Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Алгоритмы и структуры данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Сортировка слиянием на основе кольцевой очереди»

Выполнили:

Дынина Е.А., студент группы N3249

_____
(подпись)

Проверил:

Ерофеев С.А.

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург

2025 г.

СОДЕРЖАНИЕ

Содержание 3
Постановка задачи 4
Техническая задача 5
Входные данные 5
Промежуточные данные 5
Выходные данные 6
Используемые функции 6
Блок-схема 7
Код 10
Тестирование 13
Заключение 14

ПОСТАНОВКА ЗАДАЧИ

Цель работы — разработать программу сортировки нисходящим слиянием для чисел из файла, используя кольцевую очередь.

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить структуру кольцевой очереди;
- Создать блок-схему алгоритма с обработкой ошибок;
- Написать программу на языке C++ с интерфейсом для пользователей.

ТЕХНИЧЕСКАЯ ЗАДАЧА

Очередь — абстрактный тип данных, экземпляр которого представляет собой список, в котором вставка и удаление элементов выполняются с разных концов. Элементы добавляются с конца, а удаляются с начала. Базовые операции с очередью:

- Проверка на пустоту;
- Добавление элементов;
- Удаление элементов;
- Определение размера очереди.

Сортировка слиянием заключается в рекурсивном разбиении очереди на две равные части. Каждая часть сортируется отдельно той же сортировкой. После две отсортированные части сливаются в одну упорядоченную очередь, сравнивая элементы по очереди.

Входные данные

На вход в программу через указанный заранее файл подается кольцевая очередь с числами типа double (диапазон $[-1{,}7 \cdot 10^{308}, 1{,}7 \cdot 10^{308}]$).

Промежуточные данные

В ходе работы программы задействованы промежуточные переменные, указанные в таблице 1. Во время рекурсивной сортировки используются очереди left и right с числами типа double (диапазон $[-1{,}7 \cdot 10^{308}, 1{,}7 \cdot 10^{308}]$).

Таблица 1 – Промежуточные данные

Название переменной	**Тип в C++**	**Диапазон типа**	**Значение**
file_path	string	—	Путь к файлу с числами
num	long double	$[-1{,}2 \cdot 10^{4932}, 1{,}2 \cdot 10^{4932}]$	Переменная, в которую считываем числа из файла
cur	int	$[-2147483647, 2147483647]$	Позиция указателя
half	int	$[-2147483647, 2147483647]$	Половина очереди
N	int	$[-2147483647, 2147483647]$	Максимальный размер очереди

i	int	$[-2147483647, 2147483647]$	Переменная для реализации перебора в цикле
x	double	$[-1{,}7 \cdot 10^{308}, 1{,}7 \cdot 10^{308}]$	Переменная для удаления числа из очереди

Выходные данные

В результате работы программы в консольное приложение выводится отсортированная кольцевая очередь с числами типа double (диапазон $[-1{,}7 \cdot 10^{308}, 1{,}7 \cdot 10^{308}]$).

Используемые функции

- **ifstream —** открывает файл для чтения;
- **.is_open() —** проверяет, открыт ли файл;
- **numeric_limits <тип>::lowest —** возвращает наименьшее отрицательное конечное значение типа;
- **numeric_limits <тип>::max —** возвращает максимальное конечное значение типа;
- **.clear() —** сбрасывает состояние потока ifstream:
- **.tellg() —** возвращает текущую позицию указателя в файле;
- **.seekg() —** устанавливает указатель на определенную позицию в файле;
- **.close()** — закрывает файл;
- **merge(queue &q, queue &left, queue &right) —** сравнивает две части очереди, добавляет сначала меньшие значения в очередь, а потом оставшиеся;
- **merge_sort(queue &q) —** делит очередь на две части и проводит сортировку;

Функции класса queue

- **.add(double x) —** добавление элемента x в конец очереди;
- **.pop() —** удаление и возвращение элемента из начала очереди;
- **.empty() —** проверка очереди на пустоту;
- **.size() —** выводит размер кольцевой очереди queue;
- **.print() —** выводит все элементы кольцевой очереди queue.

БЛОК-СХЕМА

Рассмотрим блок-схему придуманного алгоритма, где реализована сортировка нисходящим слиянием на основе кольцевой очереди и учтены всевозможные ошибки.

