ПМ4. Разработка десктопных приложений с использованием языков программирования.
РО 4.1 Написать код программы на выбранном языке программирования (C#).

1 семестр:

14 лекций.

10 лабораторных работ.

Практика – 4 недели.*

2 семестр:

15 лекций.

9 лабораторных работ.

Практика – 4 недели.*

Тема 1. Понятие алгоритма. Свойства алгоритмов. Принципы построения алгоритмов.

Цель занятия:

Разобраться в понятии алгоритма как фундаментального понятия в информатике, изучить его свойства, классификацию и методы представления.

Учебные вопросы:

1. Понятие алгоритма.
2. Свойства алгоритмов.
3. Основные виды алгоритмов.
4. Принципы построения алгоритмов.
5. Способы представления алгоритмов.

1. Понятие алгоритма.

Определение алгоритма*

Алгоритм – это точная последовательность действий, которые приводят к решению определенной задачи за конечное число шагов. Иными словами, это набор инструкций, которые указывают, что и в какой последовательности нужно делать для достижения желаемого результата.

Любой алгоритм составляется в расчете на конкретного исполнителя с учетом его возможностей.

Исполнитель — субъект, способный исполнять некоторый набор команд. Совокупность команд, которые исполнитель может понять и выполнить, называется системой команд исполнителя.

Роль алгоритмов в информатике

В информатике алгоритмы являются основополагающим понятием.

Компьютер – это машина, которая выполняет строго определенные инструкции. Эти инструкции и составляют алгоритм.

Благодаря алгоритмам компьютеры могут решать самые разнообразные задачи: от простых арифметических вычислений до сложных задач искусственного интеллекта.

Основные роли алгоритмов в информатике:

- **Программирование:** Любая программа – это реализация одного или нескольких алгоритмов на определенном языке программирования.
- **Анализ данных:** Алгоритмы используются для обработки больших объемов данных, поиска закономерностей, классификации и прогнозирования.
- **Искусственный интеллект:** Алгоритмы машинного обучения позволяют компьютерам обучаться на данных и принимать решения без явного программирования.
- **Криптография:** Алгоритмы шифрования и дешифрования обеспечивают безопасность данных.
- **Операционные системы:** Алгоритмы управления памятью, процессором и другими ресурсами компьютера лежат в основе работы операционных систем.

Алгоритм как основа программы:

- Программа начинается с создания алгоритма, который описывает, что нужно сделать.
- Алгоритм – это план, который программист затем переводит на язык программирования.

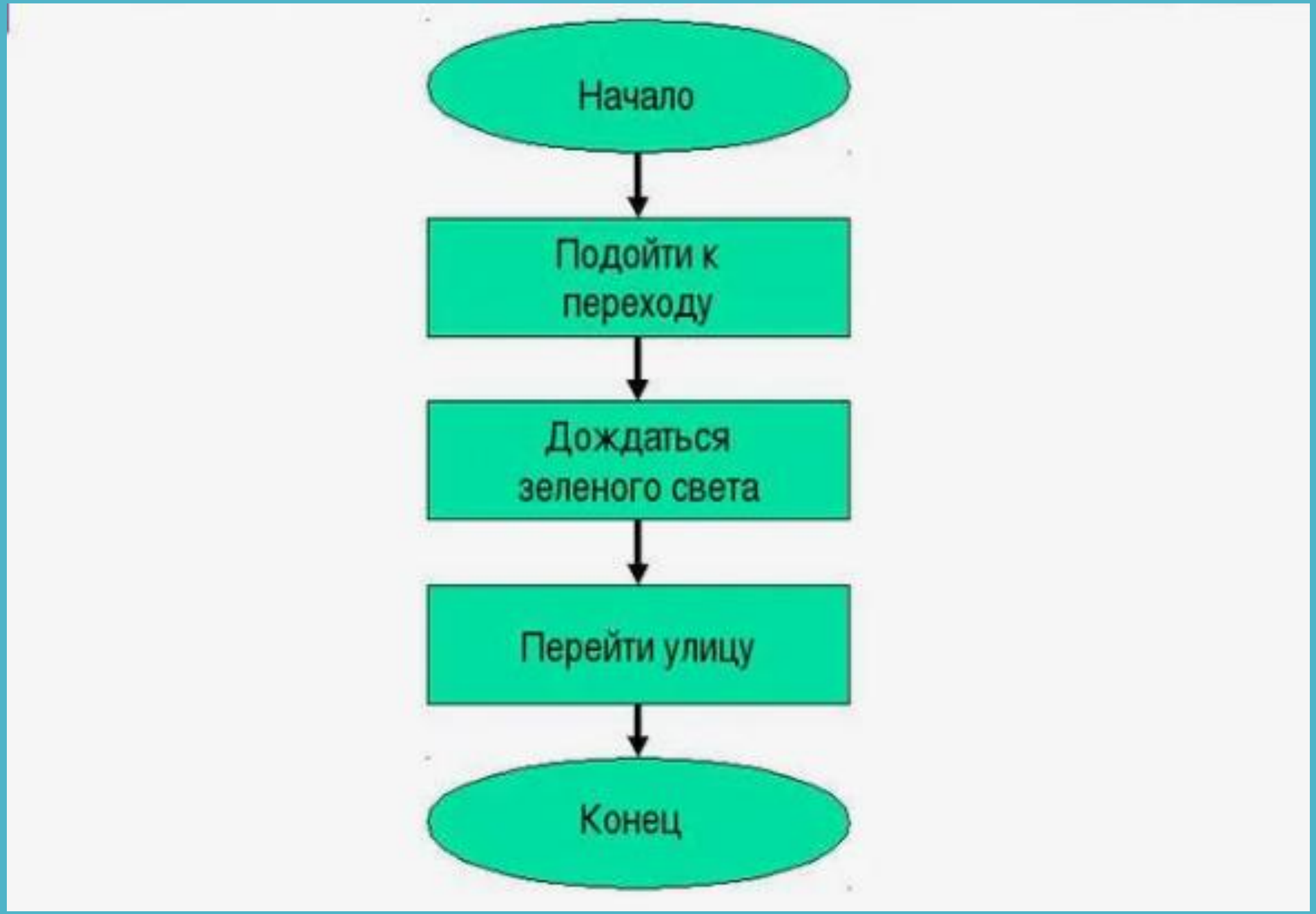
Примеры алгоритмов: от повседневности до программирования

