

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

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Алгоритмы и структуры данных »**  
**Тема: Демонстрация сортировки слабой кучей**

Студентка гр. 9382

\_\_\_\_\_

Голубева В.П.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студентка Голубева В.П.

Группа 9382

Тема работы: Демонстрация работы алгоритма сортировки

Исходные данные:

на вход программе подаётся цифра — количество элементов и  
целочисленный массив, элементы массива разделены пробелом

Содержание пояснительной записки:

«Содержание», «Введение», «Ход выполнения работы», «Заключение»,  
«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 31.10.2020

Дата сдачи реферата: 15.11.2020

Дата защиты реферата: 15.11.2020

Студентка

---

Голубева В.П.

Преподаватель

---

Фирсов М.А.

## **АННОТАЦИЯ**

В курсовой работе происходит сортировка массива. Программа демонстрирует процесс сортировки при помощи вывода на экран состояния элементов на каждом шаге, раскрашивая в другой цвет сравниваемые элементы. Результатом будет являться отсортированный массив.

Примеры работы реализованной программы представлены в приложении А, исходный код приведён в приложении Б.

## **SUMMARY**

In the course work, the array is sorted. The program demonstrates the sorting process by displaying the status of elements on the screen at each step, coloring the compared elements in a different color. The result will be a sorted array.

Examples of how the implemented program works are shown in appendix A, and the source code is given in appendix B.

## СОДЕРЖАНИЕ

	Введение	5
1.	Задание	6
2.	Ход выполнения работы	6
2.1	Класс слабой кучи WeakHeap	6
2.1.1	Описание конструктора по умолчанию WeakHeap()	6
2.1.2	Описание конструктора WeakHeap(int count, std::istream&)	6
2.1.3	Описание конструктора WeakHeap(std::ifstream&)	6
2.1.4	Описание метода void displayArray()	6
2.1.5	Описание метода void displayHeap(int i_1, int j_1, int col, int num)	6
2.1.6	Описание метода void displayHeap()	7
2.1.7	Описание метода void weakHeapMerge(unsigned char *r, int i, int j, int num)	7
2.1.8	Описание метода WeakHeap* fileInputHeap()	7
2.1.9	Описание метода WeakHeap* consoleInputHeap()	7
2.1.10	Описание деструктора ~WeakHeap()	8
2.1.11	Описание метода void weakHeapSort()	8
2.2	Описание функции double log(int a, int b)	8
2.3	Описание алгоритма сортировки	8
	Заключение	10
	Список используемых источников	10
	Приложение А. Тестирование	11
	Приложение Б. Исходный код программы	31

## **ВВЕДЕНИЕ**

Целью работы являлось изучение сортировки методом слабой кучи. Для этого потребовалось изучить её структуру, алгоритм построения, алгоритм сортировки с помощью неё, а также придумать визуализацию работы алгоритма. Результатом является программа, которая считывает и сортирует исходный целочисленный массив, визуализируя работу алгоритма.

## 1. ЗАДАНИЕ

Вариант 31. Сортировка слабой кучей. Демонстрация.

## 2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

### 2.1. Класс слабой кучи WeakHeap

Для удобной работы со слабой кучей был создан класс слабой кучи WeakHeap. Публичными полями класса являются - `std::vector<int> whear` — хранятся элементы введённого массива, это вектор, встроенная возможность языка программирования, хранится в библиотеке `<vector>`, `int size` — хранится количество элементов `whear`, `unsigned char* r` — массив для хранения информации об обмене поддеревьями слабой кучи, `int s` — размер массива `r`.

Для класса реализованы публичные методы для работы со слабой кучей.

#### 2.1.1 Описание конструктора по умолчанию WeakHeap()

Конструктор слабой кучи по умолчанию, при помощи него можно создавать объекты слабой кучи, а потом уже вводить значения

#### 2.1.2 Описание конструктора WeakHeap(int count, std::istream&)

Конструктор класса, получает на вход целочисленное количество элементов, которые нужно ввести и ссылку на поток. При помощи цикла `for` вводит значения из потока во вспомогательную переменную `member`, а затем вставляет в конец вектора `whear`. Также здесь осуществлена проверка на некорректный ввод пользователя.

#### 2.1.3 Описание конструктора WeakHeap(std::ifstream&)

Конструктор класса, получает на вход ссылку на поток ввода. Вводим из файла количество элементов массива, проверяем, что оно не меньше нуля. При помощи цикла `for` вводит значения из потока во вспомогательную переменную `member`, а затем вставляет в конец вектора `whear`. Также здесь осуществлена проверка на некорректный ввод из файла и ввод пользователя.

