

# Что такое алгоритмы?



ПРЕПОДАВАТЕЛЬ



# Артем Гордийчук

**Full-stack software engineer**

- Более 8 лет опыта работы
- Java, Spring, Hibernate, AWS, Oracle, PostgreSQL
- Проекты связанные с банковской, финансовой деятельностью, e-commerce

[artemsgor@gmail.com](mailto:artemsgor@gmail.com)

[www.linkedin.com/in/artem-g-48071a61](https://www.linkedin.com/in/artem-g-48071a61)



# ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

# ПЛАН ЗАНЯТИЯ

1. Повторение изученного
2. Вопросы по повторению
3. Основной блок
4. Вопросы по основному блоку
5. Задание для закрепления
6. Практическая работа
7. Оставшиеся вопросы

# АЛГОРИТМЫ, ЗАЧЕМ?

Знание алгоритмов позволяет:

- Быть квалифицированным специалистом в своей профессии
- Иметь карьерный рост
- Уважение коллег
- Эффективно решать поставленные задачи

# АЛГОРИТМЫ, ЗАЧЕМ?

Примеры:

- Протоколы маршрутизации в коммуникационных сетях используют классические алгоритмы поиска кратчайшего пути.
- Шифрование с открытым ключом опирается на эффективные теоретико-числовые алгоритмы.
- Компьютерная графика задействует вычислительные примитивы, которые предоставляют геометрические алгоритмы.
- Индексация в базах данных опирается на структуры данных сбалансированных деревьев поиска.
- Алгоритмы применяют для парсинга данных, фильтрации дубликатов, отрисовки динамических списков, хранения и вывода оповещений для пользователя.



TEL-RAN  
by Starta Institute

1

# ПОВТОРЕНИЕ ИЗУЧЕННОГО

# Вспомним:

- Массивы
- Одномерный массив
- Максимальный и минимальный индекс в массиве

Номера (индексы) элементов массива

0 1 2 3 4 5 6 7

Arr

5	8	10	12	-9	10	11	-3
---	---	----	----	----	----	----	----

Название массива

Элементы массива



# Экспресс-опрос

- **Вопрос 1.**

Как в памяти хранятся массивы?

- **Вопрос 2.**

Какого типа данные, могут храниться в массивах ?



# Вспомним:

Алгоритм: Сортировка пузырьком (Bubble sorting)

Описание: последовательно сравнивать значения соседних элементов и менять местами элементы, если предыдущий больше последующего.

Таким образом элементы с большим значением оказываются в конце списка, а с меньшим в начале.



# Экспресс-опрос

- **Вопрос 1.**

Если искомый элемент всегда лежит в конце списка, будет ли эффективным алгоритм линейного поиска, для нахождения элемента?

Объясните почему “да” или “нет”.

- **Вопрос 2.**

Как вы думаете из сортировки “пузырьком” можно ли сделать сортировку “камнем”?





TEL-RAN  
by Starta Institute

# 2

## ВОПРОСЫ ПО ПОВТОРЕНИЮ

# Введение

- Что такое алгоритмы
- Характеристики алгоритмов
- Свойства алгоритмов
- Типы алгоритмов
- Как создавать алгоритмы
- Способы описания алгоритмов
- Элементы блок-схем
- Синтаксис псевдо-кода
- Эффективность алгоритма
- Преимущества и недостатки алгоритмов
- Примеры





TEL-RAN  
by Starta Institute

# 3

## ОСНОВНОЙ БЛОК

# Что такое алгоритмы

- Алгоритм означает набор правил, которым необходимо следовать при вычислениях или других операциях по решению задач.
- Алгоритм относится к последовательности конечных шагов для решения конкретной проблемы.
- Алгоритмы могут быть простыми и сложными в зависимости от того, чего вы хотите достичь.
- Алгоритмизация – процесс разработки алгоритма для решения какой-либо задачи



# Характеристики алгоритмов

Ясный и не двусмысленный	Четко определенные входные данные	Четко определенные результаты	Конечный	Выполнимый	Независимый от языка
Каждый шаг алгоритма должен быть ясен во всех аспектах и должен вести только к одному смыслу.	Если алгоритм говорит принимать входные данные, это должны быть четко определенные входные данные.	Алгоритм должен четко определять, какой результат будет получен, и он также должен быть четко определен.	Алгоритм должен быть конечным, т.е. он должен завершаться через конечное время.	Алгоритм должен быть простым, универсальным и практичным, чтобы его можно было выполнить с доступными ресурсами.	Алгоритм должен быть независимым от языка, т. е. это должны быть простые инструкции, которые могут быть реализованы на любом языке, и при этом вывод будет таким же, как и ожидалось.