Алгоритмы окружают нас повсюду, даже если мы об этом не задумываемся.

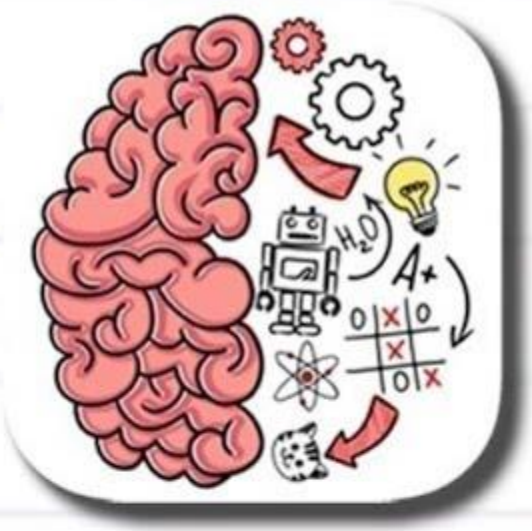
Каждый раз, когда мы следуем рецепту, собираем мебель или решаем математическую задачу, мы выполняем алгоритм.

Повседневные примеры:

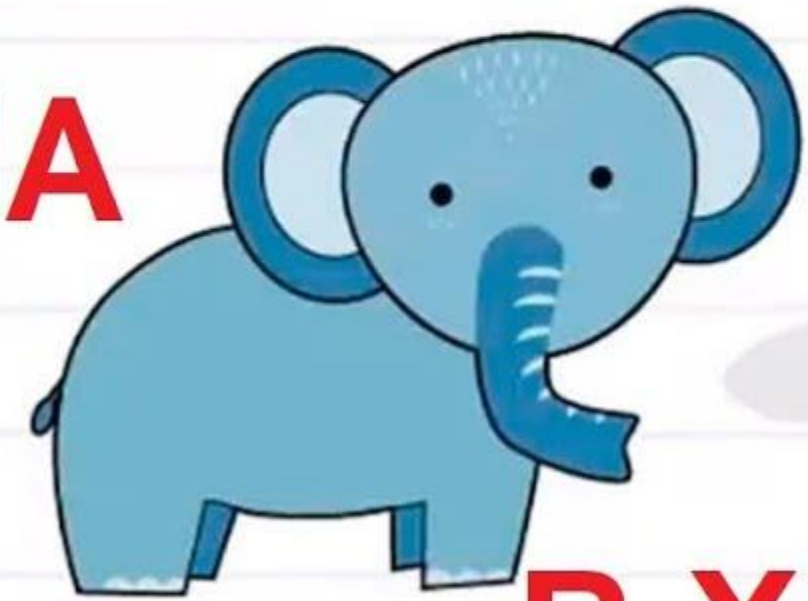
- **Рецепт:** Каждый рецепт - это подробный алгоритм приготовления блюда. Он описывает последовательность действий, ингредиенты и их количество.
- **Инструкция по сборке:** Инструкция по сборке мебели, например, шкафа или велосипеда, содержит пошаговое руководство, которое нужно следовать для получения конечного продукта.
- **Поиск пути:** Когда мы ищем кратчайший путь от дома до работы, мы неосознанно применяем алгоритм поиска пути. Навигационные приложения используют более сложные алгоритмы для прокладывания оптимальных маршрутов.