#### 2.1.4 Описание метода void displayArray()

Метод выводит элементы массива, который хранится в поле класса WeakHeap — `std::vector <int> wheap`

### **2.1.5 Описание метода `void displayHeap(int i_1, int j_1, int col, int num)`**

Получает номера `i_1` и `j_1` элементов, которые нужно подсветить определённым цветом, `int col` — цвет, `int num` — количество отсортированных элементов в массиве. Сначала записываем элементы в кучу в порядке, удобном для вывода на экран, затем в зависимости от введённого цвета выводим первый элемент. Затем вычисляем глубину кучи при помощи функции `log(2, new_size)`, где `new_size` — количество элементов в массиве за вычетом отсортированных. Затем проходимся по элементам, выводим их в наглядном виде. У нас есть счётчик `int k` — количество уже выведенных элементов. Если его значение равно `i_1` или `j_1`, то перекрашиваем выводимый элемент при помощи управляющей последовательности `\x1b[<номер цвета>m`. Затем выводим отсортированную часть массива, крася её в жёлтый. При помощи `\x1b[33m`.

### **2.1.6 Описание метода `void displayHeap()`**

Метод нужен в случае, если мы просто хотим вывести кучи, без перекрашивания элементов. Точно так же сначала записываем элементы в кучу в порядке, удобном для вывода на экран, затем в зависимости от введённого цвета выводим первый элемент. Затем вычисляем глубину кучи при помощи функции `log(2, size)`. Затем проходимся по элементам, выводим их в наглядном виде. Счётчик `k` нужен, если количество элементов в слабой куче не будет равно степени двойки, без него вместо отсутствующих элементов на последнем уровне выведутся нули.

### **2.1.7 Описание метода `void weakHeapMerge(unsigned char *r, int i, int j, int num)`**

Если суперродитель меньше потомка, то для потомка переопределяем, порядок его потомков (кто "левый", а кто "правый"), затем меняем значения "суперродителя" и потомка при помощи `std::swap()`, выводим слабую кучу на экран для демонстрации, какие элементы могли поменяться.

### **2.1.8 Описание метода WeakHeap\* fileInputHeap()**

Метод возвращает указатель на объект класса слабой кучи, открывает файл для записи, создавая поток `std::ofstream` для чтения. Создаём новый объект класса слабой кучи, выводим на экран введенный массив, закрываем поток для чтения.

### **2.1.9 Описание метода WeakHeap\* consoleInputHeap()**

Метод возвращает указатель на объект класса слабой кучи, вводим с консоли количество элементов массива, проверяем, что оно не меньше нуля, иначе просим пользователя ввести количество ещё раз. Создаём новый объект класса слабой кучи, вводя элементы с консоли.

### **2.1.10 Описание деструктора ~WeakHeap()**

Деструктор класса, очищает вектор, в `wheap` котором хранятся элементы массива

### **2.1.11 Описание деструктора void weakHeapSort()**

Метод, в котором происходит сортировка.

## **2.2 Описание функции double log(int a, int b)**

Получает на вход два целочисленных числа `a` — основание логарифма и `b` — число, от которого надо взять логарифм., возвращает вещественное число - логарифм по основанию `a` от `b`. Для вычисления используется библиотечная функция `log(a)` — вычисляет логарифм по основанию 10 от переданного числа, функция входит в библиотеку `<cmath>`

## **2.3 Описание алгоритма сортировки**

Алгоритм сортировки - сначала формируем из массива слабую кучу: перебираем элементы массива слева-направо, для текущего элемента поднимаемся вверх по родительской ветке до ближайшего «правого» родителя, сравниваем текущий элемент и ближайшего правого родителя, если ближайший правый родитель меньше текущего элемента, то: меняем местами (левый  $\Leftrightarrow$  правый) поддеревья с потомками для узла, в котором находится текущий



элемент, меняем значениями ближайший «правый» родитель и узел с текущим элементом.

Затем из корня кучи текущий максимальный элемент перемещаем в конец неотсортированной части массива, после чего восстанавливаем слабую кучу: в корне кучи находится текущий максимальный элемент для неотсортированной части массива, меняем местами максимум из корня кучи и последний элемент в неотсортированной части массива. Последний элемент с максимумом перестаёт быть узлом слабой кучи. После этого обмена дерево перестало быть слабой кучей, так как в корне оказался не максимальный элемент. Поэтому делаем просейку: опускаемся из корня кучи по левым потомкам как можно ниже. Поднимаемся по левым потомкам обратно к корню кучи, сравнивая каждый левый потомок с корнем. Если элемент в корне меньше, чем очередной левый потомок, то: меняем местами (левый  $\Leftrightarrow$  правый) поддеревья с потомками для узла, в котором находится текущий левый потомок. Меняем значениями корень кучи и узел с текущим левым потомком. В корне слабой кучи снова находится максимальный элемент для оставшейся неотсортированной части массива. Затем снова из корня кучи текущий максимальный элемент перемещаем в конец неотсортированной части массива, восстанавливаем слабую кучу, повторяем процесс, пока не будут отсортированы все элементы.

Сложность алгоритма по времени —  $O(n \cdot \log n)$ .

