

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Сортировки

Студент гр. 9304

Атаманов С.Д.

Преподаватель

Филатов Ар. Ю.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с алгоритмами сортировки. Реализовать алгоритмы, используя язык программирования C++.

Задание.

Вариант 5.

Гибрид сортировки пузырьком и сортировки выбором; шейкерная сортировка.

Выполнение работы.

Описание алгоритма.

Гибрид сортировки пузырьком и вставками.

Алгоритм заключается в последовательном проходе массива слева-направо. Во время этого прохода происходит сравнение соседних элементов: если левый больше правого, то они меняются местами. Также происходит поиск минимального элемента. По достижении конца минимальный элемент меняется с первым местами. Таким образом к концу массива справа находится максимальный элемент, а слева минимальный. Массив отсортирован либо после завершения всех циклов прохода(худший случай), либо при условии, что во время прохода цикла не было смены элементов.

Шейкерная сортировка.

Алгоритм является модификацией пузырьковой сортировки. Происходит проход массива слева-направо, при котором максимальный элемент перемещается вправо, затем справа-налево, при котором минимальный элемент перемещается влево. Массив отсортирован после того, как не было выявлено смены элементов.

Формат входных и выходных данных.

Программа получает строку, содержащая целые числа. Между ними может быть любое кол-во пробелов. Числа могут быть отрицательными.

Считывание происходит из файла.

Программа выводит промежуточные и окончательные результаты в stdout.

Используемые структуры данных и реализованные функции.

Функция `printIntermediate()` - печатает массив с количеством шагов.

Функция `modBubble()` - реализует гибрид пузырьковой сортировки и сортировки вставками.

Получает ссылку на первый элемент массива и переменную счетчик, которая считает число проходов по массиву. Запускается двойной цикл `for`: первый цикл для последовательного прохода массива, второй также для прохода массива, но в нем выполняется основная сортировка. В первом цикле инициализируются значения: `minIndex` — содержащая номер ячейки массива с минимальным элементом (по умолчанию = 0) и булева переменная `sorted` (по умолчанию `true`). Начинается сравнение соседних элементов — если левый больше правого, то соседние элементы проверяются на наличие в них минимального элемента массива, затем меняются местами, тем временем `sorted` принимает значение `false`, так как произошел «свап» элементов, затем выводятся промежуточные результаты. Затем происходит контрольная проверка «свапнутого» влево элемента. Если массив отсортирован досрочно (`sorted == true`), то выполнение алгоритма прекращается. Результат выводится на экран.

Функция `cocktailSort()` - Выполняет шейкерную сортировку массива.

Инициализируются переменные, хранящие индекс начала и конца массива, булева переменная `done`, которая проверяет, отсортирован массив или нет. Далее запускается цикл `do — while`, который выполняется до тех пор, пока `done != true`. В начале цикла `done` инициализируется `true`, затем в циклах происходит вначале пузырьковая сортировка слева-направо, а во втором цикле уже справа-налево. После каждого «свапа» элементов происходит вывод промежуточного результата на экран. Когда `done == true`, цикл заканчивается, отсортированный массив выводится на экран.

Функция `isCorrect()` - проверяет строку на валидность.

