ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Мурашова М.А. |
| 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. № | 4134K |  |  |  | Опарин С.Н |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Лабораторная работа №5: «АЛГОРИТМЫ СОРТИРОВКИ»**

**Цель работы:**

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

**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. 1)  Поиск элемента либо по его порядковой позиции, либо по его содержимому;
2. 2)  Добавление/удаление элемента с последующей пересортировкой последовательности;
3. 3)  В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

**Вариант задания:**



**Листинг программы:**

#include <iostream>

using namespace std;

int \*masFill(int size){

int \*mas = new int[size];

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

mas[i] = rand() % 10;

}

cout << endl;

return mas;

}

void showMas(int \*mas, int size){

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

cout << mas[i] << " ";

}

cout << endl;

}

void findByIndex(int \*mas, int size, int index){

bool isFound = false;

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

if (i == index){

cout << "Значение под индексом " << index << " = "<< mas[i];

isFound = true;

break;

}

}

if (isFound == false){

cout << "Элемента с данным индексом не существует" << endl;

}

cout << endl;

}

void findByValue(int \*mas, int size, int value){

bool isFound = false;

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

if (mas[i] == value){

cout << "Элемент со значением " << value << " хранится в массиве в ячейке под индексом "<< i << endl;

isFound = true;

break;

}

}

if (isFound == false){

cout << "Элемента с данным значением не существует" << endl;

}

cout << endl;

}

int \*insertElement(int \*mas, int size, int value){

int \*temp = new int[size];

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

temp[i] = mas[i];

}

temp[size - 1] = value;

delete [] mas;

return temp;

}

int \*deleteElement(int \*mas, int size, int index){

int \*temp = new int[size];

int sizeNew = 0;

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

if(i == index){

continue;

}

else{

temp[sizeNew] = mas[i];

sizeNew += 1;

}

}

delete [] mas;

return temp;

}

void InsertSort (int \*mas, int size, int comp, int resh) {

comp = resh = 0;

for(int i = 1; i < size; i++)

for(int j = i; j > 0; j--){

comp += 1;

if(mas[j - 1] > mas[j]){

resh += 1;

swap(mas[j - 1],mas[j]);

}

}

cout << "Количество сравнений = " << comp << endl;

cout << "Количество перестановок = " << resh << endl;

}

void countDiffNums(int \*mas, int size){

int count = 0;

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

if(mas[i] != mas[i + 1]){

count += 1;

}

else{

continue;

}

}

cout << "Количество различных чисел в массиве = " << count << endl;

}

int main(){

int comparisons = 0, reshuffles = 0;

int size, task, value, index;

int \*arr;

cout << "Введите размер массива: ";

cin >> size;

while(size <= 0){

cout << "Введите размер массива больше 0: ";

cin >> size;

}

while(true){

cout << "--------------------------------------------------------------" << endl;

cout << "1 - Заполнить массив значениями" << endl;

cout << "2 - Вывести массив" << endl;

cout << "3 - Отсортировать массив методом простых включений" << endl;

cout << "4 - Добавить элемент" << endl;

cout << "5 - Удалить элемент" << endl;

cout << "6 - Найти элемент по индексу" << endl;

cout << "7 - Найти элемент по значению" << endl;

cout << "8 - Подсчет количества разных чисел в массиве" << endl;

cout << "9 - Выход" << endl;

cout << "--------------------------------------------------------------" << endl;

cout << "Ваш выбор: ";

cin >> task;

switch (task) {

case 1:

arr = masFill(size);

cout << "Массив успешно заполнен случайными значениями" << endl;

break;

case 2:

cout << "Текущий массив" << endl;

showMas(arr, size);

break;

case 3:

InsertSort(arr, size, comparisons, reshuffles);

cout << "Массив успешно отсортирован" << endl;

break;

case 4:

cout << "Введите значение, которое вы хотите добавить: ";

cin >> value;

size += 1;

arr = insertElement(arr, size, value);

cout << "Элемент успешно добавлен в массив" << endl;

InsertSort(arr, size, comparisons, reshuffles);

cout << "Массив успешно отсортирован" << endl;

break;

case 5:

cout << "Введите индекс значения, которое вы хотите удалить: ";

cin >> index;

size -= 1;

arr = deleteElement(arr, size, index);

cout << "Элемент успешно удален из массива" << endl;

InsertSort(arr, size, comparisons, reshuffles);

cout << "Массив успешно отсортирован" << endl;

break;

case 6:

cout << "Введите индекс элемента, который вы хотите найти: ";

cin >> index;

findByIndex(arr, size, index);

break;

case 7:

cout << "Введите значение, которое вы хотите найти: ";

cin >> value;

findByValue(arr, size, value);

break;

case 8:

InsertSort(arr, size, comparisons, reshuffles);

cout << endl;

countDiffNums(arr, size);

break;

case 9:

exit(0);

break;

default:

break;

}

}

return 0;

}

**Расчет сложности алгоритма:**

Разработанный алгоритм использует следующие данные:

- один массив размерностью n = sizeof(mas) / sizeof(int);

- три входных переменных целого типа.

