Министерство образования и науки Российской Федерации

**ФГАОУ ВО «РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ» (RUDN University )**

Инженерная академия

Институт Космических Технологий

Департамент механики и мехатроники

**КУРСОВАЯ РАБОТА**

**По дисциплине:** Программирование и основы алгоритмизации

**По теме:** Разбиение заданной последовательности чисел на два непересекающихся подмножества, дисперсии в которых наиболее близки друг к другу. Сортировка выбором по убыванию

Выполнил: ст-т. гр. ИУБ-103

Кормилицын Артем

Проверил: Рындин Д.А

Дата:

Подпись:

Москва, 2018

**Содержание**

ВВЕДЕНИЕ ………………………………………………………………..

ФОРМУЛИРОВКА ЗАДАЧИ ……………………………………………..

ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ……………………………………

ИНТЕРФЕЙС ПРОГРАММЫ……………………………………………...

СОРТИРОВКА ВЫБОРОМ………………………………………………..

ДИСПЕРСИЯ МНОЖЕСТВА ЧИСЕЛ ……………………………………

ЗАКЛЮЧЕНИЕ ………………………………………………………………

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…………….…………..

ПРИЛОЖЕНИЕ А – Исходный код программы…………………………….

ПРИЛОЖЕНИЕ Б – Блок-схема программы……………………………….

**Введение**

Данные являются сырьем для работы компьютерных программ. Массивы данных это одна из разновидностей способов передачи информации. В современном мире обработка данных играет одно из главенствующих ролей после искусственного интеллекта и постоянного увеличения вычислительных мощностей. Объектом исследования в данной курсовой работе является одномерный массив целочисленных элементов. Процессор компьютера воспринимает два числа – это 0 и 1, поэтому объект исследования является актуальным для современного общества, т.к человек ежедневно сталкивается с потоком чисел, по-другому это можно назвать – потоком данных. Результаты исследований массива чисел можно легко масштабировать под любые задачи, связанные с потоками данных. Массивы можно создавать вручную, но для чистоты исследований было предложено генерировать одномерный массив целочисленных элементов в произвольной форме, т.е рандомно. С массивами можно проводить различные операции сортировки. Пузырьковая сортировка, сортировка выбором, сортировка Хоара, сортировки слиянием и вставками. Методы сортировки разрабатывались на заре первой половины 20 века, в 40-е годы Джон фон Нейман разработал метод сортировки слиянием, тогда это была необходимо для тестирования ряда команд на одном из первых двоичных компьютеров EDVAC, в этом же году немецкий инженер Конрад Цузе придумал программу для сортировки методом простой вставки. К этому времени уже появились быстрые специализированные сортировальные машины, в сопоставлении с которыми и оценивалась эффективность разрабатываемых ЭВМ. Первым опубликованным обсуждением сортировки с помощью вычислительных машин стала лекция [Джона Мокли](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BA%D0%BB%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD), прочитанная им в 1946 году. Мокли показал, что сортировка может быть полезной также и для численных расчетов, описал методы сортировки [простой вставки и бинарных вставок](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0%D0%BC%D0%B8), а также [поразрядную сортировку](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) с частичными проходами. Мы приходим к логичному выводу, сортировка важнейший инструмент любого программиста, и методы различных сортировок активно используются спустя десятки лет, не потеряв своей актуальности.

Целью курсовой работы является изучение методов сортировки, а также создание проекта с помощью программы c++ builder либо альтернативной программы visual studio, где можно проводить различные операции с массивами.

