education

Добро пожаловать на курс «Алгоритмы и структуры данных»

Преподаватели курса
Илья Почуев | Михаил Павлов

Что будет на занятии

- Что такое алгоритмы и структуры данных
- Зачем программисту знать алгоритмы и структуры данных
- О курсе
- Что будет в курсе и чего не будет

education

Что такое алгоритмы и структуры данных

Понятие алгоритма

Аль-Хорезми, «Краткая книга о восполнении и противопоставлении» (820 г.)

- Основы арифметических вычислений в десятичной системе счисления
- Правила решений линейных и квадратных уравнений
- Правила измерения площадей и объёмов

- Решение задач на ЭВМ
- Определение и свойства алгоритма
- Способы представления алгоритмов
- Машина Тьюринга
- Показатели эффективности алгоритма

Понятие алгоритма

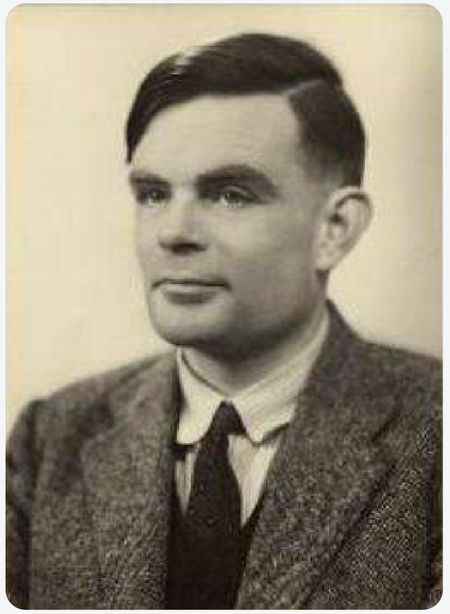
Алгоритм – конечная система формальных правил (набор инструкций), точно и однозначно описывающая процесс решения поставленной задачи исполнителем (получение выходного результата из входных данных) в виде конечной последовательности действий (операций)

Программа – реализация алгоритма на формальном языке исполнителя или языке программирования для последующей трансляции

Абстрактные исполнители (для формального исследования алгоритмической разрешимости и вычислительной сложности):

- машина Тьюринга
- машина Поста
- нормальный алгоритм Маркова
- и пр.

Эти абстрактные исполнители, как и современные ЭВМ, могут быть имитированы на машине Тьюринга – являются полными по Тьюрингу



Алан Тьюринг

(1912–1954)

Способы представления алгоритма

- Естественное или полуформальное представление
  - Схематическое
    1. Блок-схема
    2. Диаграмма (граф) состояний и переходов
  - Символьное
    1. Словесное
    2. Псевдокод
    3. Формула
- Формальное представление
  - Программное
    1. Машинный код
    2. Язык программирования
  - Математическое
    1. Машина Тьюринга
    2. Машина Поста
    3. Нормальный алгоритм Маркова
    4. Рекурсивные функции

Примеры полуформальных представлений алгоритма

Формула

$$n! = \begin{cases} \prod_{i=1}^{n} i, & \text{если } n > 0; \\ 1, & \text{если } n = 0; \\ \text{не определено}, & \text{если } n < 0 \end{cases}$$

Словесное описание алгоритма

1. Ввести целое n.
2. Если n < 0, то вывести 0 (признак ошибки), КОНЕЦ.
3. Присвоить x = 1, i = 1.
4. Если i ≥ n, то вывести x, КОНЕЦ.
5. Присвоить x = x · i, i = i + 1.
6. Перейти на шаг 4.