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения работы была изучена сортировка методом слабой кучи. Была изучена структура слабой кучи, алгоритм её построения, алгоритм сортировки с помощью неё, а также визуализирована работа алгоритма. Была написана программа, которая считывает, сортирует исходный целочисленный массив и визуализирует работу алгоритма.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. The C++ Resources Network. URL: <http://www.cplusplus.com> (дата обращения: 15.11.2020)
2. Habr. URL: <https://habr.com/en/company/edison/blog/499786/> (дата обращения: 15.11.2020)

## ПРИЛОЖЕНИЕ А

### ТЕСТИРОВАНИЕ

Номер	Входные данные(без учёта выбора способа ввода)	Выходные данные	Комментарии
1		<p>Выберите способ ввода массива для сортировки  Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы  Синим будут подсвечиваться проверяемые в куче элементы  Зелёным будут подсвечиваться элементы, которые в результате сравнения поменяли местами  Жёлтым будут подсвечиваться отсортированные элементы</p> <p>Введите команду  4  Нет такой команды:  4  Введите команду  2  Вы завершили программу!</p>	Проверка на некорректных данных при вводе несуществующей команды
2	2 й 1 н 9 3	<p>Выберите способ ввода массива для сортировки  Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы  Синим будут подсвечиваться проверяемые в куче элементы  Зелёным будут подсвечиваться элементы, которые в результате сравнения поменяли местами  Жёлтым будут подсвечиваться отсортированные элементы</p>	Проверка на некорректных данных при вводе с консоли

		<p>Введите команду 1 Вы выбрали ввод с консоли Введите количество элементов в массиве 2 Введите элементы через пробел й Некорректное значение элемента. Введите все элементы снова 1 н Некорректное значение элемента. Введите все элементы снова 9 3 Построение первоначальной слабой кучи 9 3</p> <p>----- ----- Слабая куча построена 9 3</p> <p>----- ----- 9 3</p> <p>Меняем местами корень 9 и следующий за ним элемент 3 3 9</p> <p>----- ----- Отсортированный массив: 3 9</p> <p>Введите команду 2 Вы завершили программу!</p>	
3	2 9 1	<p>Выберите способ ввода массива для сортировки Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы</p>	<p>Проверка на некорректных данных при вводе из файла</p>

		<p>Синим будут подсвечиваться проверяемые в куче элементы</p> <p>Зелёным будут подсвечиваться элементы, которые в результате сравнения поменяли местами</p> <p>Жёлтым будут подсвечиваться отсортированные элементы</p> <p>Введите команду</p> <p>0</p> <p>Вы выбрали ввод из файла</p> <p>Нам нужно считать из файла 2 элементов</p> <p>Осталось считать 1 элементов</p> <p>9</p> <p>1 9</p> <p>Построение первоначальной слабой кучи</p> <p>1</p> <p>9</p> <p>Суперродитель 1 меньше потомка 9, меняем их местами</p> <p>9</p> <p>1</p> <p>-----</p> <p>-----</p> <p>Слабая куча построена</p> <p>9</p> <p>1</p> <p>-----</p> <p>-----</p> <p>9</p> <p>1</p> <p>Меняем местами корень 9 и следующий за ним элемент 1</p> <p>1</p> <p>9</p> <p>-----</p> <p>-----</p> <p>Отсортированный массив: 1 9</p> <p>Введите команду</p>	
--	--	--	--

		2 Вы завершили программу!	
4	7 1 9 7 5 4 8 2	<p>Выберите способ ввода массива для сортировки Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы <b>Синим</b> будут подсвечиваться проверяемые в куче элементы <b>Зелёным</b> будут подсвечиваться элементы, которые в результате сравнения поменяли местами <b>Жёлтым</b> будут подсвечиваться отсортированные элементы</p> <p>Введите команду 0 Вы выбрали ввод из файла Нам нужно считать из файла 7 элементов 1 9 7 5 4 8 2</p> <p>Построение первоначальной слабой кучи 1 <b>9</b> 7 5 4 8 <b>2</b></p> <p>-----</p> <p>----- 1 9 <b>7</b> 5 4 <b>8</b> 2</p> <p>Суперродитель 7 меньше потомка 8, меняем их местами 1 9 <b>8</b> 5 4 <b>7</b> 2</p> <p>-----</p> <p>----- <b>1</b> 9</p>	

		<div> <div>8 5</div> <div>4 7 2</div> <div>Суперродитель 1 меньше потомка 4, меняем их местами</div> <div>4</div> <div>9</div> <div>8 5</div> <div>1 7 2</div> <div>-----</div> <div>-----</div> <div>4</div> <div>9</div> <div>8 5</div> <div>1 7 2</div> <div>-----</div> <div>-----</div> <div>4</div> <div>9</div> <div>8 5</div> <div>1 7 2</div> <div>Суперродитель 4 меньше потомка 8, меняем их местами</div> <div>8</div> <div>9</div> <div>4 5</div> <div>1 7 2</div> <div>-----</div> <div>-----</div> <div>8</div> <div>9</div> <div>4 5</div> <div>1 7 2</div> <div>Суперродитель 8 меньше потомка 9, меняем их местами</div> <div>9</div> <div>8</div> <div>4 5</div> <div>1 7 2</div> <div>-----</div> <div>-----</div> </div>	
--	--	---	--