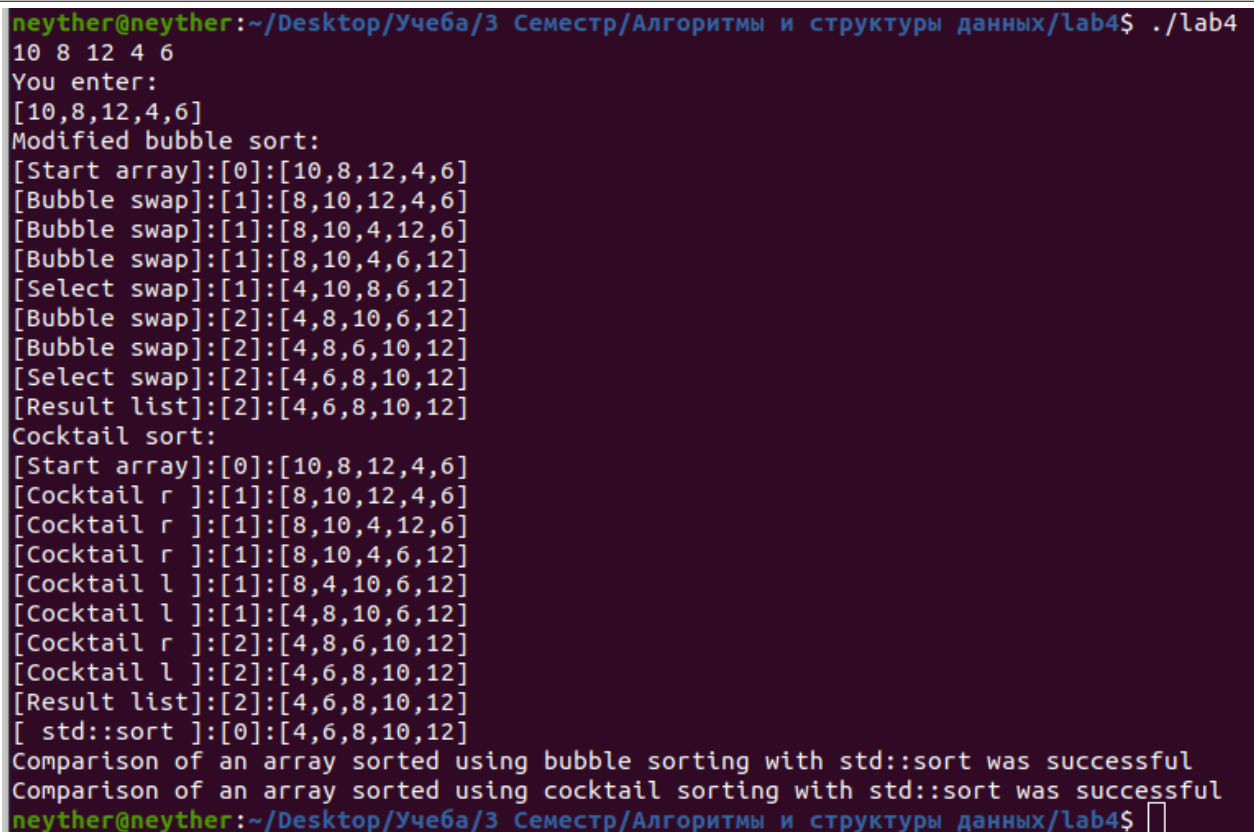
Получает итератор строки, который указывает на её начало. Происходит выполнение цикла while, пока итератор не равен концу строки. С помощью условных операций строка проверяется на валидность.

Функция printResult() - печатает финальный вывод программы в stdout.

Разработанный программный код смотри в приложении А.

Тестирование.

Для тестирования был написан bash-скрипт, который считывает тестовые данные из текстовых документов папки tests и передает их программе. В самой программе значения сравниваются с результатом сортировки с помощью библиотечной функции std::sort(). Данные о результатах тестирования выводятся в stdout.



```
neyther@neyther:~/Desktop/Учеба/3 Семестр/Алгоритмы и структуры данных/lab4$ ./lab4
10 8 12 4 6
You enter:
[10,8,12,4,6]
Modified bubble sort:
[Start array]:[0]:[10,8,12,4,6]
[Bubble swap]:[1]:[8,10,12,4,6]
[Bubble swap]:[1]:[8,10,4,12,6]
[Bubble swap]:[1]:[8,10,4,6,12]
[Select swap]:[1]:[4,10,8,6,12]
[Bubble swap]:[2]:[4,8,10,6,12]
[Bubble swap]:[2]:[4,8,6,10,12]
[Select swap]:[2]:[4,6,8,10,12]
[Result list]:[2]:[4,6,8,10,12]
Cocktail sort:
[Start array]:[0]:[10,8,12,4,6]
[Cocktail r ]:[1]:[8,10,12,4,6]
[Cocktail r ]:[1]:[8,10,4,12,6]
[Cocktail r ]:[1]:[8,10,4,6,12]
[Cocktail l ]:[1]:[8,4,10,6,12]
[Cocktail l ]:[1]:[4,8,10,6,12]
[Cocktail r ]:[2]:[4,8,6,10,12]
[Cocktail l ]:[2]:[4,6,8,10,12]
[Result list]:[2]:[4,6,8,10,12]
[ std::sort ]:[0]:[4,6,8,10,12]
Comparison of an array sorted using bubble sorting with std::sort was successful
Comparison of an array sorted using cocktail sorting with std::sort was successful
neyther@neyther:~/Desktop/Учеба/3 Семестр/Алгоритмы и структуры данных/lab4$
```

Рисунок 1 — Пример работы программы

Результаты тестирования смотри в приложении Б.

Выводы.

В ходе выполнения работы были получены навыки работы с алгоритмами сортировки: коктейльная сортировка и гибридной пузырьковой сортировки и сортировки вставками. Сложность обеих сортировок по памяти составляет $O(n)$. По времени: $O(n^2)$ — коктейльная сортировка и $O((n^2)/2)$ — гибридная пузырьковая сортировка и сортировки вставками.

