Поиск элемента либо по его порядковой позиции, либо по его содержимому;

2) Добавление/удаление элемента с последующей пересортировкой последовательности;

3) В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

Вариант: 14

Задание: Найти количество повторяющихся чисел среди элементов массива

Алгоритм сортировки: Простым извлечением

* 1. Листинг

#include <iostream>

#include <vector>

using namespace std;

void found(vector<int>& a, int n) {

int q, k;

cout << "Поиск элемента.\nЧтобы искать по порядковой позиции введите 1.\nЧтобы искать по содержимому нажмите 2\n";

cin >> q;

if (q == 1) {

cout << "Введите порядковую позицию : ";

cin >> k;

if (k > n - 1) {

cout << "Выход за границу\n";

return;

}

cout << "Порядковая позиция : " << k << " значение : " << a[k] << endl;

return;

}

else if (q == 2) {

cout << "Введите значение : ";

cin >> k;

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

if (a[i] == k) {

cout << "Порядковая позиция : " << i << " значение : " << k << endl;

return;

}

}

cout << "Элемент не найден.\n";

}

else {

cout << "Error\n";

return;

}

}

void show(vector<int>& a) {

for (int i = 0; i < a.size(); i++)

cout << a[i] << endl;

}

void foundu(vector<int>& a) {

int f = 0;

int stec = a[0] - 1;

for (int i = 0; i < a.size() - 1; i++) {

if (a[i] == a[i + 1] && a[i] != stec) {

f++;

stec = a[i];

if (i == a.size() - 1)

f++;

}

}

cout << "Повтор : " << f << endl;

}

void sort(vector<int>& a, int n) {

int min, x, per = 0, sr = 0;

int indx;

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

indx = i;

min = a[i];

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

sr++;

if (a[j] < min) {

min = a[j];

indx = j;

}

}

if (i != indx) {

swap(a[i], a[indx]);

per++;

}

}

cout << "Сортировка закончена.\n";

cout << "Cравнений : " << sr << endl << "Перестановок : " << per << endl;

}

void main() {

setlocale(LC\_ALL, "rus");

cout << "Введите количество элементов массива : ";

int n;

cin >> n;

vector<int> a = {};

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

int ar;

cout << i + 1 << " элемент : ";

cin >> ar;

a.push\_back(ar);

}

int k = 5;

while (k != 0) {

switch (k)

{

case 1: {

found(a, a.size());

break;

}case 2: {

cout << "Добавление элемента.\nВведите значение : ";

int neww;

cin >> neww;

a.push\_back(neww);

sort(a, a.size());

show(a);

break;

}case 3: {

cout << "Удаление элемента.\nВведите порядковую позицию : ";

int del;

cin >> del;

if (del > a.size() || del < 0) {

cout << "Выход за границу\n";

break;

}

a.erase(a.begin() + del);

sort(a, a.size());

show(a);

break;

}case 4: {

cout << "Сортировка.\n";

sort(a, a.size());

break;

}case 5: {

cout << "Меню\n";

cout << "1 - поиск\n";

cout << "2 - Добавление элемента.\n";

cout << "3 - Удаление элемента.\n";

cout << "4 - Сортировка.\n";

cout << "5 - Меню\n";

cout << "0 - выход\n";

}case 6: {

show(a);

break;

}case 7: {

sort(a, a.size());

foundu(a);

break;

}

}

cout << "команда : ";

cin >> k;