Начало
Подойти к переходу
Дождаться зеленого света
Перейти улицу
Конец


приготовление бутерброда
НАЧАЛО
отрезать ломтик хлеба
намазать масло на хлеб
отрезать ломтик сыра
положить сыр на хлеб
КОНЕЦ



КАК ЗАСУНУТЬ



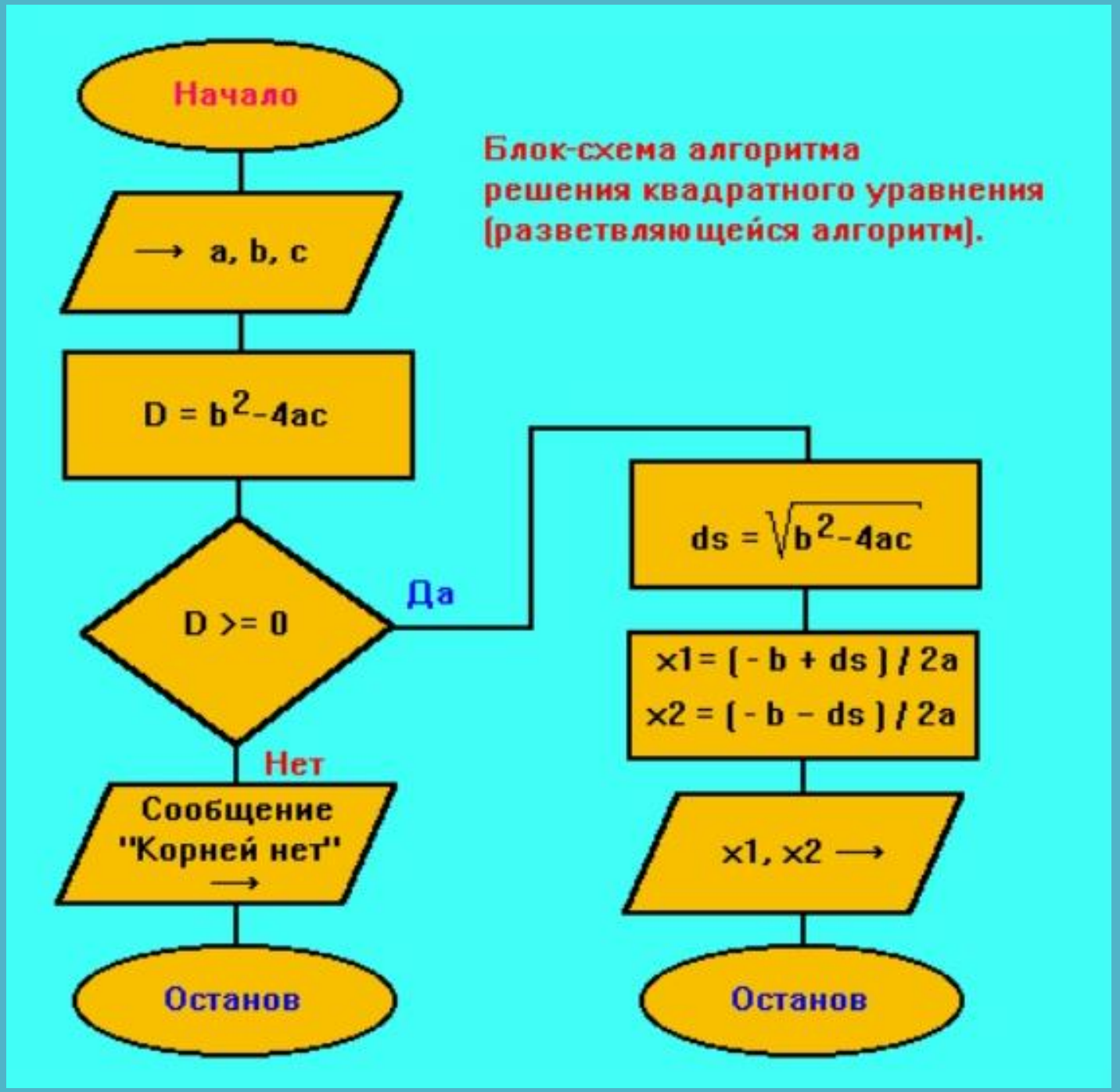
СЛОНА

В ХОЛОДИЛЬНИК

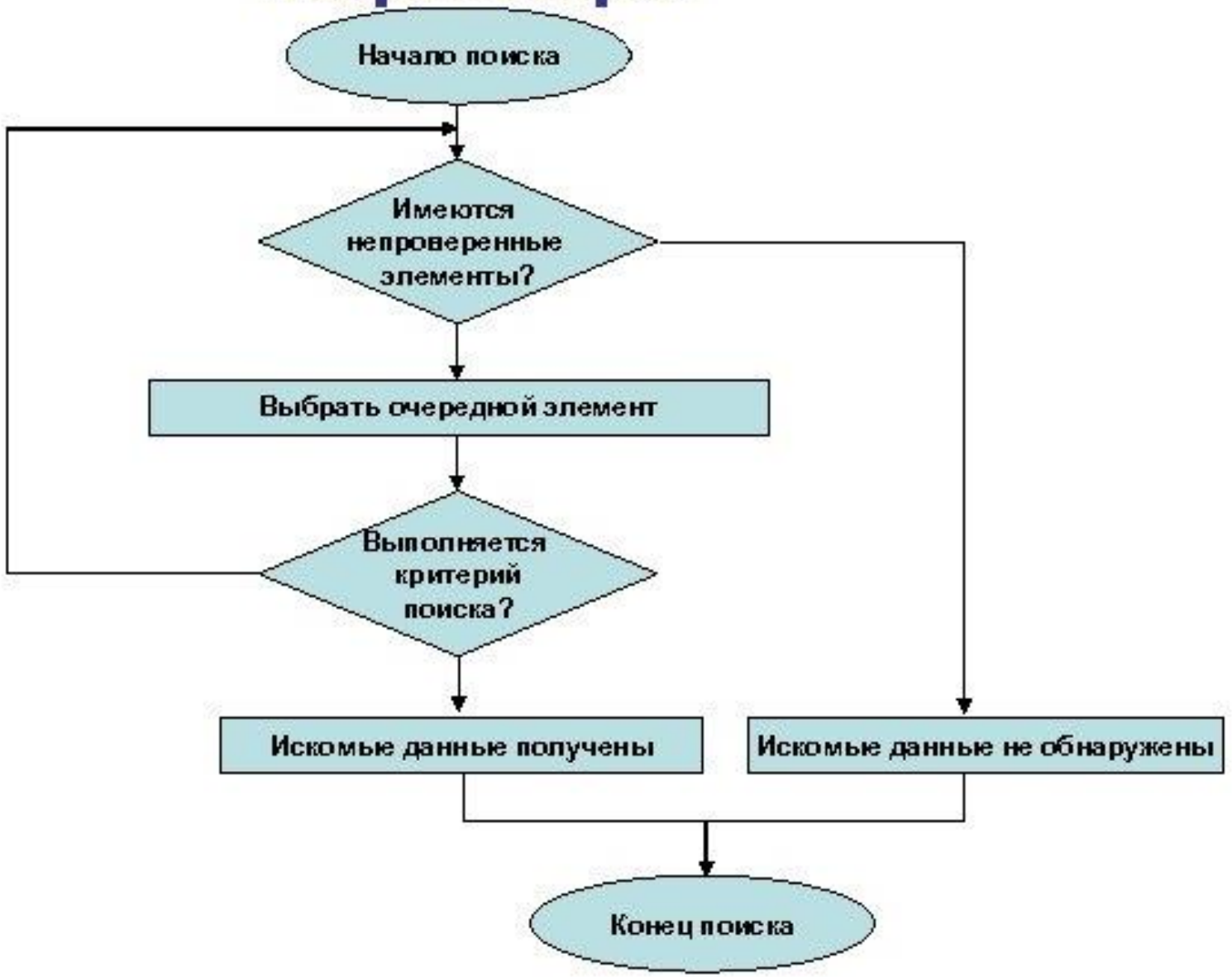
Разветвляющийся алгоритм.

1. Подойти к дороге.
2. Если горит зеленый свет - перейти дорогу.
3. Если не горит зеленый свет – не переходить дорогу.


Начало
Включи воду
Возьми тряпку
Возьми тарелку
Помой тарелку
Поставь тарелку в шкаф
Есть еще тарелки?
да
нет
Закрой воду
Конец


Блок-схема алгоритма
решения квадратного уравнения
(разветвляющейся алгоритм).
Начало
⟶ a, b, c
D = b2-4ac
D >= 0
Да
Нет
Сообщение
"Корней нет"
⟶
Останов
ds = √b2-4ac
x1= (- b + ds) / 2a
x2 = (- b – ds) / 2a
x1, x2 ⟶
Останов

Алгоритм последовательного перебора



Для выполнения алгоритма исполнителю недостаточно только самого алгоритма.

Выполнить алгоритм — значит применить его к решению конкретной задачи, т. е. выполнить запланированные действия по отношению к определенным **входным данным**.

Поэтому исполнителю необходимо иметь исходные (входные) данные — те, что задаются до начала алгоритма.

2. Свойства алгоритмов.

Алгоритм должен обладать определенными свойствами.

Наиболее важные свойства алгоритмов: **Дискретность, результативность, понятность, определенность, массовость.**

1. Дискретность. Алгоритм состоит из отдельных, четко определенных шагов.

Каждый шаг алгоритма должен быть простым и понятным, не допускающим неоднозначной интерпретации. Это позволяет разбить сложную задачу на более мелкие, легко выполнимые подзадачи.