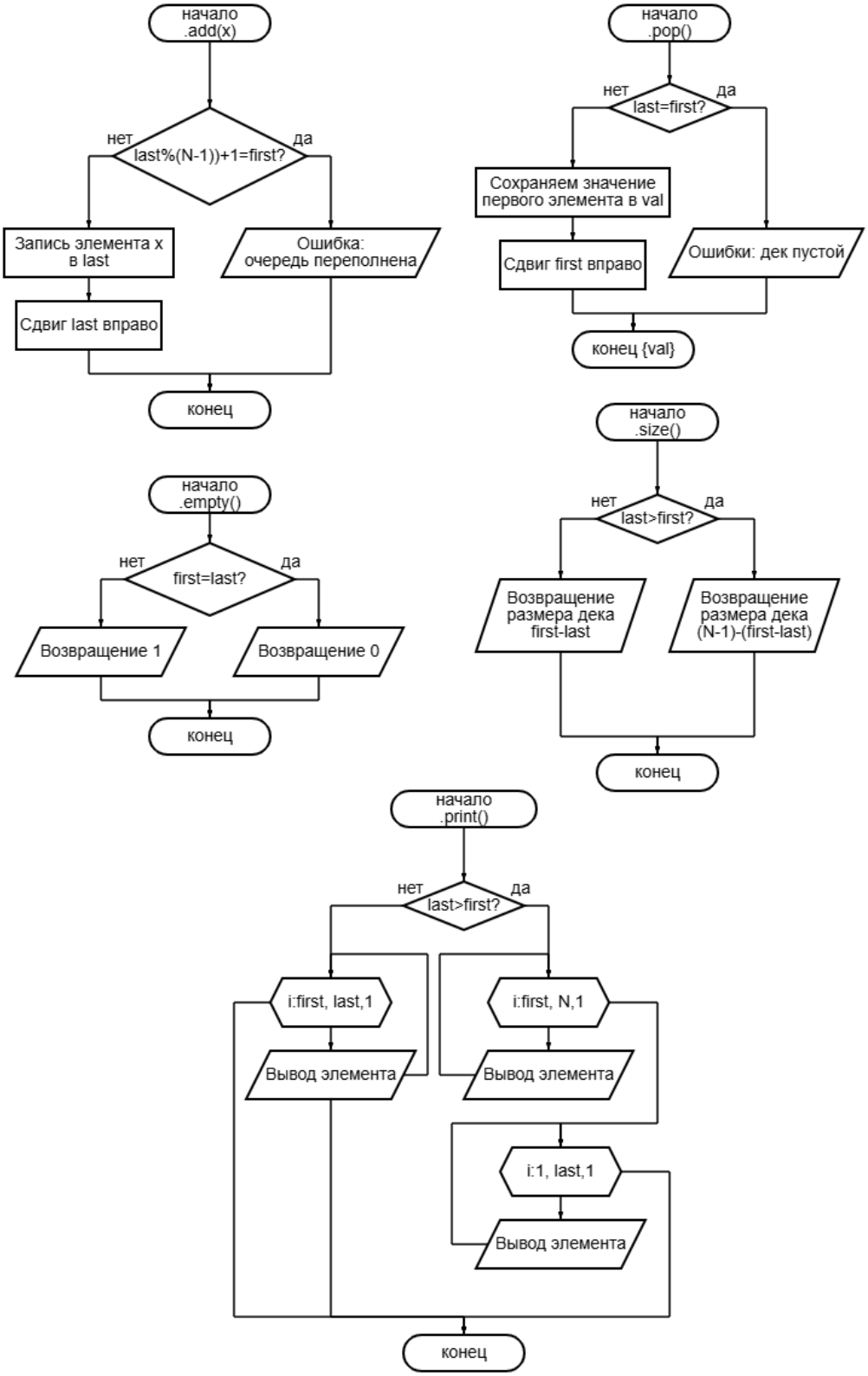

Рисунок 1 – Блок-схема функций класса queue

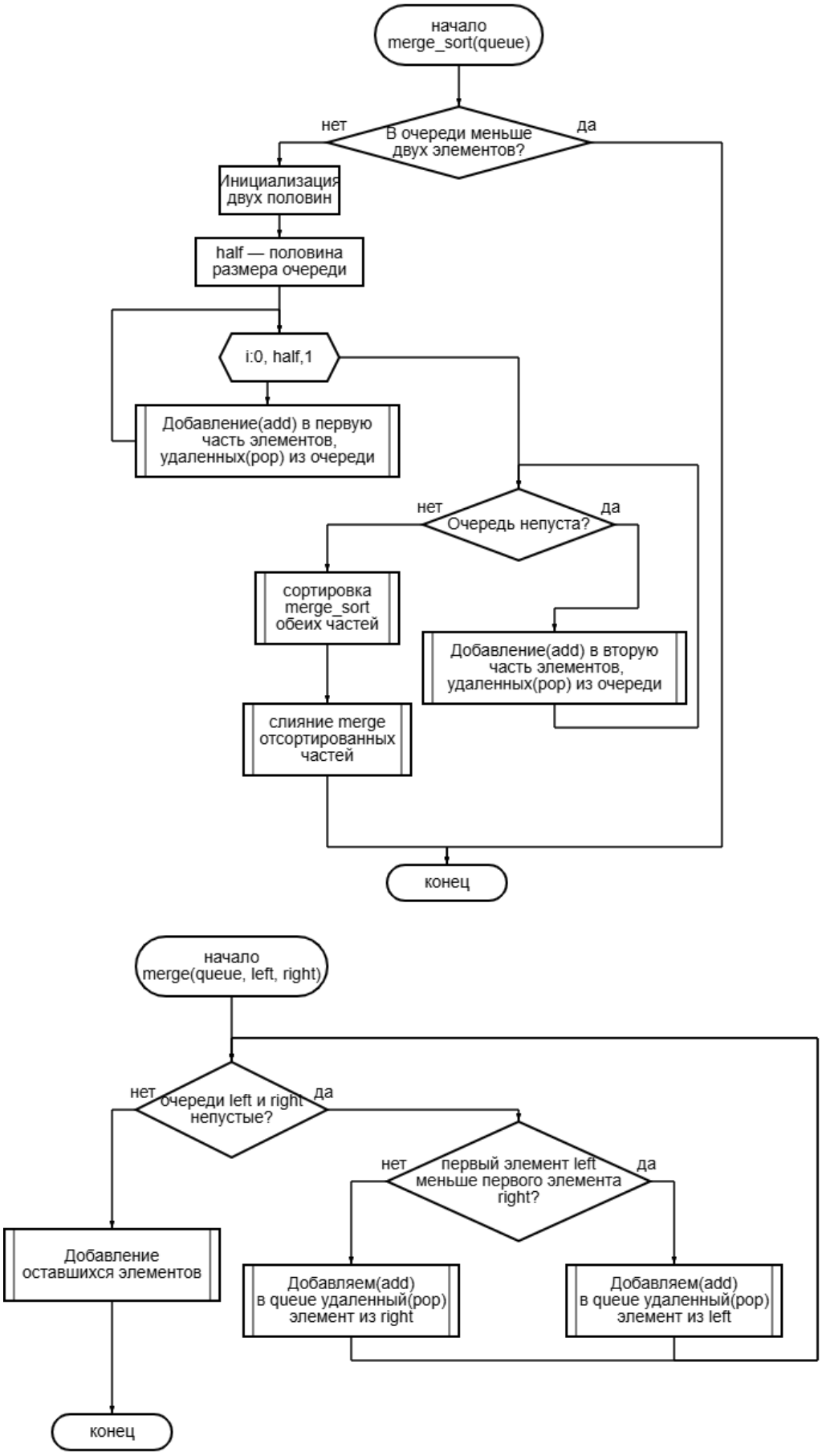

Рисунок 2 – Блок-схема функций слияния и сортировки

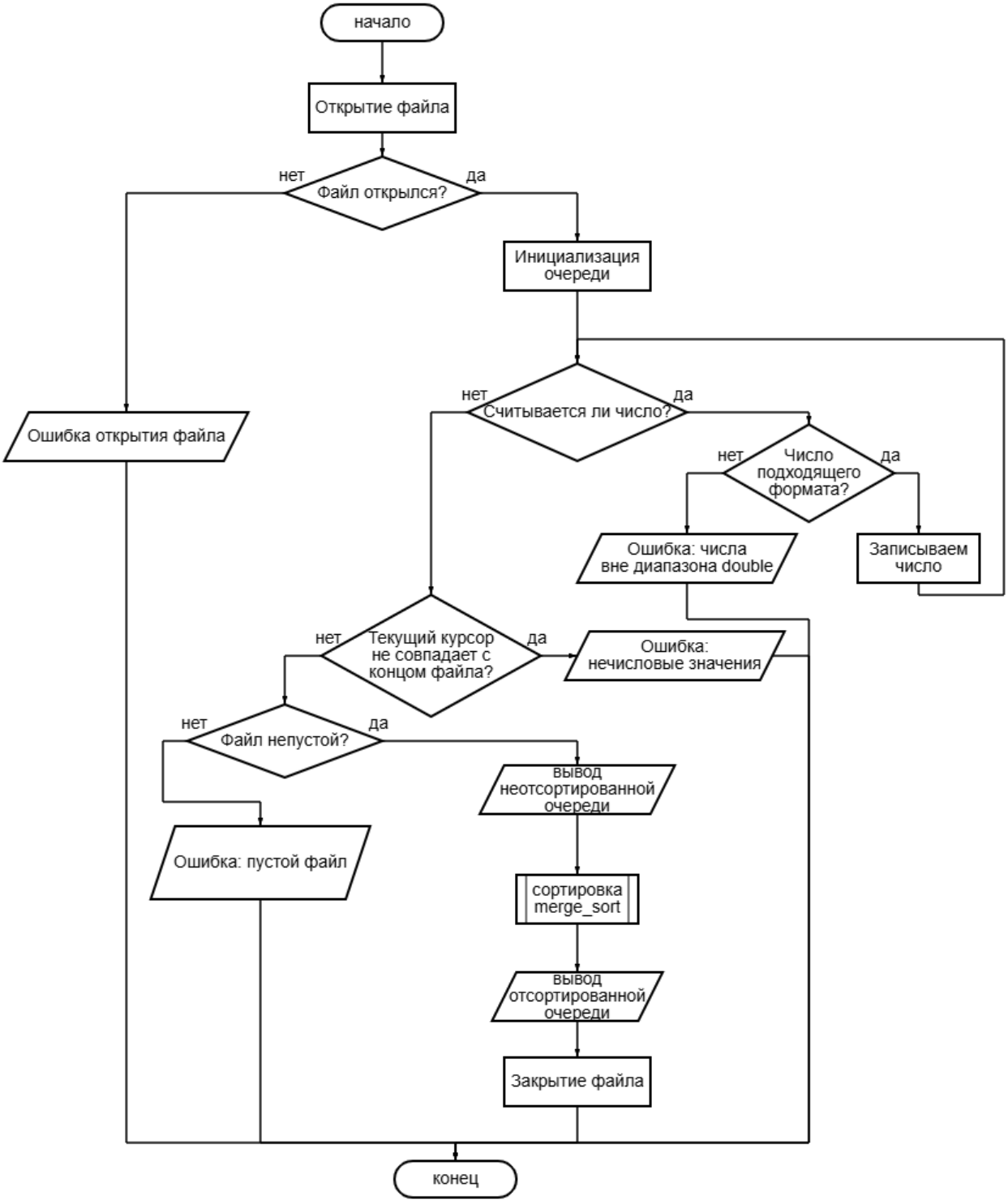

Рисунок 3 – Блок-схема основной части программы

КОД

Ниже приведен код, написанный для сортировки слиянием чисел, подаваемых через файл в кольцевую очередь, по указанной выше блок-схеме. Программа реализована на языке C++.

```
#include <iostream>
#include <fstream>
#include <limits>
#include <stdexcept>

using namespace std;
const int N=10000;
class queue {
public:
	int data[N];
	int first; //указатель на начало
	int last; //указатель на конец
	queue(): first(1), last(1) {}//оставляем зазор в виде одного элемента
	bool empty() {//проверка очереди на пустоту
		if (last==first) return true;
		else return false;
	}
	// Добавление элемента
	void add(double value) {
		if ((last%(N-1))+1==first){
			throw runtime_error("The queue is full\n");
		}
		else {
			data[last]=value;
			last=(last%(N-1))+1;// Сдвиг last вправо
		}
	}
	// Удаление элемента
	double pop() {
		if (last==first){
			throw runtime_error("The queue is empty\n");
		}
		else {
			double x=data[first];
			first=(first%(N-1))+1;// Сдвиг first вправо
			return x;
		}
	}
	// Размер очереди
	int size() {
		if (first>last)
			return (N-1)-(first-last);