**Формулировка задачи**

Вариант индивидуального задания №33 :

**Разбить заданную последовательность чисел на два непересекающихся подмножества , дисперсии в которых наиболее близки друг к другу .**

Вариант сортировки №18 :

**Сортировка выбором по убыванию**

**Задание**

Приложение должно обеспечить:

* Генерацию одномерного массива из M случайных целых чисел в диапазоне от Q до P
* Отображение элементов массива на форме в табличном виде (String Grid)
* Сохранение элементов массива в текстовый файл
* Считывание чисел из текстового файла в массив
* Возможность коррекции этих чисел путем редактирования в String Grid
* Сортировка массива **выбором по убыванию**
* Возможность полного автоматического перемешивания элементов массива
* Возможность разбиения заданного массива на два подмножества, в которых дисперсии будут максимально близки друг к друг
* Вывод найденного подмножества на экран и запись ее в другой текстовый файл

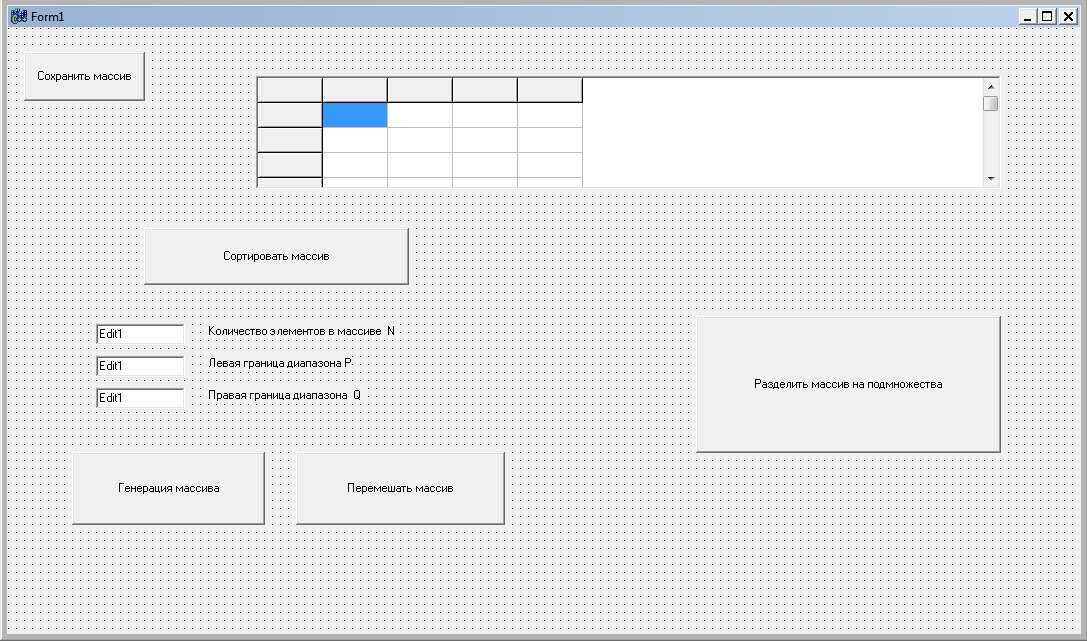
Приложение должно включать:

* Удобный и понятный интерфейс
* ФИО студента, номер группы, номер варианта, приведенные в отдельном модальном окне
* Меню, содержащее пункты, обеспечивающие удобную работу с приложением, например пункт «Файл» может содержать подпункты «Создать», «Открыть», «Сохранить». Пункт «Данные» - «Ввод диапазона», «Коррекция данных» и т.д
* Обработка исключительных ситуаций, обеспечивающую стабильную работу приложения при вводе любых данных, неправильном или отсутствующем файле данных и т.д

**Описание работы программы**

Программа выполнена в среде разработки C++ Builder. Каждая кнопка – это отдельный программный код. Цель программы – решение различных алгоритмических и математических задач связанных с множествами целочисленных значений. Поля ввода чисел подписаны лейблами, таким образом в первое поле вводятся количество элементов массива, данное число не может быть отрицательным, во второе поле вводится левая граница диапазона, а в третье вводится правая граница диапазона, таким образом второе и третье поле для ввода – это интервал случайных значений массива. Далее нажмем кнопку «Генерация массива», в верхнем поле dataGrid появится массив случайных чисел из заданного диапазона с фиксированным количеством элементов, которое было указано пользователем. Теперь можем нажать на кнопку «Найти подмножества», во втором поле dataGrid появятся два новых массива, которые между собой не пересекаются, т.е каждый элемент массива уникальный и разброс значений в этих массивах максимально близок друг к другу. Можем сохранить полученные два непересекающихся подмножества исходного массива в текстовые файлы в формате txt. Далее кнопка «Перемешать массив» случайным образом изменит положение элементов в ячейках исходного массива. При этом, мы можем заметить, что значение и количество элементов не изменились, а изменилось только положение значений в клетках. Нажмем еще раз на кнопку «Найти подмножества» и мы увидим аналогичный результат, что и до перемешивания, это доказывает истинность выражений описанных выше.

