**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра математического обеспечения и применения ЭВМ**

# ОТЧЕТ

**по лабораторной работе №4**
**по дисциплине «Алгоритмы и структуры данных»**
**Тема: Сортировки**

Студент гр. 9304 _____ Силкин В.А.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург
2020

**Цель работы.**

Изучить способы сортировки в массиве данных и реализовать одну из них с помощью языка программирования C++.

**Задание.**

**Вариант 23.1.**

J-сортировка, 1 вариант:

- Обычная J-сортировка.

**Выполнение работы.**

Программа принимает на вход строку из чисел, разделённых любым количеством пробелов. Если любое число больше, чем int max, то программа не выполняет сортировку. Также она не выполнит её в случае, если 1 из символов строки не является пробелом или цифрой.

Все числа записываются в вектор, а затем этот вектор передаётся классу Tree. В нём присутсвуют публичные методы на распечатку самого вектора как дерево, J-сортировку, сортировку вставками, а также на получение обратного вектора (сам вектор также доступен публично).

Приватны методы распечатывания перестановки и сортировки кучи по невозрастанию/неубыванию, т.к. их использует только J-сортировка.

**Тестирование.**

Был написан bash-скрипт для тестирования программы, который выводит результат работы с данными, поступающими из файлов вида "test_<номер>.txt"

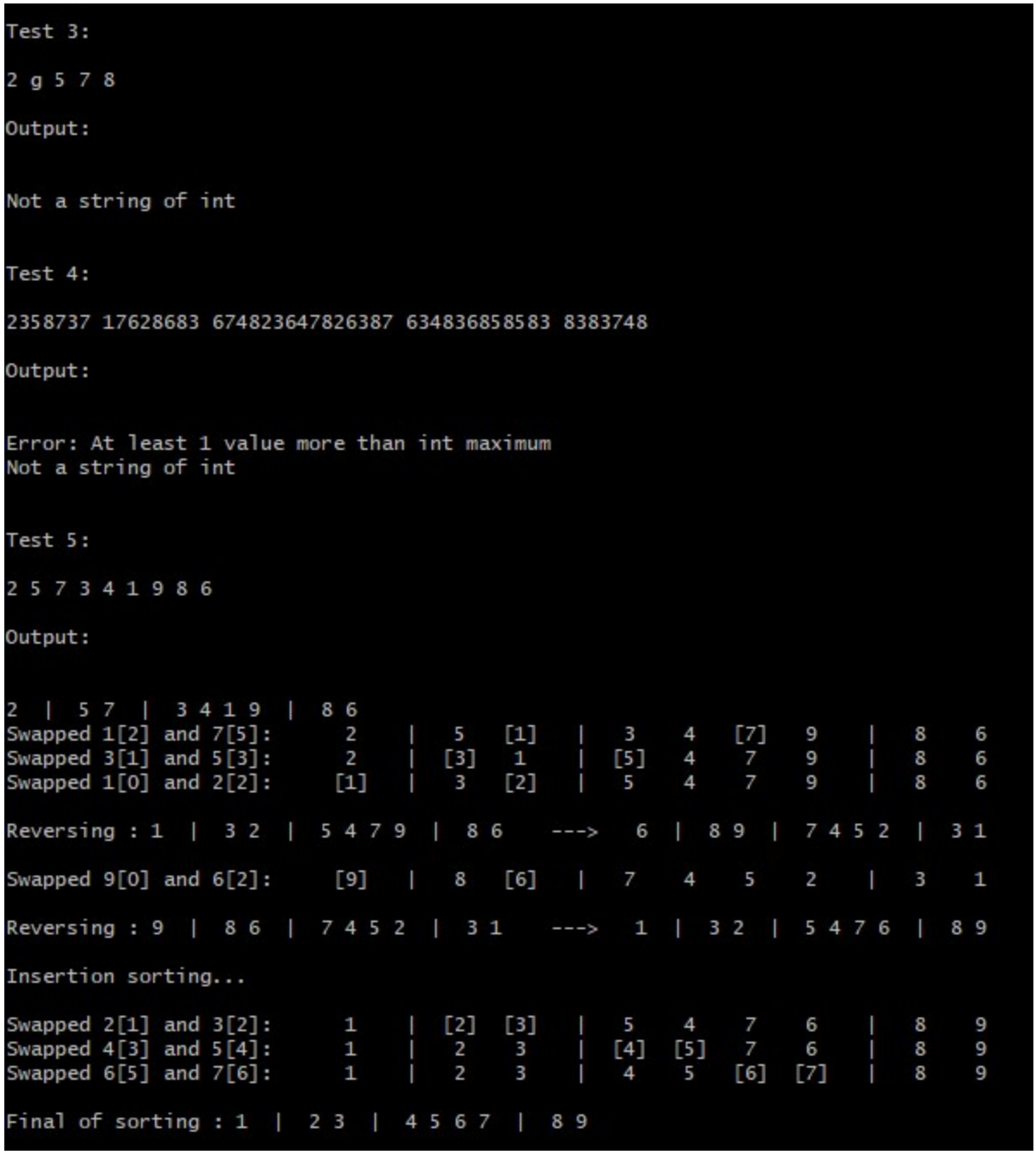


Рисунок 1 - Часть вывода bash-скрипта

**Выводы.**

Был реализован обычный алгоритм сортировки J-sort со средней сложностью O(n*log(n)), а также был изучен способ сортировки вставками, являющийся частью J-sort сортировки