Была разработана программа на языке C++, которая выполняет сортировку массива шейкерной и гибридной пузырьковой сортировки и сортировки вставками.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: main.cpp

```
#include <vector>
#include <iostream>
#include <cstdlib>
#include <sstream>
#include <algorithm>

template <typename base>
void printIntermediate(std::vector<base> futureString, int count){
    std::cout << "[" << count << "]" << ":" << "[";
    for(int i=0;i<futureString.size();i++) {
        if (i == futureString.size() - 1) {
            std::cout << futureString[i];
            break;
        }
        std::cout << futureString[i] << ",";
    }
    std::cout << "\n";
}

template <typename base>
void modBubble(std::vector<base>& first, int count = 0){
    base temp;
    int minIndex;
    bool sorted;
    std::cout << "[Start array]:";
    printIntermediate(first, count);
    for(int i=0;i<first.size();i++) {
        minIndex = i;
        sorted = true;
        count++;
        for (int j = i; j < first.size() - i - 1; j++) {
            if (first[j] > first[j + 1]) {
                if (first[j] < first[minIndex])
                    minIndex = j;
                std::swap(first[j], first[j + 1]);
                sorted = false;
                std::cout << "[Bubble swap]:";
                printIntermediate(first, count);
            }
            if (first[j] < first[minIndex])
                minIndex = j;
        }
        if(sorted) {
            std::cout << "[Result list]:";
            printIntermediate(first, --count);
            return;
        }
        std::cout << "[Select swap]:";
        std::swap<base>(first[i], first[minIndex]);
        printIntermediate(first, count);
    }
}

template <typename base>
void cocktailSort(std::vector<base>& first, int count = 0){
```

```

int begin = 0;
int end = first.size()-1;
bool done;
std::cout << "[Start array]:";
printIntermediate(first, count);
do{
    done = true;
    count++;
    for(int i=begin;i<end;i++){
        if(first[i] > first[i+1]) {
            std::swap(first[i], first[i + 1]);
            std::cout << "[Cocktail r ]:";
            printIntermediate(first, count);
            done = false;
        }
    }
    end--;
    for(int i=end;i>=begin;i--) {
        if (first[i + 1] < first[i]) {
            std::swap(first[i], first[i + 1]);
            std::cout << "[Cocktail l ]:";
            printIntermediate(first, count);
            done = false;
        }
    }
    begin++;
}while (!done);
std::cout << "[Result list]:";
printIntermediate(first, --count);
}

bool isCorrect(std::string::iterator iterator){
    int flag = 0;
    while(*iterator != '\0'){
        if(isdigit(*iterator)) {
            iterator++;
            flag = 0;
        }
        else if(*iterator == ' ' && flag == 0)
            iterator++;
        else if(*iterator == ' ' && flag == 1)
            return false;
        else if(*iterator == '-' && flag == 0) {
            iterator++;
            flag = 1;
        }
        else if(*iterator == '-' && flag == 1)
            return false;
        else
            return false;
    }
    return true;
}

template <typename base>
void printResult(std::vector<base> bubble, std::vector<base> cocktail, std::vector<base>
sort){
    std::cout << "[ std::sort ]:";
    std::cout << "[";
    for(int i=0;i<sort.size();i++){
        if(i == sort.size()-1){

```

```

        std::cout << sort[i];
        break;
    }
    std::cout << sort[i] << ",";
}
std::cout << "]\n";
if(std::equal(bubble.begin(), bubble.end(), sort.begin()))
    std::cout << "Comparison of an array sorted using bubble sorting with std::sort was
successful\n";
else
    std::cout << "Comparison of an array sorted using bubble sorting with std::sort failed\n";
if(std::equal(cocktail.begin(), cocktail.end(), cocktail.begin()))
    std::cout << "Comparison of an array sorted using cocktail sorting with std::sort was
successful\n";
else
    std::cout << "Comparison of an array sorted using cocktail sorting with std:: sort failed\n";
}

int main(){
    std::vector<int> numbers, numbersCopy, stdNumbers;
    std::string vectorString;
    std::getline(std::cin, vectorString);
    auto begin = vectorString.begin();
    if(!isCorrect(begin) || vectorString.empty()){
        std::cout << "Error:\nWrong string format\n";
        exit(EXIT_FAILURE);
    }
    std::stringstream getNum(vectorString);
    int extractNumber;
    while(getNum >> extractNumber)
        numbers.push_back(extractNumber);
    std::cout << "You enter:\n[";
    for(int i=0;i<numbers.size();i++){
        if(i == numbers.size()-1) {
            std::cout << numbers[i];
            break;
        }
        std::cout << numbers[i] << ",";
    }
    std::cout << "]\n";
    numbersCopy = numbers;
    stdNumbers = numbers;
    std::cout << "Modified bubble sort:\n";
    modBubble(numbers);
    std::cout << "Cocktail sort:\n";
    cocktailSort(numbersCopy);
    std::sort(stdNumbers.begin(), stdNumbers.end());
    printResult(numbers, numbersCopy, stdNumbers);
    return 0;
}

```


ПРИЛОЖЕНИЕ Б.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.

