ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		К.А. Кочин
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. № 4136		Бобрович Н. С.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы

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

Задание

Вариант №11.

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.

Найти количество повторяющихся чисел среди элементов массива сортировкой подсчётом.

Листинг программы

```cpp
#include <iostream>

using namespace std;

int gcount = 0;
const int n = 10;

int r(int min, int max)
{
    int num = min + rand() % (max - min + 1);
    return num;
}

void f1(int a, int* arr, const int n)
{
    int count = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] == a) {
            cout << "Элемент равный " << arr[i] << " находится на позиции " <
            count++;
        }
    }
    if (count == 0) {
        cout << "Элемента с таким значением нет." << endl;
    }
}

void f2(int a, int* arr, const int n)
{
    int count = 0;
    for (int i = 0; i < n; i++) {
        if (i == a) {
            cout << "Элемент равный " << arr[i] << " находится на позиции " <<
            count++;
        }
    }
    if (count == 0) {
        cout << "Элемента с такой позицией нет." << endl;
    }
}

void sortch(int* arr, const int n)
{
    int i, j, k, count;
    count = 1;
    int* brr = new int[n];
    for (i = 0; i < n; i++) {
        k = 0;
        for (j = 0; j < n; j++) {
            if ((arr[j] < arr[i]) || ((arr[j] == arr[i]) && (j < i))) {
```