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
/*
23. JSort обычный, и многократное построение пары во всём массиве, затем на половине элементов, четверти... Сорт вставками на всём массиве.
*/
#include <iostream>
#include <string>
#include <vector>

bool isfulltree(int cur) {
    if (cur == 1) {
        return true;
    } else if((cur+1)%2 != 0) {
        return false;
    }
    return isfulltree(((cur+1)/2)-1);
}

class Tree
{

    void swap_parent_MT(int pos_par, std::vector<int>& vec) {
        int add = 1;
        if(vec[2*pos_par+1] > vec[2*pos_par+2]) {
            add++;
        }
        if(vec[pos_par] > vec[2*pos_par+add]) {
            std::swap(vec[pos_par], vec[2*pos_par+add]);
            printswap(pos_par, 2*pos_par+add, vec);
            if(2*pos_par+add <= ((vec.size()-1)/2)-1) {
                swap_parent_MT(2*pos_par+add, vec);
            }
        }
    }

    void swap_parent_LT(int pos_par, std::vector<int>& vec) {
        int add = 1;
        if(vec[2*pos_par+1] < vec[2*pos_par+2]) {
            add++;
        }
        if(vec[pos_par] < vec[2*pos_par+add]) {
            std::swap(vec[pos_par], vec[2*pos_par+add]);
            printswap(pos_par, 2*pos_par+add, vec);
```

```
            if(2*pos_par+add <= ((vec.size()-1)/2)-1) {
                swap_parent_LT(2*pos_par+add, vec);
            }
        }
    }

    void printswap(int a, int b, std::vector<int>& vec) {
        int space = 4;
        if(a>=10 || b>=10) space--;
        if(a>=10 && b>=10) space--;
        if(vec[a]>=10 || vec[b]>=10) space--;
        if(vec[a]>=10 && vec[b]>=10) space--;
        std::cout << "Swapped " << vec[a] << '[' << a << ']' << " and " << vec[b] << '[' << b <<
']' << ": ";
        for (int i = 0; i < space; i++) {
            std::cout << ' ';
        }
        for (int i = 0; i < vec.size(); i++) {
            if(isfulltree(i)) {
                std::cout << " |  ";
            }
            if(i == a || i == b) {
                std::cout << '[' << vec[i] << "] ";
                if(vec[i] <= 10) {
                    std::cout << ' ';
                }
                continue;
            }
            std::cout << ' ' << vec[i] << "  ";
            if(vec[i] <= 10) {
                std::cout << ' ';
            }
        }
        std::cout << '\n';
    }

public:

    std::vector<int> self_vec;

    Tree(std::vector<int>& vec) : self_vec(vec) {
        printtree(self_vec);
        std::cout << '\n';
    }

    void JSort() {
        for(int i = ((self_vec.size()-1)/2)-1; i >= 0; i--) {
            swap_parent_MT(i, self_vec);
        }
        std::vector<int> mirror = get_mirror_tree(self_vec);
```

```
        std::cout << "\nReversing : ";
        printtree(self_vec);
        std::cout << "   --->   ";
        printtree(mirror);
        std::cout << "\n\n";
        for(int i = ((mirror.size()-1)/2)-1; i >= 0; i--) {
            swap_parent_LT(i, mirror);
        }
        this->self_vec = get_mirror_tree(mirror);
        std::cout << "\nReversing : ";
        printtree(mirror);
        std::cout << "   --->   ";
        printtree(self_vec);
        std::cout << "\n\nInsertion sorting...\n\n";
        InSort(self_vec);
        std::cout << "\nFinal of sorting : ";
        printtree(self_vec);
        std::cout << "\n\n";
    }

    std::vector<int> get_mirror_tree(std::vector<int> vec) {
        std::vector<int> mirror;
        for(int i=0; i < vec.size(); i++) {
            mirror.push_back(vec[(vec.size()-1)-i]);
        }
        return mirror;
    }

    void printtree(std::vector<int> vec) {
        for (int i = 0; i < vec.size(); i++) {
            if(isfulltree(i)) {
                std::cout << " |  ";
            }
            std::cout << vec[i] << ' ';
        }
    }

    void InSort(std::vector<int>& vec){
              for(int i=1;i<vec.size();i++){
                        for(int j=i; j>0 && vec[j-1]>vec[j];j--){
                                  std::swap(vec[j-1], vec[j]);
               printswap(j-1,j, vec);
                        }
              }
    }

//2k+1 - левый потомок
//2k+2 - правый потомок
};
```

```
bool ConvertString(std::string expression, std::vector<int>& vec) {
    int count = 0;
    std::string tmp = "";
    for(auto iter = expression.begin(); iter != expression.end();) {
        if(isspace(*iter)) {
            iter++;
            continue;
        } else if(!isdigit(*iter)) {
            return false;
        }
        tmp.push_back(*iter);
        iter++;
        while(isdigit(*iter) && iter != expression.end()) {
            tmp.push_back(*iter);
            if(tmp.size()>10) {
                std::cout << "Error: At least 1 value more than int maximum\n";
                return false;
            }
            iter++;
        }
        if(std::stol(tmp) < 2147483647) {
            vec.push_back(std::stoi(tmp));
        } else {
            std::cout << "Error: At least 1 value more than int maximum\n";
            return false;
        }
        tmp.erase();
        count++;
    }
    return true;
}

int main() {
    std::cin.ignore (std::string::npos,'\n');
    std::string str;
    std::getline(std::cin,str);
    std::vector<int> vec;
    if(ConvertString(str, vec)) {
        Tree sapling(vec);
        sapling.JSort();
    } else {
        std::cout << "Not a string of int\n";
    }
    return 0;
}
```