Резюмируя, можно сказать, что в программе можно генерировать массив случайных чисел, исходный массив можно перемешивать, данное множество можно разделять на два непересекающихся подмножества дисперсии которых будут максимально близки друг к другу. Все массивы из dataGrid можно сохранять в файлы с расширением txt, и редактировать значение элементов массива прямо в таблице dataGrid.   
Ниже приведены иллюстрации интерфейса программы.



**Интерфейс программы**

2

1

Информация о студенте

**ГЕНЕРАЦИЯ МАССИВА**

Количество элементов в массиве

Левая граница диапазона массива

Правая граница диапазона массива

Таблица StringGrid\_1, где отображается генерируемый массив

Таблица StringGrid\_2, где отображаются найденные подмножества

**НАЙТИ ПОДМНОЖЕСТВА**

Файлы

Открыть файл

Сохранить в

3

**Сортировать массив**

7

8

4

6

5

**ПЕРЕМЕШАТЬ МАССИВ**

Программа состоит из следующих компонентов:

1. Кнопка, по нажатию которой, появляется информация о студенте;
2. Кнопка «Файл» позволяет открывать файл в формате txt, в котором находится массив случайных чисел. Также в этой кнопке есть дополнительные функции, а именно сохранение массивов в файлы в формате txt;
3. Таблицы StringGrid , где отображаются полученные массивы, и в которых можно редактировать значения в ячейках таблицы;
4. Поля Edit1, Edit2, Edit3, позволяющие вводить следующие параметры: Количество элементов в массиве, левый предел диапазона простых чисел, правый предел диапазона простых чисел;
5. Кнопка генерации случайного массива N элементов в диапазоне от P до Q
6. Кнопка перемешивания элементов массива
7. Кнопка, которая подразделяет массив на два подмножества дисперсии в которых максимально близки друг к другу.
8. Кнопка «Сортировать массив» проводит сортировку массива выбором по убыванию

**Сортировка выбором**

Сортировка выбором – возможно, самый простой в реализации алгоритм сортировки. Как и в большинстве других подобных алгоритмов, в его основе лежит операция сравнения. Сравнивая каждый элемент с каждым, и в случае необходимости производя обмен, метод приводит последовательность к необходимому упорядоченному виду.

Исходный массив длиной N разбивается на две части: итог и остаток. Участок массива, называемый **итогом**, располагается с начала массива и должен быть упорядоченным, а участок массива, называемый **остатком**, располагается вплотную за итогом и содержит исходные числа не отсортированной части исходного массива.

Текстуальный алгоритм сортировки выбором:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.

Шаг 2. В остатке массива ищется минимальный элемент и он меняется местом с первым элементом остатка (i-ым элементом массива). После чего значение i увеличивается на единицу, тем самым расширяя итоговый участок массива ( отсортированную часть исходного массива).

Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

Блок-схема алгоритма методом сортировки выбора представлена в приложении-Б в разделе блок-схемы программы.