2. Результативность. Алгоритм должен приводить к решению задачи за конечное число шагов, при этом должен быть получен результат.

Алгоритм не должен бесконечно выполняться. Он должен гарантированно завершиться, достигнув желаемого результата.

3. Понятность. Алгоритм должен быть понятен исполнителю.

Алгоритм должен быть записан на языке, понятном исполнителю (человеку или компьютеру). Каждый шаг должен быть ясен и не требовать дополнительных разъяснений.

4. Определенность. Каждый шаг алгоритма должен быть точно и однозначно описан.

Не должно быть никаких неопределенностей или двусмысленностей в том, какое действие следует выполнить на каждом этапе. Это гарантирует, что разные исполнители, следуя одному и тому же алгоритму, получат одинаковый результат.

5. Массовость. Алгоритм должен быть применим к широкому классу входных данных.

Один и тот же алгоритм можно использовать для решения задач с различными исходными данными. Например, алгоритм сортировки может быть применен как к массиву чисел, так и к массиву строк.


Свойства алгоритма
Дискретность
Путь решения задачи разделён на отдельные шаги
Понятность
Алгоритм состоит из команд, входящих в СКИ
Определённость
Команды понимаются однозначно
Результативность
Обеспечивается получение ожидаемого результата
Массовость
Обеспечивается решение задач с различными исходными данными

Пример: Простой алгоритм приготовления чая:

1.Вскипятить воду в чайнике.

2.Налить кипяток в пустую чашку.

3.Положить в чашку чайный пакетик.

4.Добавить сахар по вкусу.

Этот алгоритм обладает всеми указанными свойствами:

•**Дискретность:** Задача разбита на отдельные шаги.

•**Определенность:** Каждый шаг описан четко и однозначно.

•**Результативность:** Алгоритм завершается за конечное число шагов, результат – чашка чая.

•**Массовость:** Алгоритм можно применять для приготовления разных сортов чая.

•**Понятность:** Алгоритм понятен любому человеку, умеющему читать.

3. Основные виды алгоритмов.

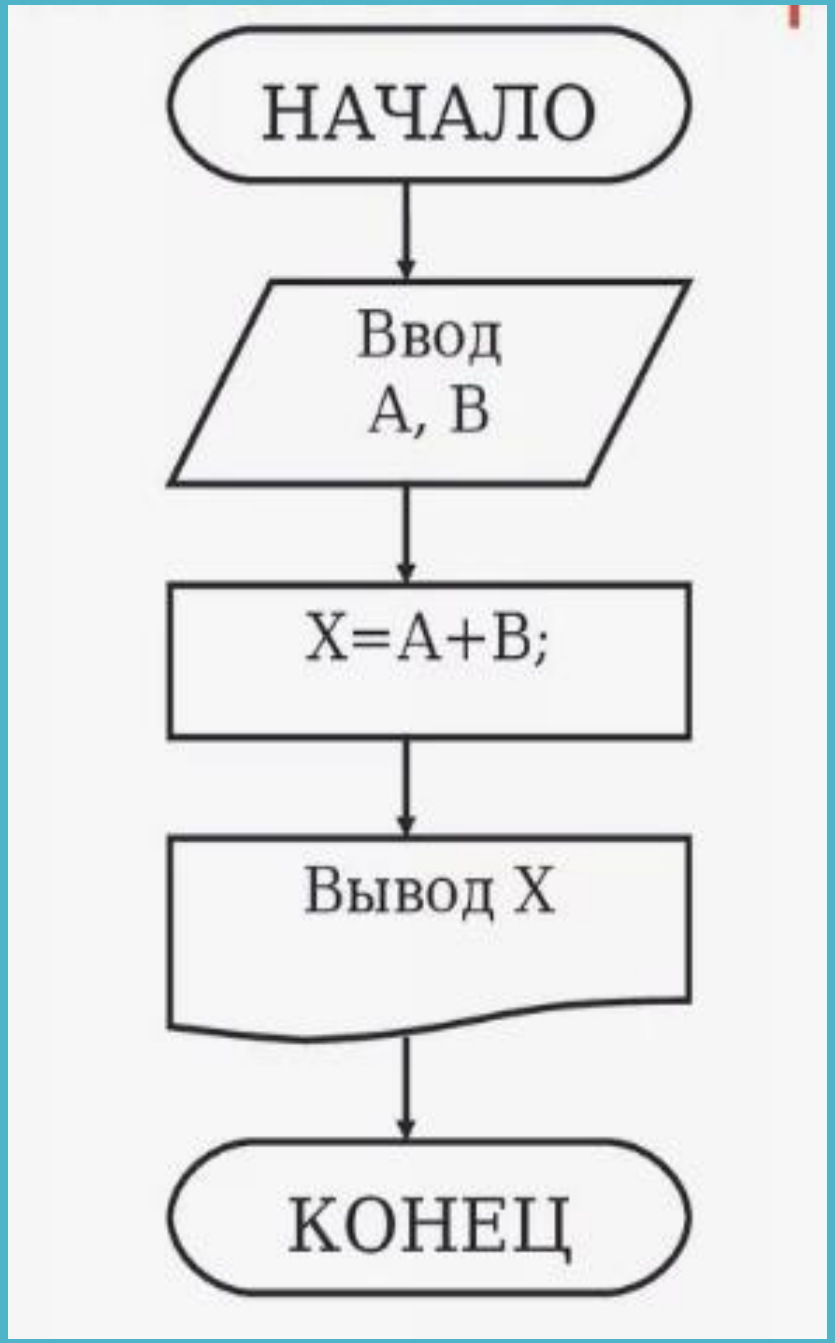
Алгоритмы можно классифицировать по различным признакам, но наиболее распространенное деление основано на структуре их выполнения.