# Свойства алгоритмов

- Алгоритм должен завершиться через конечное время.
- Алгоритм должен принимать ноль или более входных данных.
- Алгоритм должен давать один и тот же результат для одного и того же входного случая.
- Каждый шаг в алгоритме должен быть эффективным.



# Экспресс-опрос

- **Вопрос 1.**

Как вы поняли, что такое алгоритм?

- **Вопрос 2.**

Какие характеристики алгоритмов вы запомнили?

- **Вопрос 3.**

Может ли результат одного и того же входного случая быть разным?



# Типы алгоритмов

- Алгоритм грубой силы.
- Рекурсивный алгоритм.
- Алгоритм поиска с возвратом.
- Алгоритм поиска.
- Алгоритм сортировки.
- Алгоритм хеширования.
- Алгоритм «разделяй и властвуй».
- Жадный алгоритм.
- Алгоритм динамического программирования.
- Рандомизированный алгоритм.



# Как создавать алгоритмы

- Проблема, которая должна быть решена с помощью этого алгоритма, т.е. четкое определение проблемы.
- При решении проблемы необходимо учитывать все ограничения.
- Входные данные, которые необходимо принять для решения этой проблемы.
- Ожидаемый результат после решения проблемы.
- Решение этой проблемы находится в рамках заданных ограничений.



# Способы описания алгоритмов

Существует три основных способа описания алгоритма:

Текстовый

Графический

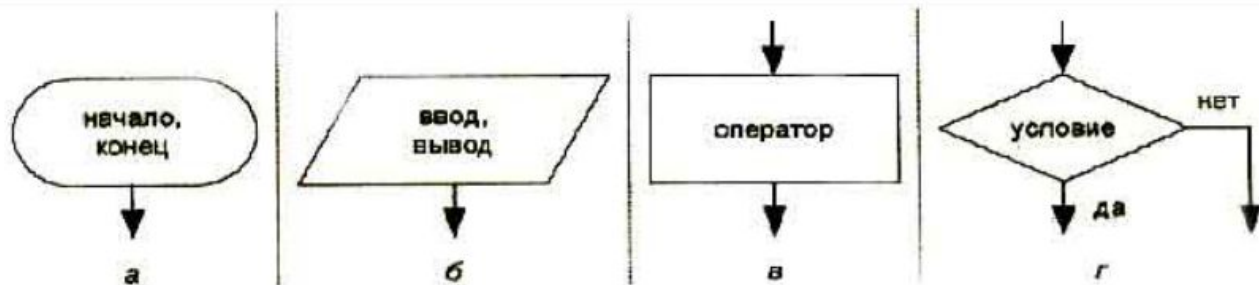
Псевдокод

Расписать шаги  
алгоритма  
последовательно в  
тексте

Изобразить  
графически в виде  
блок-схем

Специальный  
символьный язык

# Основные элементы блок-схем



а — начало (конец) алгоритма

б — блок ввода/вывода

в — операционный блок

г — логический (условный) блок

д — цикл с параметром (для параметра цикла указывается его начальное и конечное значение, шаг равен единице)



# Синтаксис псевдокода

Псевдокод = алгоритмический язык

START / END - начало / конец алгоритма

WRITE / READ - ввод / вывод данных

IF THEN ELSE - выбор

FOR / WHILE / REPEAT - итерация(Циклы)

# Экспресс-опрос

- **Вопрос 1.**

За что отвечает этот элемент блок-схемы?



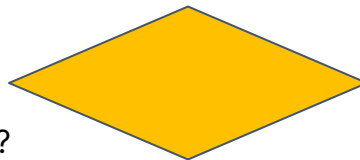
- **Вопрос 2.**

Для чего используется данный элемент блок-схемы?



