**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 16-17**

**Дисциплина:** Основы алгоритмизации и программирования

**Тема:** Алгоритм сортировки «быстрая»

**Цель:** Получить практические навыки разработки алгоритмов и их программной реализации.

**Выполнил: студент группы 201-723**

Карпушкин Сергей Евгеньевич

(Фамилия И.О.)



**Дата, подпись** 24.12.2020\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2020**

Оглавление

[Цель 3](#_Toc59749866)

[Задача 3](#_Toc59749867)

[Идея алгоритма 3](#_Toc59749868)

[Словесное представление алгоритма 3](#_Toc59749869)

[Блок-схема с использованием элемента “решение” 5](#_Toc59749870)

[Блок-схема с использованием элемента “модификация” 6](#_Toc59749871)

[Исходный код программы “Сортировка гномья с циклом while” 7](#_Toc59749872)

[Исходный код программы “Сортировка гномья с циклом for” 8](#_Toc59749874)

# Цель

Получить практические навыки разработки алгоритмов и их программной реализации.

# Задача

Необходимо выполнить и оформить описание следующих пунктов:

1. Сформулировать идею алгоритма
2. Выполнить словесное представление алгоритма
3. Выполнить полнить представление алгоритма с помощью блок схем с использованием элемента модификации и без него.
4. Выполнить программную реализацию алгоритмов на языке С с использованием параметрического цикла и цикла с предусловием.

# Идея алгоритма

Выбираем из массива элемент, называемый опорным, и запоминаем его значение. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.

Далее начинаем двигаться от начала массива по возрастающей, а потом от конца массива по убывающей. Цель: переместить в правую часть элементы больше опорного, а в левую – элементы меньше опорного. Если во время движения по возрастающей находится элемент со значением больше опорного, то мы выходим из цикла, прибавляем единицу к индексу элемента, на котором остановились, и переходим к циклу с движением по убывающей. В этом цикле мы остаемся до тех пор, пока не находится элемент со значением меньше опорного. Как только такой элемент найден, мы отнимаем единицу от его индекса, и меняем значение элемента со значением элемента, на котором мы остановились в предыдущем цикле. Делаем так до тех пор, пока индекс левого элемента (найденного в первом цикле) меньше либо равен индексу правого элемента (найденного во втором цикле). В итоге получаем два подмассива (от начала до индекса правого элемента и от индекса левого элемента до конца). С этими подмассивами мы рекурсивно проделываем все то же самое, что и с большим массивом до тех пор, пока все элементы окончательно не отсортируются.

# Словесное представление алгоритма

array – массив, piv – номер опорного элемента, b – индекс первого элемента массива, e – индекс последнего элемента массива

1 Номер опорного элемента равен (b+e)/2

2 Начало курсор. Если l <= r (где l = b, a r = e), то переходим к пункту 3, иначе к пункту 13

3 Если array[l] < piv, то переходим к пункту 4, иначе к пункту 6

4 Прибавляем единицу к индексу левого элемента (l++).

5 Переходим к пункту 3

6 Если array[r] > piv, то переходим к пункту 7, иначе к пункту 9

7 Убавляем на единицу индекс правого элемента (r--)

8 Переходим к пункту 6

9 Если l <= r, то переходим к пункту 10, иначе переходим к пункту 13

10 Меняем значения элементов с индексами l и r местами (array[l] и array[r])

11 Прибавляем единицу к индексу левого элемента и убавляем на единицу индекс правого элемента.

12 Переходим к пункту 2

13 Если b < r (индекс первого элемента массива меньше индекса правого элемента массива), то переходим к пункту 14, иначе к пункту 15

14 Вызов курсор: (array[], b, r).

15 Если l < e (индекс левого элемента меньше индекса последнего элемента массива), то переходим к пункту 16, иначе к пункту 17

16 Вызов курсор: (array[], l, e).