Основные виды алгоритмов по структуре выполнения:

- Линейные алгоритмы
- Разветвляющиеся алгоритмы
- Циклические алгоритмы

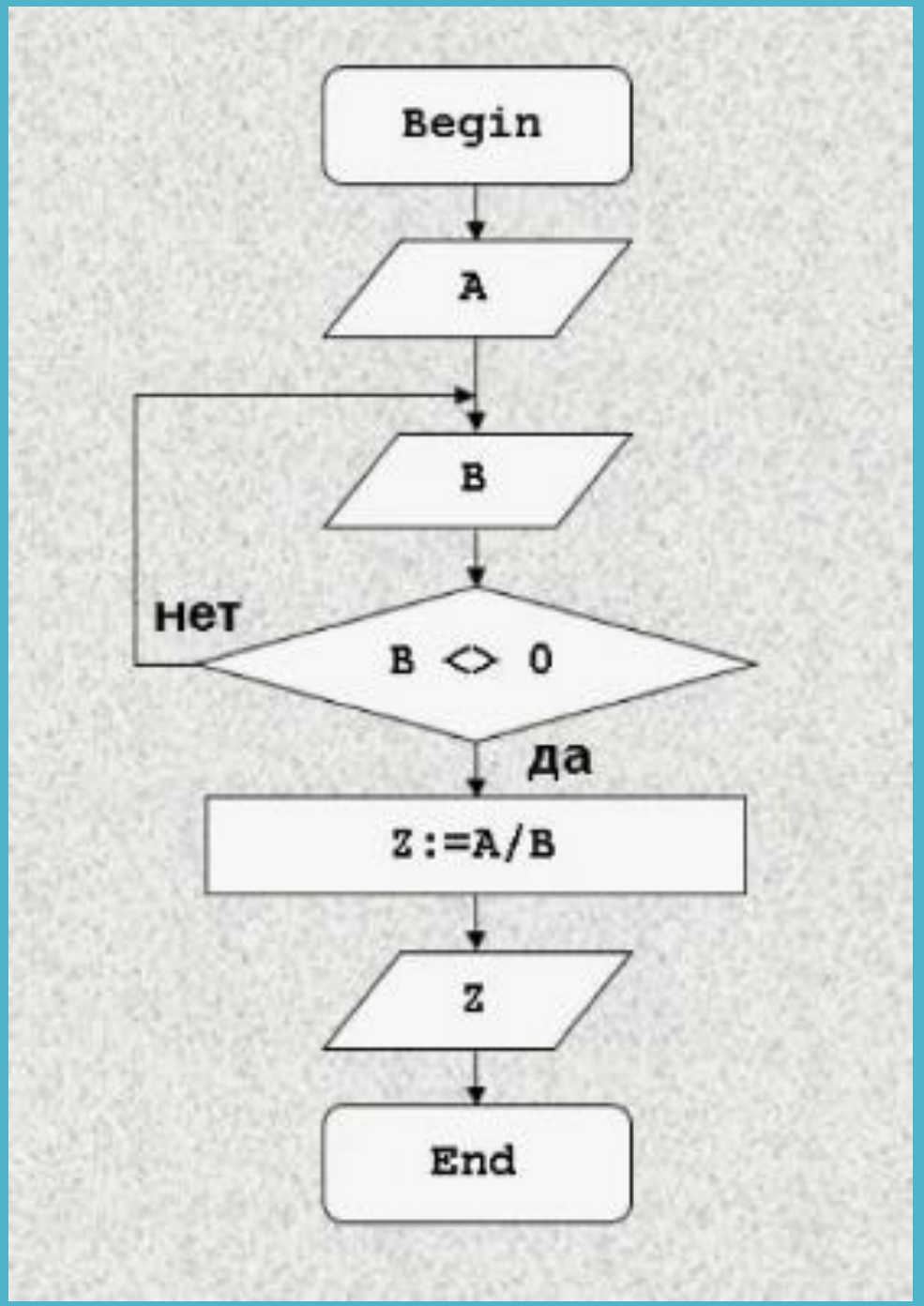
1. Линейные алгоритмы

- **Описание:** Действия в алгоритме выполняются последовательно, одно за другим.
- **Пример:** Рецепт приготовления чая, инструкция по сборке мебели.
- **Блок-схема:** Линейная последовательность блоков.