- **Вопрос 3.**

Когда необходимо использовать следующий элемент блок-схемы?





# Эффективность алгоритма

Чтобы стандартный алгоритм был хорошим, он должен быть эффективным. Следовательно, эффективность алгоритма должна проверяться и поддерживаться.

- Фактор времени : время измеряется путем подсчета количества ключевых операций.
- Фактор пространства : пространство измеряется путем подсчета максимального объема памяти, требуемого алгоритмом.



# Преимущества алгоритмов

- Алгоритм легко понять.
- Алгоритм — это пошаговое представление решения данной задачи.
- В алгоритме проблема разбивается на более мелкие части или шаги, поэтому программисту легче преобразовать ее в настоящую программу.



# Недостатки алгоритмов

- Написание алгоритма занимает много времени.
- Понимание сложной логики с помощью алгоритмов может быть очень трудным.
- Операторы ветвления и цикла трудно показать в алгоритме.



# ПРИМЕРЫ

# Текстовый, словесный алгоритм

## Алгоритм сравнения переменных:

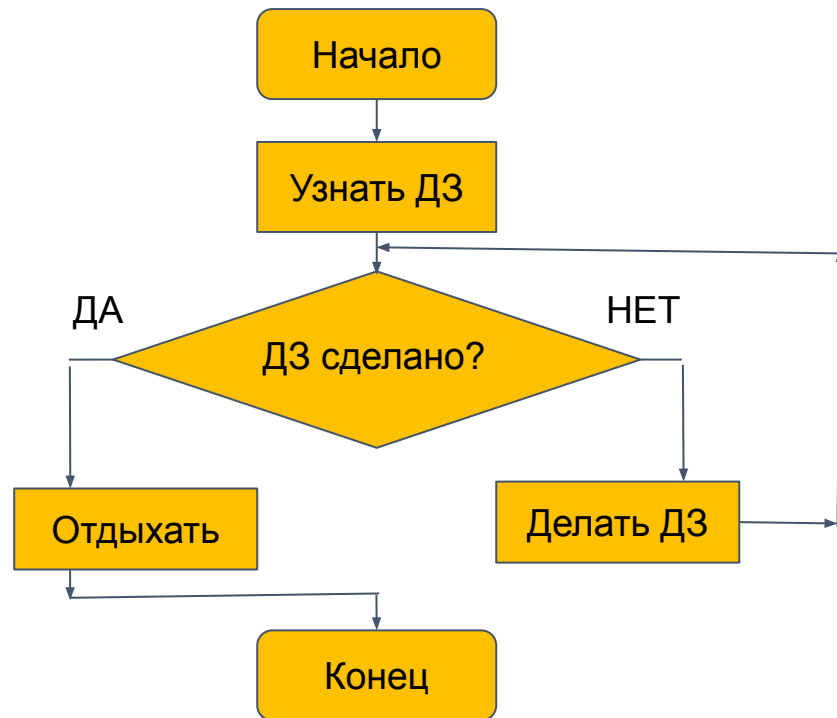
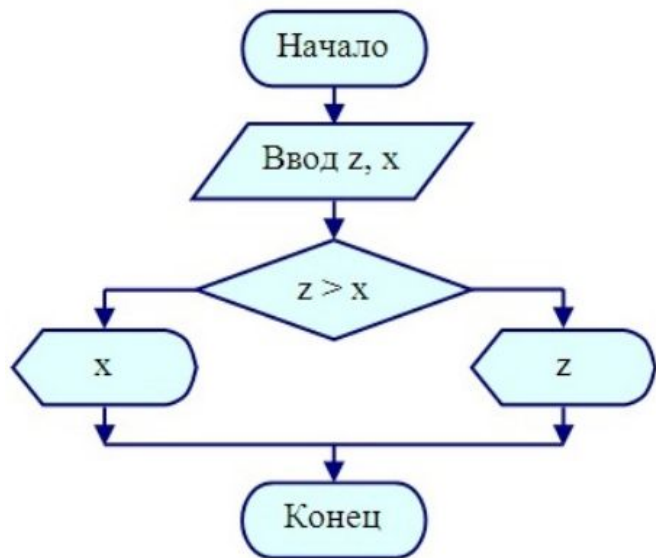
- Ввести  $z, x$
- Если  $z > x$ , то выводим  $z$
- Если  $x > z$ , то выводим  $x$