		<p>Слабая куча построена</p> <p>9 8 4 5 1 7 2</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку</p> <p>9 8 4 5 1 7 2</p> <p>Переместили корень 9 и элемент из конца неотсортированной части 2</p> <p>2 8 4 5 1 7 9</p> <p>-----</p> <p>-----</p> <p>2 8 4 5 1 7 9</p> <p>Суперродитель 2 меньше потомка 5, меняем их местами</p> <p>5 8 4 2 1 7 9</p> <p>-----</p> <p>-----</p> <p>5 8 4 2 1 7 9</p> <p>Суперродитель 5 меньше потомка 8, меняем их местами</p> <p>8 5</p>	
--	--	--	--

		<div> <div> <div>4 2</div> <div>1 7 9</div> </div> <div>-----</div> <div>-----</div> <div> <div>Переносим максимум из корня, применяем слабую просейку</div> <div>8</div> <div>5</div> <div>4 2</div> <div>1 7</div> </div> <div>Отсортированная часть массива: 9</div> <div> <div>Переместили корень 8 и элемент из конца неотсортированной части 7</div> <div>7</div> <div>5</div> <div>4 2</div> <div>1 8</div> </div> <div>Отсортированная часть массива: 9</div> <div>-----</div> <div>-----</div> <div> <div>7</div> <div>5</div> <div>4 2</div> <div>1 8</div> </div> <div>Отсортированная часть массива: 9</div> <div>-----</div> <div>-----</div> <div> <div>Переносим максимум из корня, применяем слабую просейку</div> <div>7</div> </div> </div>	
--	--	---	--

		<p>5 4 2 1</p> <p>Отсортированная часть массива: 8 9</p> <p>Переместили корень 7 и элемент из конца неотсортированной части 1</p> <p>1 5 4 2 7</p> <p>Отсортированная часть массива: 8 9</p> <p>-----</p> <p>-----</p> <p>1 5 4 2 7</p> <p>Отсортированная часть массива: 8 9</p> <p>Суперродитель 1 меньше потомка 4, меняем их местами</p> <p>4 5 1 2 7</p> <p>Отсортированная часть массива: 8 9</p> <p>-----</p> <p>-----</p> <p>4 5 1 2 7</p> <p>Отсортированная часть массива: 8 9</p> <p>Суперродитель 4 меньше потомка 5, меняем их местами</p> <p>5 4</p>	
--	--	--	--

		<p>1 2 7</p> <p>Отсортированная часть массива: 8 9</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку</p> <p>5 4 1 2</p> <p>Отсортированная часть массива: 7 8 9</p> <p>Переместили корень 5 и элемент из конца неотсортированной части 2</p> <p>2 4 1 5</p> <p>Отсортированная часть массива: 7 8 9</p> <p>-----</p> <p>-----</p> <p>2 4 1 5</p> <p>Отсортированная часть массива: 7 8 9</p> <p>Суперродитель 2 меньше потомка 4, меняем их местами</p> <p>4 2 1 5</p> <p>Отсортированная часть массива: 7 8 9</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку</p> <p>4 2 1</p>	
--	--	---	--

		<p>Отсортированная часть массива: 5 7 8 9</p> <p>Переместили корень 4 и элемент из конца неотсортированной части 1</p> <p>1</p> <p>2</p> <p>4</p> <p>Отсортированная часть массива: 5 7 8 9</p> <p>-----</p> <p>-----</p> <p>1</p> <p>2</p> <p>4</p> <p>Отсортированная часть массива: 5 7 8 9</p> <p>Суперродитель 1 меньше потомка 2, меняем их местами</p> <p>2</p> <p>1</p> <p>4</p> <p>Отсортированная часть массива: 5 7 8 9</p> <p>-----</p> <p>-----</p> <p>2</p> <p>1</p> <p>Отсортированная часть массива: 4 5 7 8 9</p> <p>Меняем местами корень 2 и следующий за ним элемент 1</p> <p>1</p> <p>2</p> <p>Отсортированная часть массива: 4 5 7 8 9</p> <p>-----</p> <p>-----</p> <p>Отсортированный массив: 1 2 4 5 7 8 9</p> <p>Введите команду</p> <p>2</p> <p>Вы завершили программу!</p>	
--	--	--	--

5	4 8 10 39 25	<p>Выберите способ ввода массива для сортировки  Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы  Синим будут подсвечиваться проверяемые в куче элементы  Зелёным будут подсвечиваться элементы, которые в результате сравнения поменяли местами  Жёлтым будут подсвечиваться отсортированные элементы</p> <p>Введите команду  0  Вы выбрали ввод из файла  Нам нужно считать из файла 4 элементов  8 10 39 25</p> <p>Построение первоначальной слабой кучи  8  10  39 25</p> <p>Суперродитель 10 меньше потомка 25, меняем их местами  8  25  39 10</p> <p>-----</p> <p>-----  8  25  39 10</p> <p>Суперродитель 8 меньше потомка 39, меняем их местами  39  25  8 10</p> <p>-----</p> <p>-----  39  25</p>	
---	-----------------	--	--

