

МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

СОРТИРОВКА С ПОМОЩЬЮ ПРЯМОГО ВКЛЮЧЕНИЯ
ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
направления 100501 — Компьютерная безопасность
факультета КНиИТ
Токарева Никиты Сергеевича

Проверил
доцент

А. Н. Гамова

СОДЕРЖАНИЕ

1	Описание алгоритма	3
2	Код программы, реализующий алгоритм	4
3	Анализ алгоритма	6
ЗАКЛЮЧЕНИЕ		7
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ		8

1 Описание алгоритма

Пусть имеется последовательность элементов a_0, a_1, \dots, a_{n-1} . Эту последовательность мысленно можно разделить на уже упорядоченную последовательность a_0, a_1, \dots, a_{i-1} и на исходную, неупорядоченную, $a_i, a_{i+1}, \dots, a_{n-1}$. Очевидно, что первоначально упорядоченная последовательность состоит из единственного элемента a_0 . При каждом шаге, начиная с $i = 1$ и увеличивая i каждый раз на единицу, из исходной последовательности извлекается i -й элемент a_i , и вставляется в готовую последовательность в нужное место, при необходимости, сдвигая элементы готовой последовательности на одну позицию вправо вплоть до i -й позиции. Поиск подходящего места осуществляется сравнением a_i с элементами a_j , где $j = i - 1, i - 2, \dots, 0$, и одновременным обменом местами a_i и a_j , если $a_i < a_j$, т.е. сдвигом a_j на одну позицию вправо.

Процесс поиска заканчивается при выполнении одного из условий:

- если $a_i \geq a_j$;
- если достигнут левый конец готовой последовательности.

2 Код программы, реализующий алгоритм

Далее представлена реализация сортировки с помощью прямого включения.

```
#include <iostream>
#include <vector>

using namespace std;

void straight_insertion_sort(vector <int>& a, int n) {
    int val, j;
    for (int i = 1; i < n; i++) {
        val = a[i];
        j = i - 1;
        while (j >= 0 && val < a[j]) {
            a[j + 1] = a[j];
            a[j] = val;
            j--;
        }
    }
}

void print_array (vector<int>& a, int n) {
    for (int i = 0; i < n; i++) {
        cout << a[i] << ' ';
    }
    cout << "\n";
}

int main() {
    cout << "Введите число элементов массива\n";
    int n;
    cin >> n;
    vector <int> a(n);

    cout << "Введите элементы массива\n";
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
}
```

```

cout << "Исходный массив:\n";

print_array(a, n);

cout << "Отсортированный массив:\n";

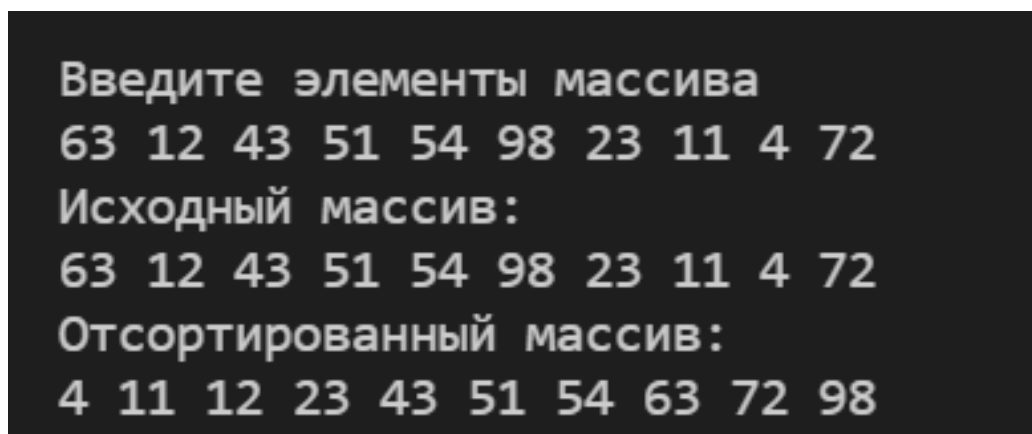
straight_insertion_sort(a, n);
print_array(a, n);

return 0;
}

```

Далее приведем несколько примеров, подав некоторые случайные числа в качестве входных данных, чтобы проверить корректность, написанного кода.