## Алгоритм выполнения домашнего задания:

- Узнать домашнее задание
- Выполнить домашнее задание
- Если домашнее задание выполнено, то отдыхай
- Если домашнее задание не выполнено, то выполняй домашнее задание



# Графический алгоритм



# Псевдокод, специальный язык

## Алгоритм сравнения переменных:

START

Number input: Z, X

IF  $Z > X$  THEN output Z

ELSE output X

END

## Алгоритм выполнения домашнего задания:

START

READ determine the task

WHILE (task is done)

doing task

END



4

# ВОПРОСЫ ПО ОСНОВНОМУ БЛОКУ



5

# ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

## Задание для закрепления:

- Придумайте алгоритм не более, чем из 5 действий
- Создайте его:
  - в текстовом виде
  - в виде блок-схемы, графический
  - на псевдокоде, специальном языке
- Можно использовать любую удобную программу для реализации.



TEL-RAN  
by Starta Institute

# 6

# ПРАКТИЧЕСКАЯ РАБОТА

# Практическое задание 1

Рассмотрим приведенный ниже алгоритм линейного поиска:

Шаг 1: НАЧАТЬ

Шаг 2: Получить массив в arr и число для поиска в x

Шаг 3: Начните с крайнего левого элемента arr[] и один за другим сравните x с каждым элементом arr[]

Шаг 4: Если x соответствует элементу Print True.

Шаг 5: Если x не совпадает ни с одним из элементов, выведите False.

Шаг 6: КОНЕЦ



# Реализация задания 1

Реализация на псевдокоде

```
START
READ array arr[], key k
  FOR i = 0 to end array do
    IF arr[i] = k THEN
      return true
    return false;
  END
```

# Реализация задания 1

## Реализация на Java

```
public static void main(String[] args) {  
    int[] arr = {1, 2, 3, 4, 5, 6};  
    System.out.println("Number exists? - " + getNumber(arr, 6));  
}
```

1 usage

```
private static boolean getNumber(int[] arr, int number) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == number) {  
            return true;  
        }  
    }  
    return false;  
}
```

## Реализация на Java Script

```
function linearSearch() {  
    let arr = [1, 2, 3, 4, 5, 6];  
    console.log(`Number exists? = ${getNumber(arr, 6)}`);  
}
```

1 usage

```
function getNumber(arr, number) {  
    for (let i = 0; i < arr.length; i++) {  
        if (arr[i] === number) {  
            return true;  
        }  
    }  
    return false;  
}
```



TEL-RAN  
by Starta Institute

7

# ОСТАВШИЕСЯ ВОПРОСЫ

# Домашнее задание

1. Выполнить тест для лучшего усвоения пройденного материала.

Вы получите ссылку на Google Forms, по ссылке найдете тест.

2. Написать псевдокод для алгоритма: сложить три числа и вывести сумму.

Реализовать алгоритм в коде.

Шаг 1: Выполнение предварительных условий

Шаг 2: Разработка алгоритма

- Алгоритм сложения 3 чисел и вывода их суммы:
- Получить от пользователя 3 целочисленные переменные num1, num2 и num3.
- Возьмите три добавляемых числа в качестве входных данных для переменных num1, num2 и num3 соответственно.
- Объявите целочисленную переменную sum для хранения результирующей суммы трех чисел.
- Добавьте 3 числа и сохраните результат в переменной sum.
- Вывести значение переменной sum

Шаг 3: Проверка алгоритма путем его реализации.





# Полезные ссылки

- [Грокаем Алгоритмы. Иллюстрированное пособие для программистов и любопытствующих - 2017.PDF at master · mduisenov/GrokkingAlgorithms · GitHub](#)
- [Introduction to Algorithms - Wikipedia](#)
- [Pseudocode - Wikipedia](#)
- [Data structure - Wikipedia](#)

# ЗАКЛЮЧЕНИЕ