Значит, пространственная сложность алгоритма определяется следующим образом:

1. ,

где – константа, характеризующая объем памяти, отводимый под вещественную переменную размера 8 байт целого типа.

Теоретическая пространственная сложность алгоритма составляет:

,

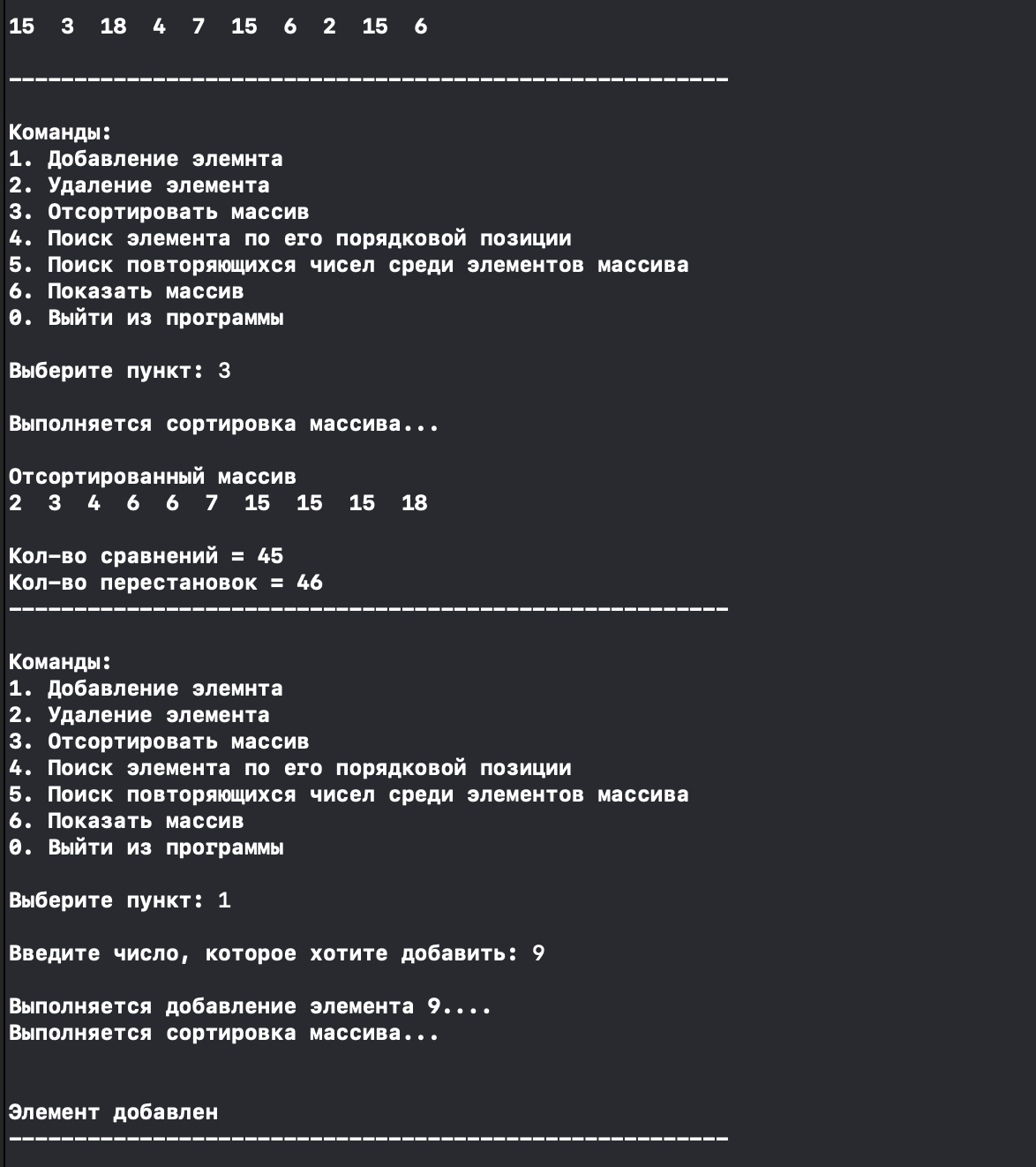
Теоретическую временную сложность алгоритма определяем на основе анализа текста программы, реализующей данный алгоритм.

Рассчитаем теоретическую временную основного алгоритма сортировки за одну рекурсию :

Где – операции сравнения, присваивания, используемые в алгоритме и имеющие временную сложность «1», функции сортировки массива.

1. Теоретическая временная сложность функций составляет:

На основе этих примеров можно сделать вывод, что была разработана программа, характеристики которой соответствуют поставленному заданию.

**Результаты работы программы: **

**Вывод:**

В ходе работы была изучена шейкерная сортировка , а также получены практические навыки её реализации.