		<p>8 10</p> <p>-----</p> <p>-----</p> <p>Слабая куча построена</p> <p>39</p> <p>25</p> <p>8 10</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку</p> <p>39</p> <p>25</p> <p>8 10</p> <p>-----</p> <p>-----</p> <p>Переместили корень 39 и элемент из конца неотсортированной части 10</p> <p>10</p> <p>25</p> <p>8 39</p> <p>-----</p> <p>-----</p> <p>10</p> <p>25</p> <p>8 39</p> <p>-----</p> <p>-----</p> <p>10</p> <p>25</p> <p>8 39</p> <p>-----</p> <p>-----</p> <p>Суперродитель 10 меньше потомка 25, меняем их местами</p> <p>25</p> <p>10</p> <p>8 39</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку</p> <p>25</p>	
--	--	--	--

		<p>10 8</p> <p>Отсортированная часть массива: 39</p> <p>Переместили корень 25 и элемент из конца неотсортированной части 8</p> <p>8 10 25</p> <p>Отсортированная часть массива: 39</p> <p>-----</p> <p>-----</p> <p>8 10 25</p> <p>Отсортированная часть массива: 39</p> <p>Суперродитель 8 меньше потомка 10, меняем их местами</p> <p>10 8 25</p> <p>Отсортированная часть массива: 39</p> <p>-----</p> <p>-----</p> <p>10 8</p> <p>Отсортированная часть массива: 25 39</p> <p>Меняем местами корень 10 и следующий за ним элемент 8</p> <p>8 10</p> <p>Отсортированная часть массива: 25 39</p> <p>-----</p> <p>-----</p> <p>Отсортированный массив: 8 10 25 39</p> <p>Введите команду</p>	
--	--	--	--



		2 Вы завершили программу!	
6	4 98 45 76 21	<p>Выберите способ ввода массива для сортировки  Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы  Синим будут подсвечиваться проверяемые в куче элементы  Зелёным будут подсвечиваться элементы, которые в результате сравнения поменяли местами  Жёлтым будут подсвечиваться отсортированные элементы</p> <p>Введите команду  1  Вы выбрали ввод с консоли  Введите количество элементов в массиве  4  Введите элементы через пробел  98 45 76 21  Построение первоначальной слабой кучи  98  45  76 21</p> <p>-----  -----  98  45  76 21</p> <p>-----  -----  98  45  76 21</p> <p>-----  -----  Слабая куча построена  98</p>	

		<p>45 76 21</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку 98 45 76 21</p> <p>Переместили корень 98 и элемент из конца неотсортированной части 21 21 45 76 98</p> <p>-----</p> <p>-----</p> <p>21 45 76 98</p> <p>Суперродитель 21 меньше потомка 76, меняем их местами 76 45 21 98</p> <p>-----</p> <p>-----</p> <p>76 45 21 98</p> <p>-----</p> <p>-----</p> <p>Переносим максимум из корня, применяем слабую просейку 76 45 21</p> <p>Отсортированная часть массива: 98</p> <p>Переместили корень 76 и элемент из конца</p>	
--	--	---	--

		<p>неотсортированной части 21 21 45 76</p> <p>Отсортированная часть массива: 98</p> <p>-----</p> <p>-----</p> <p>21 45 76</p> <p>Отсортированная часть массива: 98</p> <p>Суперродитель 21 меньше потомка 45, меняем их местами</p> <p>45 21 76</p> <p>Отсортированная часть массива: 98</p> <p>-----</p> <p>-----</p> <p>45 21</p> <p>Отсортированная часть массива: 76 98</p> <p>Меняем местами корень 45 и следующий за ним элемент 21</p> <p>21 45</p> <p>Отсортированная часть массива: 76 98</p> <p>-----</p> <p>-----</p> <p>Отсортированный массив: 21 45 76 98 Введите команду 2 Вы завершили программу!</p>	
7	1 1	<p>Выберите способ ввода массива для сортировки Введите 1 для ввода с консоли, 0 для ввода</p>	

		<p>из файла, 2 для окончания работы программы</p> <p><b>Синим</b> будут подсвечиваться проверяемые в куче элементы</p> <p><b>Зелёным</b> будут подсвечиваться элементы, которые в результате сравнения поменяли местами</p> <p><b>Жёлтым</b> будут подсвечиваться отсортированные элементы</p> <p>Введите команду 1 Вы выбрали ввод с консоли Введите количество элементов в массиве 1 Введите элементы через пробел 1</p> <p>Отсортированный массив: 1 Введите команду 2 Вы завершили программу!</p>	
8	3 8 0 3	<p>Выберите способ ввода массива для сортировки</p> <p>Введите 1 для ввода с консоли, 0 для ввода из файла, 2 для окончания работы программы</p> <p><b>Синим</b> будут подсвечиваться проверяемые в куче элементы</p> <p><b>Зелёным</b> будут подсвечиваться элементы, которые в результате сравнения поменяли местами</p> <p><b>Жёлтым</b> будут подсвечиваться отсортированные элементы</p> <p>Введите команду 1 Вы выбрали ввод с консоли Введите количество элементов в массиве 3 Введите элементы через пробел 8 0 3</p> <p>Построение первоначальной слабой кучи</p>	