Входные данные	Выходные данные	Комментарий
1 2 3 4 5 6 7 8 9 10	You enter: [1,2,3,4,5,6,7,8,9,10] Modified bubble sort: [Start array]:[0]:[1,2,3,4,5,6,7,8,9,10] [Result list]:[0]:[1,2,3,4,5,6,7,8,9,10] Cocktail sort: [Start array]:[0]:[1,2,3,4,5,6,7,8,9,10] [Result list]:[0]:[1,2,3,4,5,6,7,8,9,10] [std::sort]:[0]:[1,2,3,4,5,6,7,8,9,10] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	Заранее отсортированны й массив
10 9 8 7 6 5 4 3 2 1	You enter: [10,9,8,7,6,5,4,3,2,1] Modified bubble sort: [Start array]:[0]:[10,9,8,7,6,5,4,3,2,1] [Bubble swap]:[1]:[9,10,8,7,6,5,4,3,2,1] [Bubble swap]:[1]:[9,8,10,7,6,5,4,3,2,1] [Bubble swap]:[1]:[9,8,7,10,6,5,4,3,2,1] [Bubble swap]:[1]:[9,8,7,6,10,5,4,3,2,1] [Bubble swap]:[1]:[9,8,7,6,5,10,4,3,2,1] [Bubble swap]:[1]:[9,8,7,6,5,4,10,3,2,1] [Bubble swap]:[1]:[9,8,7,6,5,4,3,10,2,1] [Bubble swap]:[1]:[9,8,7,6,5,4,3,2,10,1] [Bubble swap]:[1]:[9,8,7,6,5,4,3,2,1,10] [Select swap]:[1]:[1,8,7,6,5,4,3,2,9,10] [Bubble swap]:[2]:[1,7,8,6,5,4,3,2,9,10] [Bubble swap]:[2]:[1,7,6,8,5,4,3,2,9,10] [Bubble swap]:[2]:[1,7,6,5,8,4,3,2,9,10] [Bubble swap]:[2]:[1,7,6,5,4,8,3,2,9,10] [Bubble swap]:[2]:[1,7,6,5,4,3,8,2,9,10] [Bubble swap]:[2]:[1,7,6,5,4,3,2,8,9,10] [Select swap]:[2]:[1,2,6,5,4,3,7,8,9,10] [Bubble swap]:[3]:[1,2,5,6,4,3,7,8,9,10] [Bubble swap]:[3]:[1,2,5,4,6,3,7,8,9,10] [Bubble swap]:[3]:[1,2,5,4,3,6,7,8,9,10]	Элементы в порядке убывания