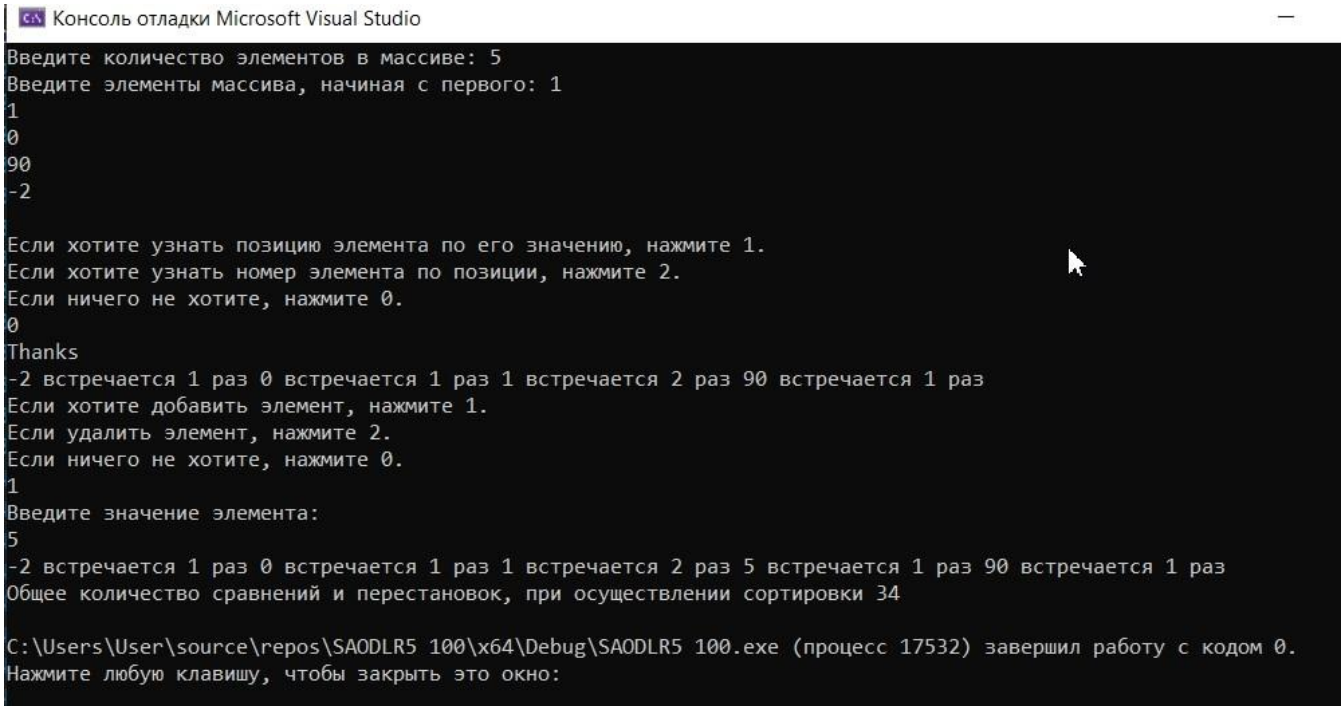
```
                k++;
                gcount++;
            }
        }
        brr[k] = arr[i];
    }
    for (i = 0; i < n; i++) {
        arr[i] = brr[i];
    }
    i = 1;
    while (i < n) {
        if (arr[i] == arr[i - 1]) {
            count++;
            i++;
            gcount++;
        }
        else {
            cout << arr[i - 1] << " встречается " << count << " раз ";
            count = 1;
            i++;
            gcount++;
        }
    }
    cout << arr[i - 1] << " встречается " << count << " раз";
}
```

```
int main()
{
    setlocale(LC_ALL, "Rus");
    srand(time(NULL));

    int n, m, k, l, i, x;
    cout << "Введите количество элементов в массиве: ";
    cin >> n;
    int* a = new int[n];
    cout << "Введите элементы массива, начиная с первого: ";
    for (i = 0; i < n; i++) {
        cin >> x;
        a[i] = x;
    }

    cout << endl;
    cout << "Если хотите узнать позицию элемента по номеру, нажмите 1.\nЕсли
    cin >> m;
    if ((m != 1) && (m != 2) && (m != 0)) {
        cout << "Error" << endl;
    }
    if (m == 1) {
        cin >> k;
        f1(k, a, n);
```

```
    }
    if (m == 2) {
        cin >> k;
        f2(k, a, n);
    }
    if (m == 0) {
        cout << "Thanks" << endl;
    }

    sortch(a, n);

    cout << endl;
    cout << "Если хотите добавить элемент, нажмите 1.\nЕсли удалить элемент, н
    cin >> m;
    if ((m != 1) && (m != 2) && (m != 0)) {
        cout << "Error" << endl;
    }
    if (m == 1) {
        n++;
        cout << "Введите значение элемента: " << endl;
        cin >> l;
            a[n - 1] = l;
            sortch(a, n);
    }
    if (m == 2) {
```

```
        cout << "Введите позицию элемента: " << endl;
        cin >> l;
        if (0 <= l < n) {
            for (i = l; i < n; i++) {
                a[i - 1] = a[i];
            }
            n--;
            sortch(a, n);
        }
        else {
            cout << "Не попадает в количество элементов!" << endl;
        }
    }
    if (m == 0) {
        cout << "Thanks" << endl;
    }

    cout << "Общее количество сравнений и перестановок, при осуществлении сор

    return 0;
}
```

Контрольный пример



Количество выполненных операций сравнения и перестановок элементов массива

Переменная gcount в данном контрольном примере равна 34.

Временную и пространственную сложность алгоритма

Разработанный алгоритм использует следующие данные:

- один массив размерностью n, один массив размерностью max - min;
- семь переменных целого типа.

Значит, пространственная сложность алгоритма определяется следующим образом:
v = n*Cuint16_t + 7*Cuint16_t, где Cuint16_t – константа, характеризующая объем памяти, отводимый под переменную беззнакового 16-тиразрядного целого типа.
Теоретическая пространственная сложность алгоритма составляет:

V(n) = O(v) = O(max(O(n*Cuint16_t), O(7*Cuint16_t)
Теоретическую временную сложность алгоритма определяем на основе анализа текста программы, реализующей данный алгоритм. Для начала рассчитаем теоретическую временную алгоритма сортировки:
TSort = O(max(O(K1), O(*K2))) = O(max(O(1), O( ))) = O(n), где K1 – операции сравнения, присваивания, используемые в алгоритме и имеющие временную сложность «1», K2 – цикличные операции, занимающие в наихудшем случае шагов.

Вывод

Написан код, отвечающий всем требованиям. В итоге были изучены алгоритмы внутренней сортировки и получены практические навыки их использования, и анализа их сложности.