17 Конец алгоритма

# Блок-схема с использованием элемента “решение”



Рисунок 1 - Блок-схема с использованием элемента “решение”

# Блок-схема с использованием элемента “модификация”



Рисунок 2 - Блок-схема с использованием элемента “модификация”

# Исходный код программы “Сортировка быстрая с циклом while”

Листинг 1 - Исходный код программы “Сортировка быстрая с циклом while”

#include <stdio.h>

void qsort(int\* arr, int b, int e) {

if (b < e) {

int buf, l = b, r = e, piv = arr[(b + e) / 2]; // объявляем необходимые переменные и выбираем опорный элемент

while (l <= r) {

while (arr[l] < piv) // Движемся от начала массива по возрастающей

l++;

while (arr[r] > piv) // Движемся от конца массива по убывающей

r--;

if (l <= r) { // Меняем элементы справа и слева от опорного

int t = arr[l];

arr[l] = arr[r];

arr[r] = t;

l++;

r--;

}

}

qsort(arr, b, r); // Та же сортировка в правом подмассиве

qsort(arr, l, e); // И в левом

}

}

int main()

{

int arr[] = { 5, 7, 8, 4, 9, 1, 3, 6, 2 }; // объявление массива

int N = sizeof(arr) / sizeof(int); // длина массива

qsort(arr, 0, N - 1); // сортировка

for (int i = 0; i < N; i++) // вывод упорядоченного массива

printf("%d ", arr[i]);

}

# Результат работы

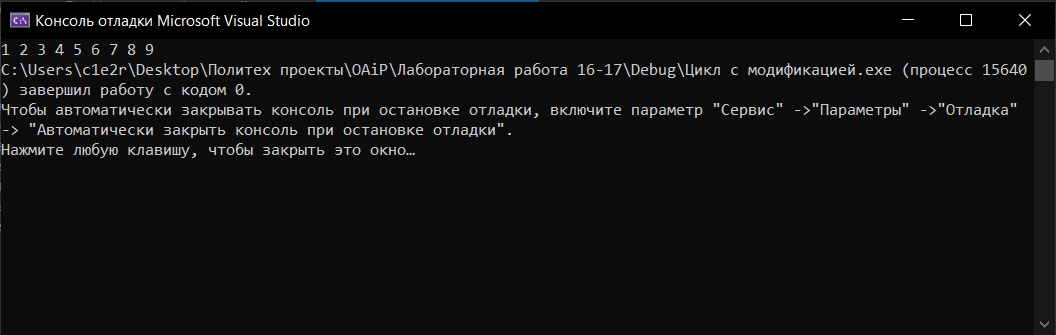


Рисунок 3 – результат работы программы “Сортировка быстрая с циклом while”

# Исходный код программы “Сортировка быстрая с циклом for”

Листинг 2 - Исходный код программы “Сортировка быстрая с циклом for”

#include <stdio.h>

void qsort(int\* arr, int b, int e) {

if (b < e) {

int buf, l = b, r = e, piv = arr[(b + e) / 2]; // объявляем необходимые переменные и выбираем опорный элемент

for (; l <= r;) {

while (arr[l] < piv) // Движемся от начала массива по возрастающей

l++;

while (arr[r] > piv) // Движемся от конца массива по убывающей

r--;

if (l <= r) { // Меняем элементы справа и слева от опорного

int t = arr[l];

arr[l] = arr[r];

arr[r] = t;

l++;

r--;

}

}

qsort(arr, b, r); // Та же сортировка в правом подмассиве

qsort(arr, l, e); // И в левом

}

}

int main()

{

int arr[] = { 5, 7, 8, 4, 9, 1, 3, 6, 2 }; // объявление массива

int N = sizeof(arr) / sizeof(int); // длина массива

qsort(arr, 0, N - 1); // сортировка

for (int i = 0; i < N; i++) // вывод упорядоченного массива

printf("%d ", arr[i]);

}

# Результат работы

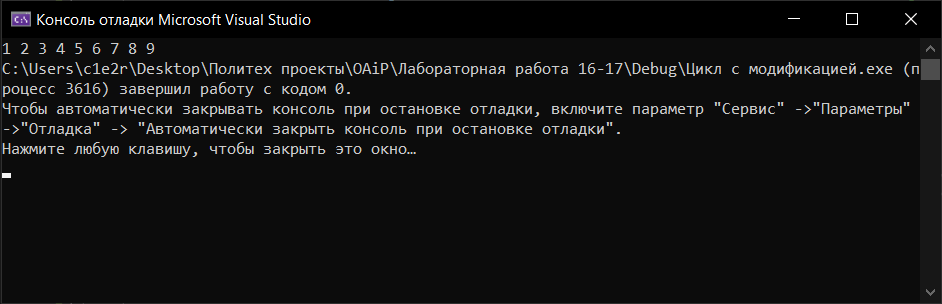


Рисунок 4 – результат работы программы “Сортировка быстрая с циклом for”