	[Select swap]:[3]:[1,2,3,4,5,6,7,8,9,10] [Result list]:[3]:[1,2,3,4,5,6,7,8,9,10] Cocktail sort: [Start array]:[0]:[10,9,8,7,6,5,4,3,2,1] [Cocktail r]:[1]:[9,10,8,7,6,5,4,3,2,1] [Cocktail r]:[1]:[9,8,10,7,6,5,4,3,2,1] [Cocktail r]:[1]:[9,8,7,10,6,5,4,3,2,1] [Cocktail r]:[1]:[9,8,7,6,10,5,4,3,2,1] [Cocktail r]:[1]:[9,8,7,6,5,10,4,3,2,1] [Cocktail r]:[1]:[9,8,7,6,5,4,10,3,2,1] [Cocktail r]:[1]:[9,8,7,6,5,4,3,10,2,1] [Cocktail r]:[1]:[9,8,7,6,5,4,3,2,10,1] [Cocktail r]:[1]:[9,8,7,6,5,4,3,2,1,10] [Cocktail l]:[1]:[9,8,7,6,5,4,3,1,2,10] [Cocktail l]:[1]:[9,8,7,6,5,4,1,3,2,10] [Cocktail l]:[1]:[9,8,7,6,5,1,4,3,2,10] [Cocktail l]:[1]:[9,8,7,6,1,5,4,3,2,10] [Cocktail l]:[1]:[9,8,7,1,6,5,4,3,2,10] [Cocktail l]:[1]:[9,8,1,7,6,5,4,3,2,10] [Cocktail l]:[1]:[9,1,8,7,6,5,4,3,2,10] [Cocktail l]:[1]:[1,9,8,7,6,5,4,3,2,10] [Cocktail r]:[2]:[1,8,9,7,6,5,4,3,2,10] [Cocktail r]:[2]:[1,8,7,9,6,5,4,3,2,10] [Cocktail r]:[2]:[1,8,7,6,9,5,4,3,2,10] [Cocktail r]:[2]:[1,8,7,6,5,9,4,3,2,10] [Cocktail r]:[2]:[1,8,7,6,5,4,9,3,2,10] [Cocktail r]:[2]:[1,8,7,6,5,4,3,9,2,10] [Cocktail r]:[2]:[1,8,7,6,5,4,3,2,9,10] [Cocktail l]:[2]:[1,8,7,6,5,4,2,3,9,10] [Cocktail l]:[2]:[1,8,7,6,5,2,4,3,9,10] [Cocktail l]:[2]:[1,8,7,6,2,5,4,3,9,10] [Cocktail l]:[2]:[1,8,7,2,6,5,4,3,9,10] [Cocktail l]:[2]:[1,8,2,7,6,5,4,3,9,10] [Cocktail l]:[2]:[1,2,8,7,6,5,4,3,9,10] [Cocktail r]:[3]:[1,2,7,8,6,5,4,3,9,10] [Cocktail r]:[3]:[1,2,7,6,8,5,4,3,9,10] [Cocktail r]:[3]:[1,2,7,6,5,8,4,3,9,10] [Cocktail r]:[3]:[1,2,7,6,5,4,8,3,9,10] [Cocktail r]:[3]:[1,2,7,6,5,4,3,8,9,10] [Cocktail l]:[3]:[1,2,7,6,5,3,4,8,9,10] [Cocktail l]:[3]:[1,2,7,6,3,5,4,8,9,10] [Cocktail l]:[3]:[1,2,7,3,6,5,4,8,9,10] [Cocktail l]:[3]:[1,2,3,7,6,5,4,8,9,10] [Cocktail r]:[4]:[1,2,3,6,7,5,4,8,9,10]	
--	--	--

	[Cocktail r]:[4]:[1,2,3,6,5,7,4,8,9,10] [Cocktail r]:[4]:[1,2,3,6,5,4,7,8,9,10] [Cocktail l]:[4]:[1,2,3,6,4,5,7,8,9,10] [Cocktail l]:[4]:[1,2,3,4,6,5,7,8,9,10] [Cocktail r]:[5]:[1,2,3,4,5,6,7,8,9,10] [Result list]:[5]:[1,2,3,4,5,6,7,8,9,10] [std::sort]:[0]:[1,2,3,4,5,6,7,8,9,10] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	
100 12 11 57 58 42 32 21 13 14	You enter: [100,12,11,57,58,42,32,21,13,14] Modified bubble sort: [Start array]:[0]:[100,12,11,57,58,42,32,21,13,14] [Bubble swap]:[1]:[12,100,11,57,58,42,32,21,13,14] [Bubble swap]:[1]:[12,11,100,57,58,42,32,21,13,14] [Bubble swap]:[1]:[12,11,57,100,58,42,32,21,13,14] [Bubble swap]:[1]:[12,11,57,58,100,42,32,21,13,14] [Bubble swap]:[1]:[12,11,57,58,42,100,32,21,13,14] [Bubble swap]:[1]:[12,11,57,58,42,32,100,21,13,14] [Bubble swap]:[1]:[12,11,57,58,42,32,21,100,13,14] [Bubble swap]:[1]:[12,11,57,58,42,32,21,13,100,14] [Bubble swap]:[1]:[12,11,57,58,42,32,21,13,14,100] [Select swap]:[1]:[11,12,57,58,42,32,21,13,14,100] [Bubble swap]:[2]:[11,12,57,42,58,32,21,13,14,100] [Bubble swap]:[2]:[11,12,57,42,32,58,21,13,14,100] [Bubble swap]:[2]:[11,12,57,42,32,21,58,13,14,100] [Bubble swap]:[2]:[11,12,57,42,32,21,13,58,14,100] [Bubble swap]:[2]:[11,12,57,42,32,21,13,14,58,100] [Select swap]:[2]:[11,12,57,42,32,21,13,14,58,100] [Bubble swap]:[3]:[11,12,42,57,32,21,13,14,58,100] [Bubble swap]:[3]:[11,12,42,32,57,21,13,14,58,100] [Bubble swap]:[3]:[11,12,42,32,21,57,13,14,58,100] [Bubble swap]:[3]:[11,12,42,32,21,13,57,14,58,100] [Bubble swap]:[3]:[11,12,42,32,21,13,14,57,58,100] [Select swap]:[3]:[11,12,13,32,21,42,14,57,58,100] [Bubble swap]:[4]:[11,12,13,21,32,42,14,57,58,100] [Bubble swap]:[4]:[11,12,13,21,32,14,42,57,58,100] [Select swap]:[4]:[11,12,13,14,32,21,42,57,58,100] [Bubble swap]:[5]:[11,12,13,14,21,32,42,57,58,100] [Select swap]:[5]:[11,12,13,14,21,32,42,57,58,100] [Result list]:[5]:[11,12,13,14,21,32,42,57,58,100] Cocktail sort:	Различные положительные значения в разброс