		<div>8</div> <div>0</div> <div>3</div> <div>-----</div> <div>-----</div> <div>8</div> <div>0</div> <div>3</div> <div>-----</div> <div>-----</div> <div>Слабая куча построена</div> <div>8</div> <div>0</div> <div>3</div> <div>-----</div> <div>-----</div> <div>Переносим максимум из корня, применяем слабую просейку</div> <div>8</div> <div>0</div> <div>3</div> <div>-----</div> <div>-----</div> <div>Переместили корень 8 и элемент из конца неотсортированной части 3</div> <div>3</div> <div>0</div> <div>8</div> <div>-----</div> <div>-----</div> <div>3</div> <div>0</div> <div>8</div> <div>-----</div> <div>-----</div> <div>3</div> <div>0</div> <div>-----</div> <div>Отсортированная часть массива: 8</div> <div>-----</div> <div>-----</div> <div>Меняем местами корень 3 и следующий за ним элемент 0</div>	
--	--	--	--

		0 3  Отсортированная часть массива: 8 ----- ----- Отсортированный массив: 0 3 8 Введите команду 2 Вы завершили программу!	
--	--	--	--

## ПРИЛОЖЕНИЕ Б

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: WeakHeap.cpp

```
#include "Weak_heap.h"
#include <limits>

#define GETFLAG(r, x) ((r[(x) >> 3] >> ((x) & 7)) & 1) //если в
качестве "левого" потомка родителя

#define TOGGLEFLAG(r, x) (r[(x) >> 3] ^= 1 << ((x) & 7)) //Для
потомка переопределяем, порядок его потомков
//(кто "левый", а кто "правый")

WeakHeap::~WeakHeap(){
    if (!wheap.empty())
        wheap.clear();
    delete this;
}

//вычисляет логарифм от b по основанию a
double log(int a, int b)
{
    return log(b) / log(a);
}

void WeakHeap::displayHeap(int i_1, int j_1, int col, int num){

    std::vector<int> heap1;
    heap1.push_back(wheap[0]);
    heap1.push_back(wheap[1]);
    for (int i=0;i<s;i++){//записываем элементы в кучу в порядке,
удобном для вывода на экран
        heap1.push_back(wheap[2*i+r[i]]);
        heap1.push_back(wheap[2*i+1-r[i]]);
    }
    if (col==0){
        if (i_1==0 || j_1==0)
            std::cout<<"\x1b[32m"<<wheap[0]<<"\x1b[0m";
        else
            std::cout<<wheap[0];
    }
    else
    {
        if (i_1==0 || j_1==0)
            std::cout<<"\x1b[34m"<<wheap[0]<<"\x1b[0m";
        else
            std::cout<<wheap[0];
    }
}
```

```

}

std::cout<<"\n";
int new_size=size-num;

int depth=(int)log(2, new_size);//вычисляем глубину дерева
if ((int)log(2, new_size)!=pow(2, depth))
    depth+=1;

int k=0;

for (int i=0;i<depth;i++){

    for (int j=0;j<pow(2,i);j++){
        if (k<new_size)
            if (col==0){//проверяем, какого цвета выводить элементы
                if (k+1==i_1 || k+1==j_1){
                    std::cout<<"\x1b[32m"<<wheap[k+1]<<"\x1b[0m"<<"
";

                }

                else{
                    std::cout<<wheap[k+1]<<" ";

                }
            }
        else
        {
            if (k+1==i_1 || k+1==j_1){
                std::cout<<"\x1b[34m"<<wheap[k+1]<<"\x1b[0m"<<" ";

            }

            else{
                std::cout<<wheap[k+1]<<" ";

            }
        }
        k++;
    }

    std::cout<<"\n";

}

if (num>1){
    std::cout<<"\n";
    std::cout<<"Отсортированная часть массива: ";
    for (int i=1;i<num;i++)
        std::cout<<"\x1b[33m"<<wheap[new_size+i]<<"\x1b[0m"<<" ";
    std::cout<<"\n";
}