На рисунке 1 показан ввод 10 случайных чисел.



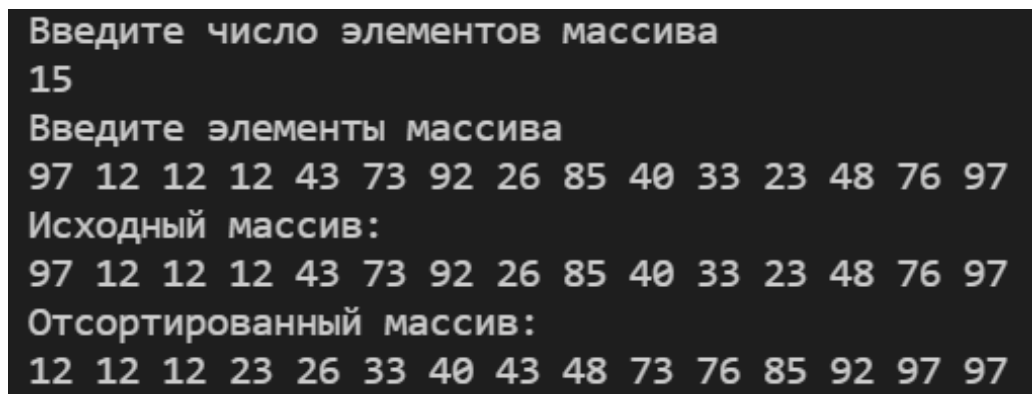
```

Введите элементы массива
63 12 43 51 54 98 23 11 4 72
Исходный массив:
63 12 43 51 54 98 23 11 4 72
Отсортированный массив:
4 11 12 23 43 51 54 63 72 98

```

Рисунок 1 – Ввод последовательности, состоящей из 10 чисел

На рисунке 2 показан ввод 15 случайных чисел. Отсюда видно, что всё работает корректно.



```

Введите число элементов массива
15
Введите элементы массива
97 12 12 12 43 73 92 26 85 40 33 23 48 76 97
Исходный массив:
97 12 12 12 43 73 92 26 85 40 33 23 48 76 97
Отсортированный массив:
12 12 12 23 26 33 40 43 48 73 76 85 92 97 97

```

Рисунок 2 – Ввод последовательности, состоящей из 15 чисел

3 Анализ алгоритма

Количество сравнений элементов (C_i) при i -м просеивании самое большее равно $i - 1$, самое меньшее – 1. Если предположить, что все перестановки из n элементов равновероятны, то среднее число сравнений – $i/2$. Число же пересылок (присваиваний элементов) M_i равно $C_i + 2$.

Поэтому общее число сравнений:

$$C_{min} = n - 1$$

$$C_{ave} = (n^2 + n - 2)/4$$

$$C_{max} = (n^2 + n - 4)/4$$

Общее число пересылок:

$$M_{min} = 3(n - 1)$$

$$M_{ave} = (n^2 + 9n - 10)/4$$

$$M_{max} = (n^2 + 3n - 4)/2$$

Минимальные оценки встречаются в случае уже упорядоченной исходной последовательности элементов, наихудшие оценки – когда они первоначально расположены в обратном порядке. В некотором смысле сортировка с помощью включений демонстрирует истинно естественное поведение. Ясно, что приведенный алгоритм описывает процесс устойчивой сортировке, так как порядок элементов с равными ключами при нем остается неизменным.

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе была рассмотрена сортировка с помощью прямого включения и был проведен анализ оценки сложности данного алгоритма. Это послужило созданием её программной реализации, написанной на языке C++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Книга Н. Вирт, "Алгоритмы и структуры данных"/ Москва: Мир, 1989 г., Яз. рус.
- 2 Книга Стивена С. Скиена, "Алгоритмы. Руководство по разработке 2-е издание"/ Санкт-Петербург: БХВ-Петербург, 2021 г., Яз. рус.