	[Start array]:[0]:[100,12,11,57,58,42,32,21,13,14] [Cocktail r]:[1]:[12,100,11,57,58,42,32,21,13,14] [Cocktail r]:[1]:[12,11,100,57,58,42,32,21,13,14] [Cocktail r]:[1]:[12,11,57,100,58,42,32,21,13,14] [Cocktail r]:[1]:[12,11,57,58,100,42,32,21,13,14] [Cocktail r]:[1]:[12,11,57,58,42,100,32,21,13,14] [Cocktail r]:[1]:[12,11,57,58,42,32,100,21,13,14] [Cocktail r]:[1]:[12,11,57,58,42,32,21,100,13,14] [Cocktail r]:[1]:[12,11,57,58,42,32,21,13,100,14] [Cocktail r]:[1]:[12,11,57,58,42,32,21,13,14,100] [Cocktail l]:[1]:[12,11,57,58,42,32,13,21,14,100] [Cocktail l]:[1]:[12,11,57,58,42,13,32,21,14,100] [Cocktail l]:[1]:[12,11,57,58,13,42,32,21,14,100] [Cocktail l]:[1]:[12,11,57,13,58,42,32,21,14,100] [Cocktail l]:[1]:[12,11,13,57,58,42,32,21,14,100] [Cocktail l]:[1]:[11,12,13,57,58,42,32,21,14,100] [Cocktail r]:[2]:[11,12,13,57,42,58,32,21,14,100] [Cocktail r]:[2]:[11,12,13,57,42,32,58,21,14,100] [Cocktail r]:[2]:[11,12,13,57,42,32,21,58,14,100] [Cocktail r]:[2]:[11,12,13,57,42,32,21,14,58,100] [Cocktail l]:[2]:[11,12,13,57,42,32,14,21,58,100] [Cocktail l]:[2]:[11,12,13,57,42,14,32,21,58,100] [Cocktail l]:[2]:[11,12,13,57,14,42,32,21,58,100] [Cocktail l]:[2]:[11,12,13,14,57,42,32,21,58,100] [Cocktail r]:[3]:[11,12,13,14,42,57,32,21,58,100] [Cocktail r]:[3]:[11,12,13,14,42,32,57,21,58,100] [Cocktail r]:[3]:[11,12,13,14,42,32,21,57,58,100] [Cocktail l]:[3]:[11,12,13,14,42,21,32,57,58,100] [Cocktail l]:[3]:[11,12,13,14,21,42,32,57,58,100] [Cocktail r]:[4]:[11,12,13,14,21,32,42,57,58,100] [Result list]:[4]:[11,12,13,14,21,32,42,57,58,100] [std::sort]:[0]:[11,12,13,14,21,32,42,57,58,100] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	
-12 -14 - 13 0 12 - 8 45 -1 - 3 4	You enter: [-12,-14,-13,0,12,-8,45,-1,-3,4] Modified bubble sort: [Start array]:[0]:[-12,-14,-13,0,12,-8,45,-1,-3,4] [Bubble swap]:[1]:[-14,-12,-13,0,12,-8,45,-1,-3,4] [Bubble swap]:[1]:[-14,-13,-12,0,12,-8,45,-1,-3,4] [Bubble swap]:[1]:[-14,-13,-12,0,-8,12,45,-1,-3,4] [Bubble swap]:[1]:[-14,-13,-12,0,-8,12,-1,45,-3,4]	Различные целые значения в разброс