```



```

    }

    if (col==0)

std::cout<<"-----\n";
    else
        std::cout<<"\n";
}

void WeakHeap::displayHeap(){

    std::vector <int> heap1;
    heap1.push_back(wheap[0]);
    heap1.push_back(wheap[1]);
    for (int i=0;i<s;i++){//записываем элементы в кучу в порядке,
        удобном для вывода на экран
        heap1.push_back(wheap[2*i+r[i]]);
        heap1.push_back(wheap[2*i+1-r[i]]);

    }
    std::cout<<wheap[0];
    std::cout<<"\n";

    int depth=(int)log(2, size);//вычисляем глубину дерева
    if ((int)log(2, size)!=pow(2, depth))
        depth+=1;

    int k=0;

    for (int i=0;i<depth;i++){

        for (int j=0;j<pow(2,i);j++){
            if (k<size-1)
                std::cout<<wheap[k+1]<<" ";
            k++;
        }
        std::cout<<"\n";
    }

std::cout<<"-----\n";
}

void WeakHeap::weakHeapMerge(unsigned char *r, int i, int j, int
num) {

    if (wheap[i] < wheap[j]) {//"Суперродитель" меньше потомка?
        //Для потомка переопределяем, порядок его потомков
        //(кто "левый", а кто "правый")

```

```

        TOGGLEFLAG(r, j);
        //Меняем значения "суперродителя" и потомка
        this->displayHeap(i, j, 1, num);
        std::cout<<"Суперродитель "<<wheap[i]<<" меньше потомка
"<<wheap[j]<<"; меняем их местами\n";

        std::swap(wheap[i], wheap[j]);
        this->displayHeap(i, j, 0, num);

    }
    else{
        this->displayHeap(i, j, 1, num);

std::cout<<"-----\n";
    }
}

void WeakHeap::weakHeapSort() {
    int n = size;
    int lef;
    int per;
    if(n > 1) {

        int i, j, x, y, Gparent;
        s = (n + 7) / 8;
        r = new unsigned char [s];

        //Массив для обозначения, какой у элемента
        //потомок "левый", а какой "правый"
        for(i = 0; i < n / 8; ++i)
            r[i] = 0;

        std::cout<<"Построение первоначальной слабой кучи\n";

        //Построение первоначальной слабой кучи
        for(i = n - 1; i > 0; --i) {
            j = i;
            //Поднимаемся на сколько возможно вверх,
            //если в качестве "левого" потомка родителя
            lef=GETFLAG(r, j >> 1);
            while ((j & 1) == lef) {
                j = j>> 1;
                lef=GETFLAG(r, j >> 1);
            }
            //И ещё на один уровень вверх как "правый" потомок
            родителя
            Gparent = j >> 1;
            //Слияние начального элемента, с которого
            //начали восхождение до "суперродителя"
            weakHeapMerge(r, Gparent, i, 1);

```

```

    }

    //Перенос максимума из корня в конец -->
    //слабая просейка --> и всё по новой
    std::cout<<"Слабая куча построена\n";
    this->displayHeap();

    for(i = n - 1; i >= 2; --i) {
        std::cout<<"Переносим максимум из корня, применяем слабую
просейку\n";
        //Максимум отправляем в конец неотсортированной части
массива
        //Элемент из конца неотсортированной части попадает в
корень
        this->displayHeap(0, i, 1, n-i);
        std::cout<<"Переместили корень "<<wheap[0]<<" и элемент из
конца неотсортированной части "<<wheap[i]<<"\n";
        std::swap(wheap[0], wheap[i]);
        this->displayHeap(0, i, 0, n-i);
        x = 1;
        //Опускаемся жадно вниз по "левым" веткам
        lef=GETFLAG(r, x);
        while((y = 2 * x + lef) < i) {
            x = y;
            lef=GETFLAG(r, x);
        }
        //Поднимаемся по "левой" ветке обратно до самого верха
        //попутно по дороге делаем слияние каждого узла с корнем
        while(x > 0) {
            weakHeapMerge(r, 0, x, n-i);
            x >>= 1;
        }
    }
    //Последнее действие - меняем местами корень
    //и следующий за ним элемент
    this->displayHeap(0, 1, 1, n-1);
    std::cout<<"Меняем местами корень "<<wheap[0]<<" и следующий
за ним элемент "<<wheap[1]<<"\n";
    std::swap(wheap[0], wheap[1]);
    this->displayHeap(0, 1, 0, n-1);
    delete[] r;
}
}
void WeakHeap::displayArray()
{
    for (int i=0;i<size;i++)
        std::cout<<wheap[i]<<" ";
    std::cout<<"\n\n";
}

WeakHeap* WeakHeap::consoleInputHeap(){
    int count=0;
    std::cout<<"Введите количество элементов в массиве\n";

```

```

std::cin>>count;

while (count<=0||!std::cin){

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    try{//проверяем корректность ввода количества элементов
        std::cout<<"Введите количество\n";
        std::cin>>count;

        if (!std::cin)
            throw "Некорректное значение количества элементов, введите целое положительное число\n";
    }
    catch(const char* s){
        std::cout<<s;
        std::cin.clear();
    }

}

std::cout<<"Введите элементы через пробел\n";
WeakHeap* wh=new WeakHeap(count, std::cin);
return wh;
}

WeakHeap* WeakHeap::fileInputHeap(){
    int count=0;
    std::ifstream file("input.txt");
    while(file.fail())
        try{
            //открываем файл для чтения
            if (file.fail())
                file.close();
            throw "Не удалось открыть файл\n";

        }
        catch(const char* s){
            std::cout<<s;
            std::string r;
            std::ifstream file("input.txt");
            std::cout<<"Исправьте файл, чтобы он открывался и сообщите об этом программе, введя какое-нибудь сообщение\n";
            std::cin>>r;

        }
    WeakHeap* wh= new WeakHeap(file);
    wh->displayArray();
    file.close();
    return wh;
}