Тестовый код сортировки на примере с заданным массивом:

**#include** <algorithm> // для std::swap. В C++11 используйте заголовок <utility>

**#include** <iostream>

**#include** <stdlib.h>

**#include** <time.h> // чтобы вывести srand

**#include**<iomanip>// нужна для того чтобы массив вывести строкой

**using** **namespace** std;

**int** **main**()

{

**const** **int** length = 6;

**int** array [length] = {10, 20, 50, 40, 30, 70};

// Перебираем каждый элемент массива

// (кроме последнего, он уже будет отсортирован к тому времени, когда мы до него доберемся)

**for** (**int** startIndex = 0; startIndex < length - 1; ++startIndex)

{

// В переменной smallestIndex хранится индекс наименьшего значения, которое мы нашли в этой итерации

// Начинаем с того, что наименьший элемент в этой итерации - это первый элемент (индекс 0)

**int** smallestIndex = startIndex;

// Затем ищем элемент поменьше в остальной части массива

**for** (**int** currentIndex = startIndex + 1; currentIndex < length; ++currentIndex)

{

// Если мы нашли элемент, который меньше нашего наименьшего элемента

**if** (array[currentIndex]> array[smallestIndex])

// запоминаем его

smallestIndex = currentIndex;

}

// smallestIndex теперь наименьший элемент

// меняем местами наше начальное наименьшее число с тем, которое мы обнаружили

swap(array[startIndex], array[smallestIndex]);

}

// Теперь, когда весь массив отсортирован - выводим его на экран

**for** (**int** index = 0; index < length; ++index)

cout << array[index] << ' ';

**return** 0;

}

**Дисперсия множества чисел**

Дисперсия смотрит разброс, как отклоняется мат ожидание. Мат ожидание – это дискретное распределение. Это сумма i равная от одного до бесконечности. Так как у нас последовательность – это дискретные величины, когда мы проходим случайные величины – есть такое понятие, как случайные величины, а есть такое понятие, как непрерывные величины.

X – элемент массива, случайная величина

– среднее значение, по-другому мат ожидание

– количество элементов в массиве

Таким образом, данные формулы будут проверять соответствие дисперсий двух образованных подмножеств.

­­

**Заключение**

Таким образом курсовая работал сделана согласно заданию. Был проведен информационный поиск в сети Интернет по теме дисперсия дискретных величин, программирование форм в c++ builder, сортировка выбором по убыванию. Мною были разобраны такие понятия, как объектно-ориентированное программирование, наследование, инкапсуляция и полиморфизм, деструктор и конструктор. Я научился сортировать выбором по убыванию и возрастанию массивы данных, научился проводить разбиение заданного массива на непересекающиеся подмножества дисперсии в которых будут максимально блики друг к другу. Я получил не только первоначальные навыки программирования на языке C++, но и еще я получил ценнейший опыт работы в таких средах разработке как Visual Studio 2017 Community, C++ Builder 6, QT Creator.

**Список используемых источников**

1. ГОСТ 7.32-2001.

2. ГОСТ Р 7.0.5-2008 СИБИД.

# 3. 7-е издание. — М.: Вильямс, 2013. — 688 с. — ISBN 978-5-8459-1825-3. Сиддхартха Рао. Освой самостоятельно C++ за 21 день

4. Бхаргава А.

Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб.: Питер, 2017. – 288 с.: и. – (Серия «Библиотека программиста»). ISBN 978-5-496-02541-6

5. Кибзун А.И., Горяинова Е. Р., Наумов А. В., Сиротин А. Н.

Теория вероятностей и математическая статистика. Базовый курс с примерами и задачами / Учебн. пособие. – М.: ФИЗМАТЛИТ, 2002. – 224 с. – ISBN 5-9221-0231-1

6. Wikipedia.com

7. YouTube.com

8. cyberforum.ru

9. habrhabr.com

**Приложение**

**Приложение – А Раздел Исходный код программы**

**#include** < vcl.h >

**#include** < algorithm >

**#include** < fstream >

**#include** < iostream >

**#include** < math.h >

**#pragma** hdrstop

**#include** "Unit1.h"

**#include** "Unit2.h"

//---------------------------------------------------------------------------

**#pragma** package(smart\_init)

**#pragma** resource "\*.dfm"

TForm1 \* Form1;

FILE \* MyFile;

AnsiString fName = "";

AnsiString fName2 = "";

**int** \* b;

**int** P = 0, Q = 0, N = 0;

**bool** error, Open = **false**;

**int** \* a;

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Button1Click(TObject \* Sender) {