	[Bubble swap]:[1]:[-14,-13,-12,0,-8,12,-1,-3,45,4] [Bubble swap]:[1]:[-14,-13,-12,0,-8,12,-1,-3,4,45] [Select swap]:[1]:[-14,-13,-12,0,-8,12,-1,-3,4,45] [Bubble swap]:[2]:[-14,-13,-12,-8,0,12,-1,-3,4,45] [Bubble swap]:[2]:[-14,-13,-12,-8,0,-1,12,-3,4,45] [Bubble swap]:[2]:[-14,-13,-12,-8,0,-1,-3,12,4,45] [Bubble swap]:[2]:[-14,-13,-12,-8,0,-1,-3,4,12,45] [Select swap]:[2]:[-14,-13,-12,-8,0,-1,-3,4,12,45] [Bubble swap]:[3]:[-14,-13,-12,-8,-1,0,-3,4,12,45] [Bubble swap]:[3]:[-14,-13,-12,-8,-1,-3,0,4,12,45] [Select swap]:[3]:[-14,-13,-12,-8,-1,-3,0,4,12,45] [Bubble swap]:[4]:[-14,-13,-12,-8,-3,-1,0,4,12,45] [Select swap]:[4]:[-14,-13,-12,-8,-3,-1,0,4,12,45] [Result list]:[4]:[-14,-13,-12,-8,-3,-1,0,4,12,45] Cocktail sort: [Start array]:[0]:[-12,-14,-13,0,12,-8,45,-1,-3,4] [Cocktail r]:[1]:[-14,-12,-13,0,12,-8,45,-1,-3,4] [Cocktail r]:[1]:[-14,-13,-12,0,12,-8,45,-1,-3,4] [Cocktail r]:[1]:[-14,-13,-12,0,-8,12,45,-1,-3,4] [Cocktail r]:[1]:[-14,-13,-12,0,-8,12,-1,45,-3,4] [Cocktail r]:[1]:[-14,-13,-12,0,-8,12,-1,-3,45,4] [Cocktail r]:[1]:[-14,-13,-12,0,-8,12,-1,-3,4,45] [Cocktail l]:[1]:[-14,-13,-12,0,-8,12,-3,-1,4,45] [Cocktail l]:[1]:[-14,-13,-12,0,-8,-3,12,-1,4,45] [Cocktail l]:[1]:[-14,-13,-12,-8,0,-3,12,-1,4,45] [Cocktail r]:[2]:[-14,-13,-12,-8,-3,0,12,-1,4,45] [Cocktail r]:[2]:[-14,-13,-12,-8,-3,0,-1,12,4,45] [Cocktail r]:[2]:[-14,-13,-12,-8,-3,0,-1,4,12,45] [Cocktail l]:[2]:[-14,-13,-12,-8,-3,-1,0,4,12,45] [Result list]:[2]:[-14,-13,-12,-8,-3,-1,0,4,12,45] [std::sort]:[0]:[-14,-13,-12,-8,-3,-1,0,4,12,45] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	
18 18 13 4 5 4 6 2 4 2	You enter: [18,18,13,4,5,4,6,2,4,2] Modified bubble sort: [Start array]:[0]:[18,18,13,4,5,4,6,2,4,2] [Bubble swap]:[1]:[18,13,18,4,5,4,6,2,4,2] [Bubble swap]:[1]:[18,13,4,18,5,4,6,2,4,2] [Bubble swap]:[1]:[18,13,4,5,18,4,6,2,4,2] [Bubble swap]:[1]:[18,13,4,5,4,18,6,2,4,2] [Bubble swap]:[1]:[18,13,4,5,4,6,18,2,4,2]	Различные значения с повторяющимес я лементами

