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

# ОТЧЕТ

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

Студентка гр. 9382 _____ Балаева М.О.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

**Цель работы.** Применить на практике знания о сортировке слиянием, реализовать сортировку слиянием для массива на языке C++.

**Основные теоретические положения.**

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке.

Сортировка естественным слиянием —это сортировка, в которой ищутся упорядоченные подмассивы, которые впоследствии соединяют в общий упорядоченный подмассив. Алгоритм повторяется, пока массив не будет отсортирован.

**Задание.**

Вариант №16

Сортировка массивов слиянием — естественное слияние.

**Выполнение работы:**

Для реализации Сортировки естественным слиянием был использован алгоритм Неймана.

1. Всего используется три «ленты» — основная inp[], на которой записаны не отсортированные данные, и две вспомогательные first[] и second[].
2. Данные последовательно считываются с основной ленты.
3. Пока последовательно считываемые данные представляют из себя упорядоченный подмассив, они переписываются на одну из вспомогательных лент.
4. Как только завершается очередной отсортированный подмассив (т.е. на основной ленте встречается элемент, меньший чем предыдущий),

берется индекс элемента, являющегося конечным в очередном упорядоченном подмассиве. Он записывается в очередь qfirst или qsecond в зависимости от того, на какой ленте был подмассив. Далее происходит переключение на другую вспомогательную ленту.

5. Пункты 2-4 повторяются снова, только данные переносятся уже на другую вспомогательную ленту. При завершении очередного упорядоченного подмассива на основной ленте происходит поочерёдное переключение то на одну вспомогательную ленту, то на другую.
6. Когда все данные с основной ленты считаны, происходит обработка вспомогательных лент.
7. С обеих вспомогательных лент поочерёдно считываются данные.
8. При этом очередные данные с двух лент сравниваются между собой. По результатами сравнения на основную ленту записывается меньший элемент из пары.
9. Так как границы массивов на вспомогательных лентах записаны в очереди qfirst и qsecond, считывание и сравнение происходит только в пределах отсортированных подмассивов.
10. Если на одной из вспомогательных лент закончились элементы очередного подмассива, то с оставшейся ленты остаток подмассива просто переносится на основную ленту.
11. Повторяем весь процесс заново до тех пор, пока данные на основной ленте не будут собой представлять полностью упорядоченный массив.