		else
			return last-first;
	}
	// Вывод очереди
	void print() {
		if (first<last){
			for(int i=first;i<last;i++){
				cout<<data[i]<<" ";
			}
			cout<<endl;
		}
```

```
            else {
                  int i;
                  for(i=first;i<N;i++){
                        cout<<data[i]<<" ";
                  }
                  for(i=1;i<last;i++){
                        cout<<data[i]<<" ";
                  }
                  cout<<endl;
            }
      }
};

void merge(queue &q, queue &left, queue &right) {
//Добавление меньших чисел
      while (!left.empty() && !right.empty()) {
            // Сравнение первых элементов из частей и добавление меньшего
                  if (left.data[left.first]<right.data[right.first]) {
                        q.add(left.pop());
                  }
            else {
                        q.add(right.pop());
                  }
}
      // Добавление оставшихся элементов
      while (!left.empty()) q.add(left.pop());
      while (!right.empty()) q.add(right.pop());
}

// Рекурсивная сортировка слиянием
void merge_sort(queue &q) {
if(q.size() <= 1) return;
      // Разделение очереди на две части
      queue left, right;
      int half = q.size() / 2;
      // Заполнение первой части
      for (int i=0; i<half; i++) {
                  left.add(q.pop());
      }
      // Заполнение второй части
      while (!q.empty()) {
                  right.add(q.pop());
      }
      // Сортировка каждой части
      merge_sort(left);
      merge_sort(right);
      // Слияние двух отсортированных частей
      merge(q, left, right);
}

int main() {
string file_path="data.txt";
      // Открываем файл для чтения
      ifstream file(file_path);
      // Проверка на ошибку при открытии
      if (!file.is_open()) {
                  cerr << "Error opening file" << endl;
                  return 1;
      }
      queue q;
      long double num;
       while (file >> num) {
```

```
      // Проверка на корректность данных
      if(num<numeric_limits<double>::max() and
num>numeric_limits<double>::lowest()){
            q.add(num);
            continue;
      }
      cout<<"Error: numbers out of range";
      return 1;
      }
      // Проверка на ввод всех символов
      file.clear(); // Сброс возможных ошибок
      int cur=file.tellg(); // Сохраняем позицию курсора
      file.seekg(0, ios::end); // Перемещаем курсор в конец файла
      if(cur!=file.tellg()) { //Сравниваем сохраненный курсор с концом файла
            cout<<"Error: non-numeric data";
            return 1;
      }
      // Проверка на наличие данных
      if(q.size()==0){
            cout<<"Error: empty file";