	[Bubble swap]:[1]:[18,13,4,5,4,6,2,18,4,2] [Bubble swap]:[1]:[18,13,4,5,4,6,2,4,18,2] [Bubble swap]:[1]:[18,13,4,5,4,6,2,4,2,18] [Select swap]:[1]:[2,13,4,5,4,6,18,4,2,18] [Bubble swap]:[2]:[2,4,13,5,4,6,18,4,2,18] [Bubble swap]:[2]:[2,4,5,13,4,6,18,4,2,18] [Bubble swap]:[2]:[2,4,5,4,13,6,18,4,2,18] [Bubble swap]:[2]:[2,4,5,4,6,13,18,4,2,18] [Bubble swap]:[2]:[2,4,5,4,6,13,4,18,2,18] [Bubble swap]:[2]:[2,4,5,4,6,13,4,2,18,18] [Select swap]:[2]:[2,2,5,4,6,13,4,4,18,18] [Bubble swap]:[3]:[2,2,4,5,6,13,4,4,18,18] [Bubble swap]:[3]:[2,2,4,5,6,4,13,4,18,18] [Bubble swap]:[3]:[2,2,4,5,6,4,4,13,18,18] [Select swap]:[3]:[2,2,4,5,6,4,4,13,18,18] [Bubble swap]:[4]:[2,2,4,5,4,6,4,13,18,18] [Bubble swap]:[4]:[2,2,4,5,4,4,6,13,18,18] [Select swap]:[4]:[2,2,4,4,5,4,6,13,18,18] [Bubble swap]:[5]:[2,2,4,4,4,5,6,13,18,18] [Select swap]:[5]:[2,2,4,4,4,5,6,13,18,18] [Result list]:[5]:[2,2,4,4,4,5,6,13,18,18] Cocktail sort: [Start array]:[0]:[18,18,13,4,5,4,6,2,4,2] [Cocktail r]:[1]:[18,13,18,4,5,4,6,2,4,2] [Cocktail r]:[1]:[18,13,4,18,5,4,6,2,4,2] [Cocktail r]:[1]:[18,13,4,5,18,4,6,2,4,2] [Cocktail r]:[1]:[18,13,4,5,4,18,6,2,4,2] [Cocktail r]:[1]:[18,13,4,5,4,6,18,2,4,2] [Cocktail r]:[1]:[18,13,4,5,4,6,2,18,4,2] [Cocktail r]:[1]:[18,13,4,5,4,6,2,4,18,2] [Cocktail r]:[1]:[18,13,4,5,4,6,2,4,2,18] [Cocktail l]:[1]:[18,13,4,5,4,6,2,2,4,18] [Cocktail l]:[1]:[18,13,4,5,4,2,6,2,4,18] [Cocktail l]:[1]:[18,13,4,5,2,4,6,2,4,18] [Cocktail l]:[1]:[18,13,4,2,5,4,6,2,4,18] [Cocktail l]:[1]:[18,13,2,4,5,4,6,2,4,18] [Cocktail l]:[1]:[18,2,13,4,5,4,6,2,4,18] [Cocktail l]:[1]:[2,18,13,4,5,4,6,2,4,18] [Cocktail r]:[2]:[2,13,18,4,5,4,6,2,4,18] [Cocktail r]:[2]:[2,13,4,18,5,4,6,2,4,18] [Cocktail r]:[2]:[2,13,4,5,18,4,6,2,4,18] [Cocktail r]:[2]:[2,13,4,5,4,18,6,2,4,18] [Cocktail r]:[2]:[2,13,4,5,4,6,18,2,4,18] [Cocktail r]:[2]:[2,13,4,5,4,6,2,18,4,18]	
--	--	--

	[Cocktail r]:[2]:[2,13,4,5,4,6,2,4,18,18] [Cocktail l]:[2]:[2,13,4,5,4,2,6,4,18,18] [Cocktail l]:[2]:[2,13,4,5,2,4,6,4,18,18] [Cocktail l]:[2]:[2,13,4,2,5,4,6,4,18,18] [Cocktail l]:[2]:[2,13,2,4,5,4,6,4,18,18] [Cocktail l]:[2]:[2,2,13,4,5,4,6,4,18,18] [Cocktail r]:[3]:[2,2,4,13,5,4,6,4,18,18] [Cocktail r]:[3]:[2,2,4,5,13,4,6,4,18,18] [Cocktail r]:[3]:[2,2,4,5,4,13,6,4,18,18] [Cocktail r]:[3]:[2,2,4,5,4,6,13,4,18,18] [Cocktail r]:[3]:[2,2,4,5,4,6,4,13,18,18] [Cocktail l]:[3]:[2,2,4,5,4,4,6,13,18,18] [Cocktail l]:[3]:[2,2,4,4,5,4,6,13,18,18] [Cocktail r]:[4]:[2,2,4,4,4,5,6,13,18,18] [Result list]:[4]:[2,2,4,4,4,5,6,13,18,18] [std::sort]:[0]:[2,2,4,4,4,5,6,13,18,18] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	
<один пробел>	You enter: [] Modified bubble sort: [Start array]:[0]:[] Cocktail sort: [Start array]:[0]:[] [Result list]:[0]:[] [std::sort]:[0]:[] Comparison of an array sorted using bubble sorting with std::sort was successful Comparison of an array sorted using cocktail sorting with std::sort was successful	Вместо строки чисел — 1 пробел
<пустая строка>	Error: Wrong string format	Пустая строка
12 13 5 6 --12 18 31 1 0 8	Error: Wrong string format	Неправильный формат отрицательного числа
0 19 b c a 22 14 18 31 y	Error: Wrong string format	Неправильный тип вводимых данных