```

```

WeakHeap::WeakHeap(std::ifstream& file){
    int count=0;

    file>>count;

    while (count<=0||!std::cin){

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\\
n');
        try{//проверяем корректность ввода количества элементов
            std::cout<<"Введите количество\\n";
            std::cin>>count;

            if (!std::cin)
                throw "Некорректное значение количества элементов,
введите целое положительное число\\n";
        }
        catch(const char* s){
            std::cout<<s;
            std::cin.clear();
        }

    }
    std::cout<<"Нам нужно считать из файла "<< count<< "
элементов\\n";
    size=count;
    int i=0;
    int f=0;
    int member=0;

    while (!file.eof())&&i<count)
    {
        file>>member;
        wheap.push_back(member);
        i+=1;
    }

    while (i<count){//проверяем, что считали из файла достаточное
количество элементов{
        count=size-i;
        if (count!=0)
            std::cout<<"Осталось считать "<<count<<" элементов\\n";
            while (!f){

                try{//проверяем корректность ввода количества
элементов
                    for (int i=0;i<count;i++){
                        std::cin>>member;
                        wheap.push_back(member);
                    }
                    if (!std::cin){
                        wheap.clear();

```

```

        count=size;
        i=0;
        throw "Некорректное значение элемента.
Введите все элементы снова\n";
    }
    else{
        f=1;
        i=count;
    }
}
catch(const char* s){
    std::cout<<s;
    std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
}
}
file.close();
}

WeakHeap::WeakHeap(int count, std::istream& stream){

    int member=0;
    size=count;
    int f=0;
    int count_input_elem=0;
    int i=0;

    while (!f){

        try{//проверяем корректность ввода количества элементов
            for (int i=0;i<count;i++){
                stream>>member;
                wheap.push_back(member);
            }
            if (! stream){
                wheap.clear();
                throw "Некорректное значение элемента. Введите все
элементы снова\n";
            }
            else
                f=1;
        }
        catch(const char* s){
            std::cout<<s;
            std::cin.clear();

stream.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        }
    }
}

```

```
}
```

Название файла: WeakHeap.h

```
#ifndef WEAK_HEAP_H
#define WEAK_HEAP_H

#include <iostream>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <iterator>
#include <fstream>
#include <cstring>
#include <cmath>

class WeakHeap{

public:

    std::vector <int> wheap;
    int size;
    unsigned char* r=nullptr;
    int s;
    WeakHeap(){}
    WeakHeap(int, std::istream&);
    WeakHeap(std::ifstream&);

    void displayArray();
    void displayHeap(int i, int j, int col, int num);
    void displayHeap();
    void weakHeapMerge(unsigned char *r, int i, int j, int num);
    void weakHeapSort();
    WeakHeap* fileInputHeap();
    WeakHeap* consoleInputHeap();
    ~WeakHeap();

};

#endif
```

Название файла: main.cpp

```
#include "Weak_heap.h"

int main()
{

    char s;
    WeakHeap* wh=nullptr;
```

```

    int flag=0;
    while (!flag){
        std::cout<<"Выберите способ ввода массива для сортировки\
n";
        std::cout<<"Введите 1 для ввода с консоли, 0 для ввода из
        файла, 2 для окончания работы программы\n";
        std::cout<<"\x1b[34mСиним\x1b[0m будут подсвечиваться
        проверяемые в куче элементы\n";
        std::cout<<"\x1b[32mЗелёным\x1b[0m будут подсвечиваться
        элементы, которые в результате сравнения поменяли местами\n";
        std::cout<<"\x1b[33mЖёлтым\x1b[0m будут подсвечиваться
        отсортированные элементы\n\n";
        std::cout<<"Введите команду\n";
        std::cin>>s;
        switch (s)
        {
            case '0':
                std::cout<<"Вы выбрали ввод из файла\n";
                wh=wh->fileInputHeap();
                if (wh)
                    wh->weakHeapSort();

                std::cout<<"Отсортированный массив: ";
                wh->displayArray();
                break;
            case '1':
                std::cout<<"Вы выбрали ввод с консоли\n";
                wh=wh->consoleInputHeap();
                if (wh)
                    wh->weakHeapSort();
                std::cout<<"Отсортированный массив: ";
                wh->displayArray();
                break;
            case '2':
                std::cout<<"Вы завершили программу!\n";
                flag=1;
                break;
            default:
                std::cout<<"Нет такой команды: \n"<<s<<"\n";

                break;
        }
    }

    return 0;
}

```

Название файла: input.txt

3

9 3 1