**Иллюстрация работы программы:**

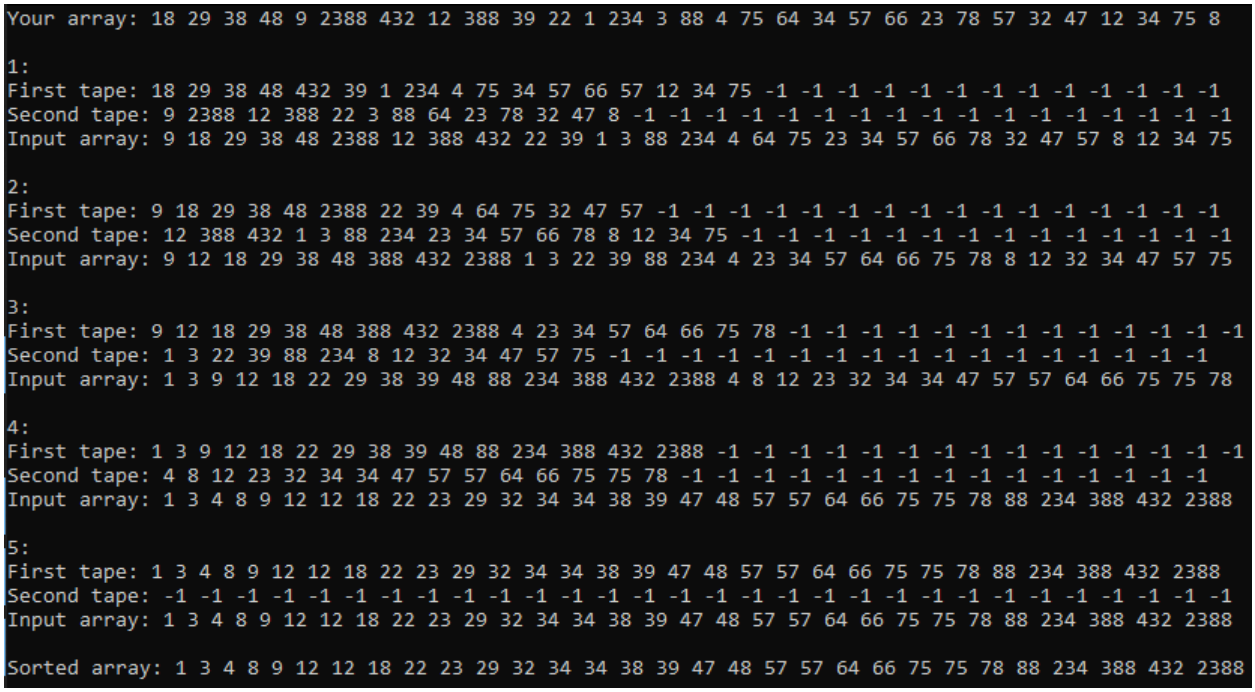


Рис. 1 – работа программы при подаче массива из 30 элементов

**Тестирование.**

| № | Входные данные | Выходные данные |
|---|---|---|
| 1. | 6<br>1 4 2 6 5 3 | Your array: 1 4 2 6 5 3<br>Sorted array: 1 2 3 4 5 6 |
| 2. | 5<br>5 2 1 4 3 | Your array: 5 2 1 4 3<br>Sorted array: 1 2 3 4 5 |
| 3. | 8<br>10 7 9 8 6 4 1 5 | Your array: 10 7 9 8 4 1 5<br>Sorted array: 1 4 5 7 8 9 10 |
| 4. | 6<br>1 4 2 -6 5 -3 | Your array: 1 4 2 -6 5 -3<br>Sorted array: -6 -3 1 2 4 5 |
| 5. | -6<br>1 4 2 -6 5 -3 | Размер массива не может быть отрицательным |

**Вывод.**

Был реализован алгоритм сортировки естественным слиянием на языке программирования C++.

## Приложение А
## Исходный код программы

Название файла: main.c

```
#include <iostream>

#include <bits/stdc++.h>

#define INF 1215752192
using namespace std;
template < typename T >
  void mergesort(int n, T inp[]) {
    T first[n]; //первая шина
    T second[n]; //вторая шина
    int cf = 0; //счетчик первой шины
    int cs = 0; //счетчик второй шины
    bool tape = true; //счетчик,показывающий,и какую шину используем
    int subarray = 0; //счетчик конца отсортированных под массивов
    queue < int > qfirst; //очередь концов отсортированных подмассивов первой шины
    queue < int > qsecond; //очередь концов отсортированных подмассивов второй шины
    int a = 0, b = 0;
    do {
      for (int i = 0; i < n; i++) {
        first[i] = INF; //заполняем шины большими числами
        second[i] = INF;
      }
      for (int i = 0; i < n; i++) //проходимся помассиву ищем все отсортированные
подмассивы
      {
        if (tape) //записываем подмассивы, меняя шины
        {
          first[cf] = inp[i];
          cf++;
        } else {
          second[cs] = inp[i];
          cs++;
        }
        if (i != n - 1) {
          if (inp[i + 1] < inp[i]) {
            if (tape) {
              qfirst.push(cf - 1); //запоминаем концы подмассивов

            } else {

              qsecond.push(cs - 1); //запоминаем концы подмассивов

            }
            tape = !tape; //меняем шину
          }
        } else {
          if (tape) {
            qfirst.push(cf - 1);
          } else {
            qsecond.push(cs - 1);
```

```
          }
        }
      }
      if (second[0] != INF) //если массив еще не отсортирован
      {
        for (int i = 0; i < n; i++) {
          if (subarray == 0) {
            inp[i] = min(first[a], second[b]); //заполняем массив,выби рая минимальный
изд вух сравниваемых
            if (inp[i] == first[a]) {
              a++;
              if (qfirst.empty() || qfirst.front() == a - 1) //если закончился подмассив
напервой шине
              {
                subarray = 1;
                qfirst.pop();
              }
            } else {
              b++;
              if (qsecond.empty() || qsecond.front() == b - 1) //если закончился
подмассив на второй шине
              {
                subarray = 2;
                qsecond.pop();
              }
            }
          } else {
            if (subarray == 1) //если закончился подмассив на первой шине
            {
              inp[i] = second[b];
              b++;
              if (qsecond.empty() || qsecond.front() == b - 1) {
                subarray = 0;
                qsecond.pop();
              }
            }
            if (subarray == 2) //если закончился подмассив на второй шине
            {
              inp[i] = first[a];
              a++;
              if (qfirst.empty() || qfirst.front() == a - 1) {
                subarray = 0;
                qfirst.pop();
              }
            }
          }
        }
      }
      tape = true; //обнуляем переменные
      subarray = 0;
      a = 0;
      b = 0;
      cf = 0;
      cs = 0;
```

```
    }
    while (second[0] != INF); //повторяем, пока массив не будет отсортирован
  }

int main() {
  ifstream fin("test.txt");
  int n;
  while (fin >> n) {
    int inp[n];
    for (int i = 0; i < n; i++) {
      fin >> inp[i]; //считываем массив
    }
    cout << "Your array: ";
    for (int i = 0; i < n; i++) {
      cout << inp[i] << " ";
    }
    cout << "\n";
    mergesort(n, inp);
    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
      cout << inp[i] << " ";
    }
    cout << "\n\n";
    fin >> n;
  }
  fin.close();
  return 0;
}
```