Q = StrToInt(Form2 - > Edit1 - > Text);

P = StrToInt(Form2 - > Edit2 - > Text);

N = StrToInt(Form2 - > Edit3 - > Text);

free(a);

a = (**int** \* ) malloc(N \* **sizeof**(**int**));

StringGrid1 - > ColCount = N + 1;

StringGrid1 - > RowCount = 4;

StringGrid1 - > Cells[0][0] = "Индекс";

StringGrid1 - > Cells[0][1] = "Массив";

StringGrid1 - > Cells[0][2] = "Делятся на K";

StringGrid1 - > Cells[0][3] = "Остальные";

**if** (N > 0 && N < 30) {

srand(time(NULL));

**for** (**int** i = 0; i < N; i++)

{

StringGrid1 - > Cells[i + 1][0] = IntToStr(i + 1);

a[i] = rand() % (P - Q + 1) + Q;

StringGrid1 - > Cells[i + 1][1] = IntToStr(a[i]);

}

error = **true**;

} **else**

ShowMessage("Ошибка, введите правильные данные!");

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Button2Click(TObject \* Sender)

{

**if** (error)

{

**for** (**int** i = 0; i < N; i++)

a[i] = StrToInt(StringGrid1 - > Cells[i + 1][1]);

**int** leftMark = 1;

**int** rightMark = N - 1;

**while** (leftMark <= rightMark)

{

**for** (**int** i = rightMark; i >= leftMark; i--)

**if** (a[i - 1] < a[i]) std::swap(a[i], a[i - 1]);

leftMark++;

**for** (**int** i = leftMark; i <= rightMark; i++)

**if** (a[i - 1] < a[i]) std::swap(a[i], a[i - 1]);

rightMark--;

}

**for** (**int** i = 0; i < N; i++)

{

StringGrid1 - > Cells[i + 1][1] = IntToStr(a[i]);

}

} **else**

ShowMessage("Ошибка, введите правильные данные!");

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Creator1Click(TObject \* Sender)

{

**if** (OpenDialog1 - > Execute()) {

free(b);

b = **new** **int**[30];

N = 0;

fName2 = OpenDialog1 - > FileName;

MyFile = fopen(fName2.c\_str(), "r");

**for** (**int** i = 0; !feof(MyFile); i++) {

N++;

**if** (N > 30) {

ShowMessage("Нельзя считать больше");

N--;

**break**;

}

fscanf(MyFile, "%i ", & b[i]);

}

**if** (!Open) {

StringGrid1 - > ColCount = N + 1;

StringGrid1 - > RowCount = 4;

StringGrid1 - > Cells[0][0] = "Индекс";

StringGrid1 - > Cells[0][1] = "Массив";

StringGrid1 - > Cells[0][2] = "Делятся на K";

StringGrid1 - > Cells[0][3] = "Остальные";

}

free(a);

Open = **true**;

a = (**int** \* ) malloc(N \* **sizeof**(**int**));

**for** (**int** i = 0; i < N; i++) {

a[i] = b[i];

}

**for** (**int** i = 0; i < N; i++)

{

StringGrid1 - > Cells[i + 1][0] = IntToStr(i + 1);

StringGrid1 - > Cells[i + 1][1] = IntToStr(a[i]);

}

error = **true**;

**delete**[] b;

fclose(MyFile);

}

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Button3Click(TObject \* Sender)

{

**if** (error) {

**for** (**int** i = 0; i < N; i++)

a[i] = StrToInt(StringGrid1 - > Cells[i + 1][1]);

**for** (**int** i = 0; i < N; i++) {

std::swap(a[i], a[rand() % (N - i) + i]);

}

**for** (**int** i = 0; i < N; i++)

{

StringGrid1 - > Cells[i + 1][1] = IntToStr(a[i]);

}

} **else**

ShowMessage("Ошибка, введите правильные данные!");

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::SaveFile1Click(TObject \* Sender)

{

**int** \* bb;

bb = (**int** \* ) malloc(N \* **sizeof**(**int**));