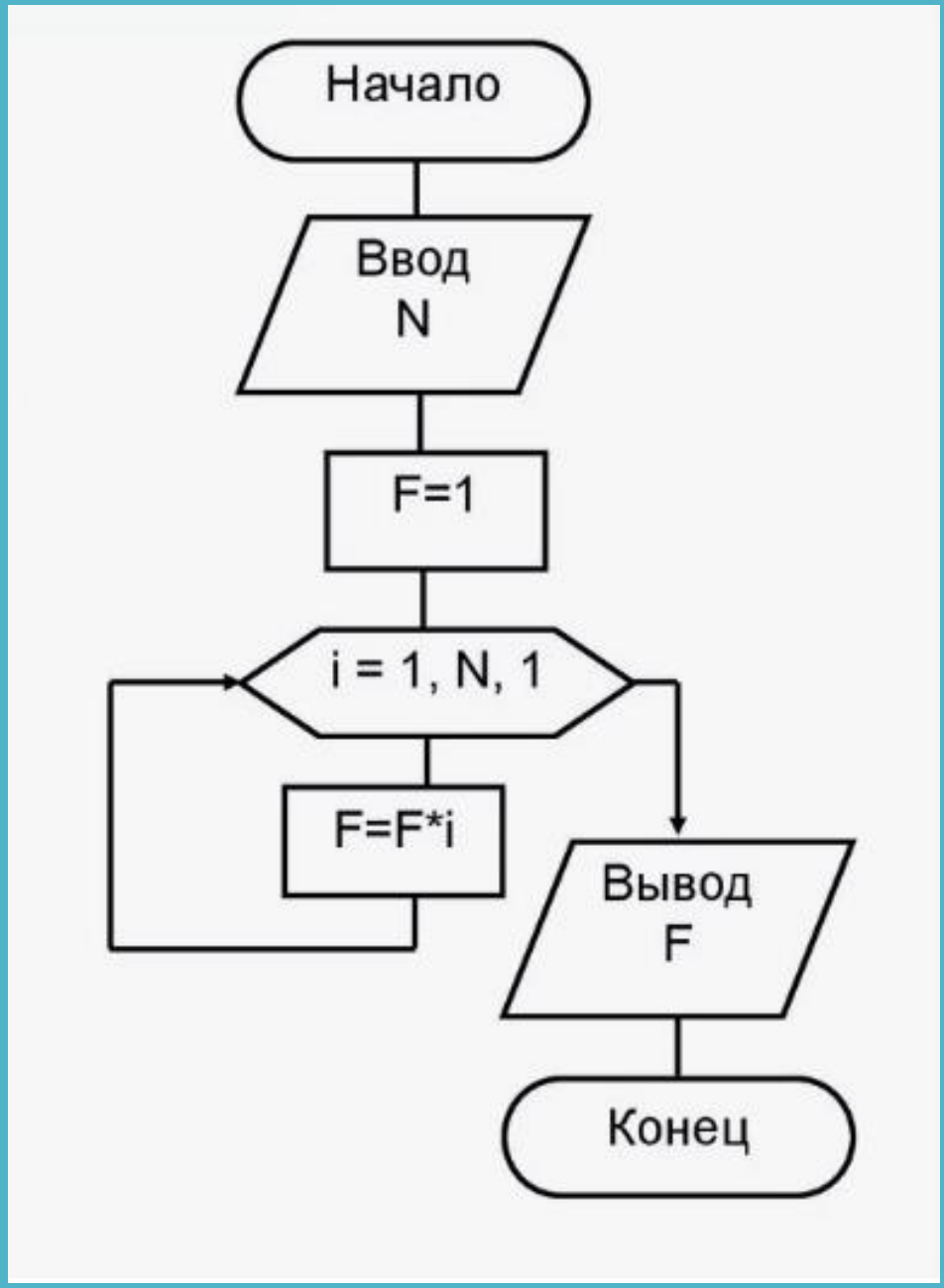
2. Разветвляющиеся алгоритмы

- **Описание:** В алгоритме присутствует условие, от выполнения которого зависит дальнейший ход выполнения.
- **Пример:** Проверка на четность числа, сортировка выбором.
- **Блок-схема:** Ветвление, как правило, изображается в виде ромба с двумя выходами.



3. Циклические алгоритмы

- **Описание:** В алгоритме присутствует блок, который выполняется многократно до тех пор, пока не выполнится определенное условие.
- **Пример:** Вычисление факториала числа, поиск максимального элемента в массиве.
- **Блок-схема:** Цикл изображается, как правило, в виде шестиугольника.