}

}

* 1. Сложность алгоритма

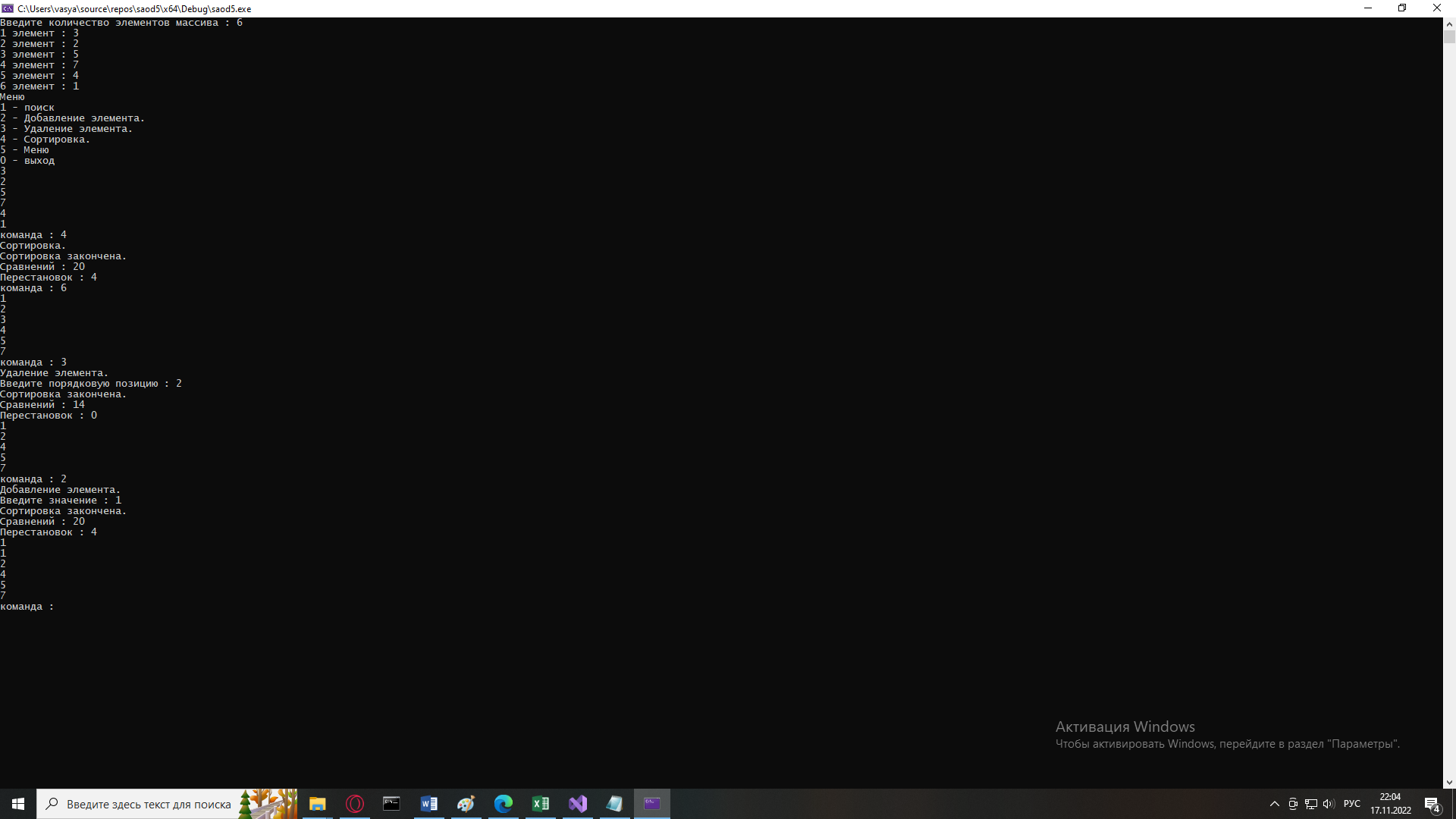
С - константа, характеризующая объем памяти, отводимый под переменную целого типа.

v = n\*Cint + 4\*Cint

V(n) = O(v) = O(max(O(n\*Cint), O(4\*Cint))) = O(max(O(n), O(1))) = O(n)

TSort = O(max(O(K1), O( \*K2))) = O(max(O(1), O())) = O()

* 1. Работа программы



* 1. Вывод

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