|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ДИСЦИПЛИНА Анализ алгоритмов

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**по теме: «Алгоритмы сортировки»**

Студент Бутолин Александр Алексеевич

Группа ИУ7-52

Москва, 2018 г.

**ОГЛАВЛЕНИЕ**

[Введение 2](#__RefHeading___Toc3021_2540108226)

[1. Аналитическая часть 4](#__RefHeading___Toc3023_2540108226)

[1.1. Описание алгоритмов 4](#__RefHeading___Toc3025_2540108226)

[2. Конструкторская часть 6](#__RefHeading___Toc3027_2540108226)

[2.1. Сравнение алгоритмов 6](#__RefHeading___Toc3029_2540108226)

[2.2. Разработка алгоритмов 6](#__RefHeading___Toc3031_2540108226)

[2.3 Вывод 10](#__RefHeading___Toc3033_2540108226)

[3. Технологическая часть 11](#__RefHeading___Toc3035_2540108226)

[3.1. Требования к программному обеспечению 11](#__RefHeading___Toc3037_2540108226)

[3.2. Средства реализации ПО 11](#__RefHeading___Toc3039_2540108226)

[3.3. Листинг кода 12](#__RefHeading___Toc3041_2540108226)

[3.4. Вывод 13](#__RefHeading___Toc3043_2540108226)

[4. Экспериментальная часть 14](#__RefHeading___Toc3045_2540108226)

[4.1. Примеры работы 14](#__RefHeading___Toc3047_2540108226)

[4.2. Постановка эксперимента 15](#__RefHeading___Toc3049_2540108226)

[4.3. Сравнительный анализ на материале экспериментальных данных 15](#__RefHeading___Toc3051_2540108226)

[4.4. Вывод 18](#__RefHeading___Toc3053_2540108226)

[Заключение 19](#__RefHeading___Toc3055_2540108226)

[Список используемых источников 20](#__RefHeading___Toc3057_2540108226)

# **Введение**

Алгоритмы сортировки находят широкое применение во многих сферах, например, для упорядочивания результатов в научных исследованиях. Поэтому сортировки являются одной из наиболее обширных и проработанных областей информатики. Одним из важнейших критериев хорошего алгоритма сортировки является его быстродействие. Поэтому, основной задачей людей, занимающихся разработкой данных алгоритмов -- создание наиболее быстрого алгоритма расчёской и прямыми включениями.

Цель работы: реализовать алгоритмы заявленных сортировок, провести сравнительный анализ их трудоемкости.

**Цель работы**: изучение методов сортировки массивов на примере трех алгоритмов:

* Сортировка прямым выбором;
* сортировка расчёской;
* сортировка прямыми включениями.

**Задачи работы:**

1. получение практических навыков реализации указанных алгоритмов;
2. сравнительный анализ реализаций указанных алгоритмов;
3. экспериментальное подтверждение различий во временной эффективности указанных алгоритмов при помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся размерах массивов;

# **Аналитическая часть**

Задача сортировки формулируется следующим образом: дана последовательность элементов

(1)

Требуется упорядочить элементы по не убыванию или по не возрастанию – найти перестановку ключей (1), либо по неубыванию:

(2)

либо по не возрастанию:

(3)

## **Описание алгоритмов**

* + 1. **Сортировка прямым выбором**

На каждом i-ом шаге алгоритма находим минимальный i-ый элемент и меняем его с j-ым элементом массива. Таким образом будет получен массив, отсортированный по не убыванию.

* + 1. **Сортировка Расчёской**

Сортировка расческой является улучшенной модификацией сортировки пузырьком. В сортировке пузырьком, когда сравниваются два элемента, промежуток (расстояние друг от друга) равен 1. Основная идея сортировки расчёской в том, что этот промежуток может быть гораздо больше, чем единица

* + 1. **Сортировка прямыми включениями**

[Алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8) в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

# **Конструкторская часть**

В этом разделе представлены схемы трех описанных в аналитической части алгоритмов – сортировки прямым выбором, расческой и прямыми включениями. Также приведено их теоретическое сравнение на основе вычисления приблизительной трудоемкости для каждого из алгоритмов.

## **2.1. Сравнение алгоритмов**

Сортировка прямым выбором работает за O(n2) во всех случаях. [1]

Сортировка расческой будем иметь сложность:

Худшее время: O(n2)

Лучшее время: O(n\*logn)

Сортировка включениями:

Худший случай: O(n2)

Лучший случай: O(n) [3].

## **2.2. Разработка алгоритмов**

Прежде чем реализовывать алгоритм на выбранном языке программирования, требуется представить его в виде схемы для более наглядного представления всех этапов вычислений и упрощения его практической реализации.

В соответствии с описанием алгоритмов сортировки, представленных в аналитическом разделе данной работы, были построены схемы трех реализаций указанных алгоритмов.