4. Принципы построения алгоритмов.

Декомпозиция: Разложение сложной задачи на более простые. Декомпозиция является первым шагом к созданию модульного кода.

Модульность: Разбиение алгоритма на более мелкие подзадачи. Это результат процесса декомпозиции, когда подзадачи оформлены в виде независимых модулей, которые могут быть повторно использованы и легко поддерживаются.

Рекурсия: Определение алгоритма через сам себя.

Абстракция: Выделение основных характеристик объекта, игнорирование несущественных деталей. Абстракция позволяет сосредоточиться на том, что должна делать программа, не отвлекаясь на то, как она это делает. (принцип "черного ящика")

5. Способы представления алгоритмов.

Существует несколько способов представления алгоритмов, каждый из которых имеет свои преимущества и области применения. Выбор конкретного способа зависит от сложности алгоритма, уровня детализации, для кого предназначено описание (программист, пользователь) и т.д.

Основные способы представления алгоритмов:

- Словесный способ.
- Графический способ.
- Псевдокод.
- Программный код

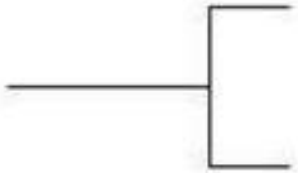
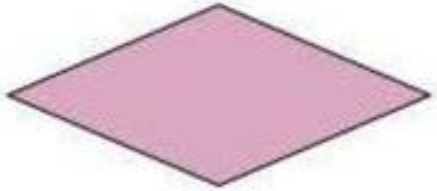
Словесный способ:

- **Описание:** Описание алгоритма на естественном языке (русском, английском и т.д.).
- **Преимущества:** Понятен для широкого круга людей, не требующий специальных знаний.
- **Недостатки:** Может быть неоднозначным, громоздким для сложных алгоритмов.
- **Пример:** "Чтобы найти среднее арифметическое двух чисел, нужно сложить эти числа и результат разделить на два."

Графический способ:

- **Описание:** Использование графических символов для отображения операций и потока выполнения алгоритма.
- **Преимущества:** Нагляден, легко воспринимается визуально, особенно для сложных алгоритмов.
- **Недостатки:** Может быть громоздким для простых алгоритмов.
- **Пример:** Блок-схема алгоритма.

Элементы блок-схем строятся из следующих элементов:

	Начало или конец алгоритма		Обращение к вспомогательному алгоритму
	Ввод или вывод информации		Начало цикла
	Простая команда, вычисление		Комментарий
	Проверка условия		Порядок выполнения действий

Псевдокод:

- **Описание:** Использование формализованного языка, близкого к языкам программирования, но без строгой синтаксической привязки.
- **Преимущества:** Более формализован, чем словесный способ, но проще для понимания, чем код на конкретном языке программирования.
- **Недостатки:** Требует знания базовых конструкций программирования.
- **Пример:**

```
НайтиМаксимальноеЧисло(массив)
    максЧисло = массив[0]
    Для каждого элемента в массиве
        Если элемент > максЧисло
            максЧисло = элемент
    Вернуть максЧисло
```

Программный код:

- **Описание:** Запись алгоритма на конкретном языке программирования (Python, C++, Java и т.д.).
- **Преимущества:** Точное и однозначное представление алгоритма, готовое к выполнению на компьютере.
- **Недостатки:** Требует знаний языка программирования.
- **Пример:**

```
def find_max(numbers):
    max_num = numbers[0]
    for num in numbers:
        if num > max_num:
            max_num = num
    return max_num
```

Контрольные вопросы:

- Что такое алгоритм? Дайте определение и приведите примеры.
- Перечислите и объясните свойства алгоритмов.
- Каковы основные виды алгоритмов? Опишите их особенности.
- Какие принципы используются при построении алгоритмов?
- Какие способы представления алгоритмов существуют? В чем преимущества и недостатки каждого из них?
- Приведите примеры алгоритмов, которые используются в повседневной жизни.
- Почему важно учитывать систему команд исполнителя при составлении алгоритма?
- Какова роль алгоритмов в программировании и других областях информатики?

Домашнее задание:

1. Опишите алгоритм (в рабочей тетради, графическим способом), выполнения повседневного действия или др. процесса, например:

- алгоритм похода в магазин за покупками
- алгоритм поиска потерянной вещи
- алгоритм приготовления бутерброда
- алгоритм уборки комнаты
- др.

Список литературы:

1. В.В. Трофимов Основы алгоритмизации и программирования. Учебник для СПО. 2019г.
2. Жаксыбаева Н.Н. Основы объектно-ориентированного программирования: язык C#. Часть 1./ Учебное пособие предназначено для учащихся технического и профессионального образования, Алматы, 2010,

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com