**for** (**int** i = 0; i < N; i++)

bb[i] = StrToInt(StringGrid1 - > Cells[i + 1][1]);

**if** (fName != "") {

MyFile = fopen(fName.c\_str(), "w");

**for** (**int** i = 0; i < N; i++)

fwprintf(MyFile, L "%i ", bb[i]);

fclose(MyFile);

} **else**

**if** (SaveDialog1 - > Execute())

{

fName = SaveDialog1 - > FileName;

MyFile = fopen(fName.c\_str(), "w");

**for** (**int** i = 0; i < N; i++)

fwprintf(MyFile, L "%i ", bb[i]);

fclose(MyFile);

}

free(bb);

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Button4Click(TObject \* Sender)

{

**for** (**int** i = 0; i < N; i++)

StringGrid1 - > Cells[i + 1][2] = " ";

**for** (**int** i = 0; i < N; i++)

StringGrid1 - > Cells[i + 1][3] = " ";

**for** (**int** i = 0; i < N; i++)

a[i] = StrToInt(StringGrid1 - > Cells[i + 1][1]);

**int** key = StrToInt(Form2 - > Edit4 - > Text);

**int** \* c = **new** **int**[N];

**int** \* d = **new** **int**[N];

**int** \* mnoj = **new** **int**[30 \* key];

**int** \* bin = **new** **int**[N];

**int** index = 0, index1 = 0, total, sum = 0, end, point;

**bool** flag, stop;

index = 0;

**for** (**int** i = 0; i < 30 \* key; i++) {

mnoj[i] = (i + 1) \* key;

}

**for** (**int** i = 0; i < N; i++) {

sum += a[i];

bin[i] = 0;

c[i] = 0;

d[i] = 0;

}

sum /= 2;

**for** (**int** i = 0; i < key; i++) {

**if** (sum % key != 0) {

sum++;

}

}

**for** (**int** i = 0; sum >= mnoj[i]; i++) {

end = i;

}

**int** k = std::powf(2, N); //количество подмножеств

**do** {

point = mnoj[end];

**for** (**int** i = 0; i < k; i++) {

total = 0;

flag = **false**;

**for** (**int** j = 0; j < N; j++) {

**if** (i & (1 << j))

{

total += a[j]; // накапливаем сумму

flag = **true**; // подмножество не пусто

bin[j] = 1;

}

}

**if** (total == point && flag) // если сумма совпала и подмножество не пусто

{

stop = **false**;

**break**;

} **else** {

**for** (**int** j = 0; j < N; j++) {

bin[j] = 0; //обнуление бинарного массива

}

}

}

end--;

} **while** (stop == **true** && end >= 0);

**for** (**int** i = 0; i < N; i++) {

**if** (bin[i] == 1) {

d[index1] = a[i];

index1++;

} **else** {

c[index] = a[i];

index++;

}

}

**for** (**int** i = 0; i < index; i++) {

StringGrid1 - > Cells[i + 1][2] = IntToStr(c[i]);

}

**for** (**int** j = 0; j < index1; j++) {

StringGrid1 - > Cells[j + 1][3] = IntToStr(d[j]);

}

**delete**[] bin;

**delete**[] mnoj;

**delete**[] c;

**delete**[] d;

}

fastcall TForm1::**TForm1**(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Creator2Click(TObject \*Sender)

{

AnsiString info = "Выполнил Кормилицын Артем \nВариант-33 Соритировка-18" ;

ShowMessage(info);

}

//---------------------------------------------------------------------------

**void** \_\_fastcall TForm1::Enterrange1Click(TObject \*Sender)

{

Form2 -> Show();

}

//---------------------------------------------------------------------------

**Приложение-Б Раздел Блок-схемы**

Начало

i = 0;

i < N – 1; i++

min = a[i]

j = I + 1 j<N

i++

min > a[j]

min = a[j]

Конец

a[i] = min

Нет

Нет

Нет

Да

Да

Да

Рис. 1

Схема алгоритма сортировки методом выбора