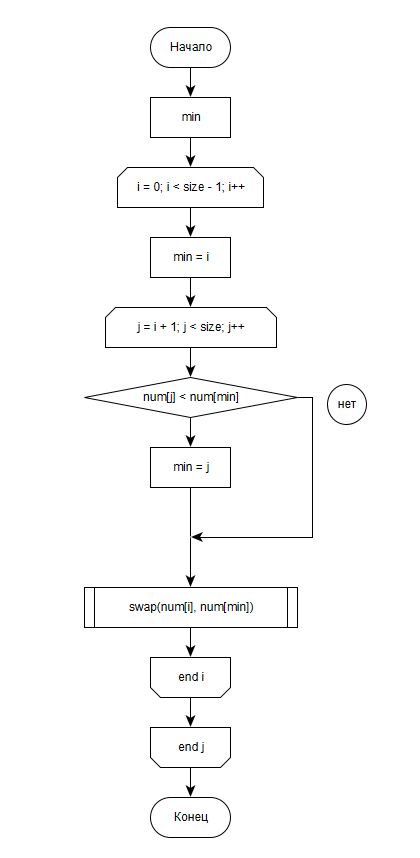


Рисунок 2.1.1. – схема алгоритма сортировки выбором

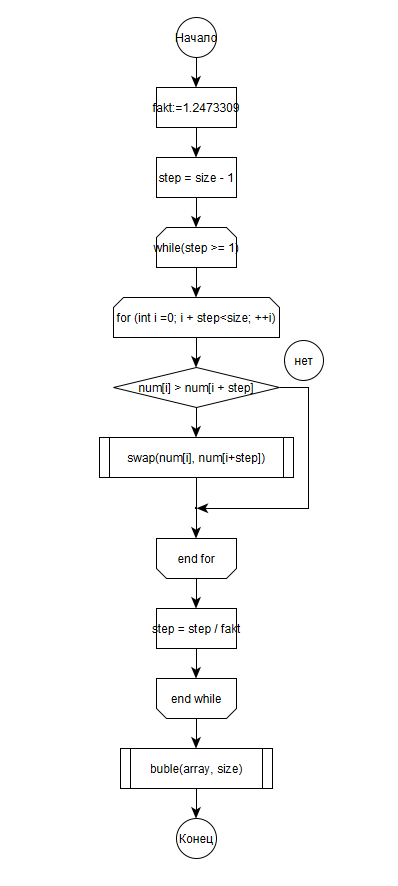


Рисунок 2.1.2. – схема алгоритма сортировки расческа

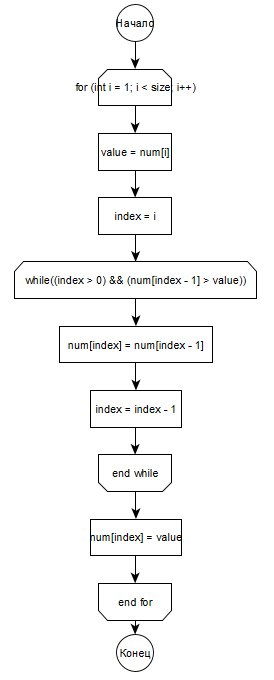


Рисунок 2.1.3. – схема алгоритма сортировки вставками

## **Вывод**

В данном разделе в соответствии с теоретическим описанием алгоритмов, представленным в аналитическом разделе, были построены схемы указанных алгоритмов. Также было проведено их теоретическое сравнение на основе вычисления приблизительной оценки трудоемкости для каждого алгоритма.

# **Технологическая часть**

В аналитическом разделе представлены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также листинг кода программы.

## **3.1. Требования к программному обеспечению**

Программное обеспечение должно реализовывать три алгоритма сортировки массивов. Пользователь должен иметь возможность проводить как единичное упорядочивание введенного массива, так и использовать программное обеспечение для проведения многократных замеров на заранее подготовленных данных с целью сравнения указанных алгоритмов.

Разработанное ПО должно предоставлять возможность замеров процессорного времени выполнения реализации каждого алгоритма. Требуется провести замеры для варьирующихся размеров массивов: не менее чем от 1000 до 10000 с шагом 1000. Один эксперимент ставится не менее 100 раз, результат одного эксперимента рассчитывается как средний из результатов проведенных испытаний с одинаковыми входными данными.

## **Средства реализации ПО**

Для реализации поставленной задачи был использован язык программирования C++.

Для измерения процессорного времени была использована ассемблерная команда rdtsc (read time stamp counter) из библиотеки <intrin.h>, который считает количество тактов процессора с момента последнего сброса.

## **Листинг кода**

Ниже приведен листинг кода для трех реализованных алгоритмов – сортировки прямым выбором, сортировки расческой и сортировки вставками.