            return 1;
      }
      cout<<"Unsorted queue:\n";
      q.print();
      merge_sort(q);
      cout<<"Sorted queue:\n";
      q.print();
return 0;
}
```

ТЕСТИРОВАНИЕ

Проведено тестирование программы с помощью различных входных данных. Был использован компилятор C++ 3.4.2 и среда разработки Embarcadero Dev-C++ 6.3.

1. Очередь с разнообразными числами (1e+2 -200 300 -9 4)

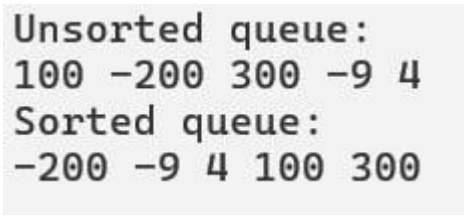

Рисунок 4 – Тест №1

2. Очередь, где значения возрастают (-9 -5 0 6 34)

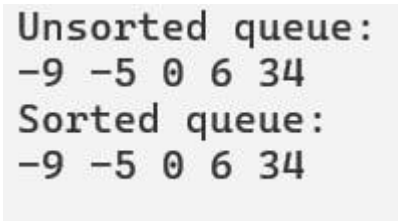

Рисунок 5 – Тест №2

3. Очередь с большим количеством чисел (-88 59 -13 -51 2 -15 8 49 -78 35 2 20 -4 52 85 6 49 1 4 28 -2 03 -2 7753 -3456 -1647 782)

Рисунок 6 – Тест №3

4. Файл с нечисловыми символами (1e+2 -20sdf0 300 -9 4)

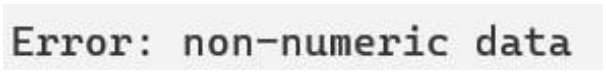

Рисунок 7 – Тест №4

5. Файл с числами, превышающими диапазон типа double (1e+3000 -200 300 -9 4)

Рисунок 8 – Тест №5

ЗАКЛЮЧЕНИЕ

Был изучен класс деков и написана блок-схема алгоритма программы, которая реализует сортировку слиянием чисел из файла, используя кольцевую очередь. На языке C++ написан код программы с выводом неотсортированной и отсортированной очереди, обработкой ошибок и интерфейсом для пользователя. Программа успешно протестирована на различных наборах данных.

Все задачи выполнены и цель достигнута.