// Функция сортировки прямым выбором

void selectionSort(vector<int> &num, int size)

{

int min, temp;

for (int i = 0; i < size - 1; i++)

{

min = i;

for (int j = i + 1; j < size; j++)

{

if (num[j] < num[min])

min = j;

}

temp = num[i];

num[i] = num[min];

num[min] = temp;

}

}

//Функция сортировки расчёской

void sortc(vector<int> &num, int size)

{

double fakt = 1.2473309; // фактор уменьшения

double step = size - 1;

while (step >= 1)

{

for (int i = 0; i + step < size; ++i)

{

if (num[i] > num[i + step])

{

swap(num[i], num[i + step]);

}

}

step /= fakt;

}

// сортировка пузырьком

for (int i = 0; i < size - 1; i++)

{

bool swapped = false;

for (int j = 0; j < size - i - 1; j++)

{

if (num[j] > num[j + 1]) {

swap(num[j], num[j + 1]);

swapped = true;

}

}

if (!swapped)

break;

}

}

// Функция сортировки прямыми вставками

void inclusionSort(vector<int> &num, int size)

{

// Для всех элементов кроме начального

for (int i = 1; i < size; i++)

{

int value = num[i]; // запоминаем значение элемента

int index = i; // и его индекс

while ((index > 0) && (num[index - 1] > value))

{ // смещаем другие элементы к концу массива пока они меньше index

num[index] = num[index - 1];

index--; // смещаем просмотр к началу массива

}

num[index] = value; // рассматриваемый элемент помещаем на освободившееся место

}

## **Вывод**

На основе схем алгоритмов, представленных в конструкторском разделе, в соответствии с указанными требованиями к реализации с использованием средств языка C++ было разработано программное обеспечение, содержащее реализации трех указанных в задании алгоритмов сортировки.

# **Экспериментальная часть**

В экспериментальном разделе представлены примеры работы разработанного программного обеспечения, а также подробный сравнительный анализ реализованных алгоритмов на основе экспериментальных данных, полученных при тестировании.

## **4.1. Примеры работы**

Ниже приведены примеры работы для различных граничных случаев с указанием входных массивов, ожидаемого и полученного результата сортировки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Случай | Вход | Ожидаемый  результат | Сортировка  Простым выбором | Сортировка расческа | Сортировка прямыми вставками |
| Сортировка | [1,2,5,7] | [1,2,5,7] | [1,2,5,7] | [1,2,5,7] | [1,2,5,7] |
| Массив из один.  эл-тов | [2,2,2,2] | [2,2,2,2] | [2,2,2,2] | [2,2,2,2] | [2,2,2,2] |
| Массив в случ. порядке | [9,-1,3,4] | [-1,3,4,9] | [-1,3,4,9] | [-1,3,4,9] | [-1,3,4,9] |

## **4.2. Постановка эксперимента**

Для проведения сравнительного анализа времени работы трех реализованных алгоритмов были проведены замеры времени сортировки на размерах массивов в диапазоне от 1000 до 10000 с шагом в 1000. Для каждого массива в указанном диапазоне было проведено по 100 замеров с использованием всех трех алгоритмов с целью получения более достоверного результата для среднего времени выполнения.

Все замеры проводились на следующем оборудовании, описанном ниже.

1. Процессор – Intel Core i5, 1.60 GHz
2. Оперативная память – DDR3 8 Гб;
3. Операционная система – Windows 10 Домашняя x64.

## **4.3. Сравнительный анализ на материале экспериментальных данных**

Результаты сравнения трех алгоритмов для массивов в диапазоне от 1000 до 10000 с шагом 1000, заполненных случайным образом, приведены в таблице 4.3.1.

Таблица 4.3.1.

Сравнение времени работы трех алгоритмов сортировки на массивах, заполненных случайным образом, в тиках

|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | Сортировка прямым выбором | Сортировка расческа | Сортировка прямыми включениями |
| 1000 | 1902252 | 464897 | 244041 |
| 2000 | 7450176 | 994844 | 696883 |
| 3000 | 16671020 | 1575399 | 1554012 |
| 4000 | 29748014 | 2168414 | 2762497 |
| 5000 | 46432198 | 2825271 | 4325035 |
| 6000 | 66916493 | 3441751 | 5985256 |
| 7000 | 91110979 | 4132186 | 8211600 |
| 8000 | 120239443 | 4711124 | 10964467 |
| 9000 | 150866368 | 5447581 | 13707879 |
| 10000 | 186715430 | 6068523 | 18393782 |



Рисунок 4.3.1. – диаграмма зависимости времени работы алгоритмов от размера массивов, заполненных случайным образом

## **4.4. Вывод**

Как видно из полученных результатов, сортировка прямым выбором является самой медленной. Сортировка прямыми включениями работает быстрее сортировки расчески на малых длинах массивов (до 3000 элементов включительно), но на больших длина (больше 3000 элементов) сортировка расческа показала результат лучше, нежели сортировка прямыми включениями.

# **Заключение**

В процессе выполнения лабораторной работы было проведено исследование алгоритмов сортировок. Также был выполнен сравнительный анализ указанных алгоритмов.

При помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся размерах массивов были экспериментально подтверждены различия во временной эффективности выбранных алгоритмов.

# **Список используемых источников**

1. «Сортировка прямым выбором» [электронный ресурс] <https://neerc.ifmo.ru/wiki/index.php?title=Сортировка_вставками> (Дата обращения: 05.11.2018 г.)
2. «Сортировка расческа» [электронный ресурс] http://algolab.valemak.com/merge-simple (Дата обращения: 05.11.2018 г.)
3. «Сортировка прямыми включениями» [электронный ресурс] http://kvodo.ru/sortirovka-shella.html, свободный. (Дата обращения: 05.11.2018 г.)