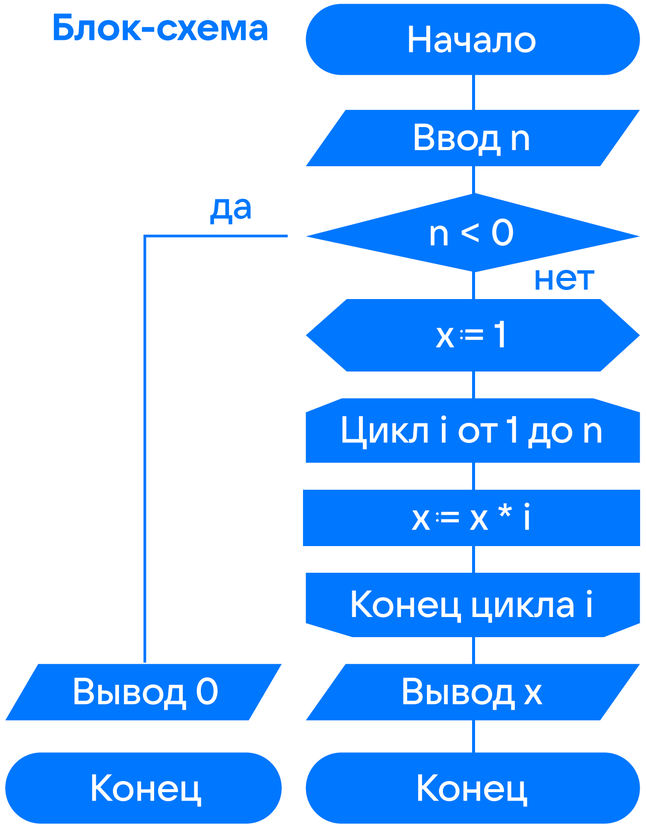
```
//представление в псевдокоде
function factorial(n) {
 if (n < 0) {
   return 'Факториал не опредёлен для отрицательных чисел';
 }

 // Инициализируем результат как 1 (поскольку факториал 0 равен 1)
 result = 1;

 // Используем цикл для умножения чисел от 1 до n
 for (i = 1; i <= n; i++) {
  result *= i;
 }

 return result;
}
```

Блок-схема



Схематическое представление: блок-схема

Блок-схема – графическое представление алгоритма, в котором шаги изображаются блоками различной формы и соединяются линиями в порядке выполнения

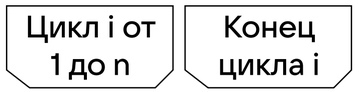
Точки начала и концов алгоритма

Подготовительный процесс (инициализация данных и пр.)

Действие или набор действий

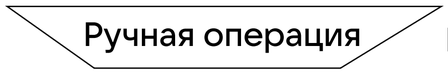
Подпрограмма, возможно, с отдельной блок-схемой

Точка ветвления по условию (один вход, несколько выходов)



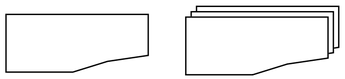
Начало и конец цикла

Абстрактный ввод/вывод информации



Процесс, требующий действия человека

Ручной ввод данных (символизирует клавиатуру)

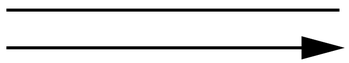


Документ, документы (например, выходной отчёт)

Сохранение данных, диск, файл, база данных

Отображение информации, вывод на экран

Соединители разрыва на одной и на разных страницах



Поток управления (данных) (по умолчанию: вправо, вверх)

Комментарий или пояснение

Примеры формальных представлений алгоритма

Программа на языке ассемблера x86-64

```
        movq    $0, -16(%rbp)
        leaq    -24(%rbp), %rax
        movq    %rax, %rsi
        movl    $_ZSt3cin, %edi
        call    _ZNSirsERl
        movq    -24(%rbp), %rax
        testq   %rax, %rax
        js      .L2
        movq    $1, -8(%rbp)
        movq    $1, -16(%rbp)
        jmp     .L3
.L4:
        movq    -16(%rbp), %rax
        imulq   -8(%rbp), %rax
        movq    %rax, -16(%rbp)
        addq    $1, -8(%rbp)
.L3:
        movq    -24(%rbp), %rax
        cmpq    %rax, -8(%rbp)
        jle     .L4
.L2:
        movq    -16(%rbp), %rax
        movq    %rax, %rsi
        movl    $_ZSt4cout, %edi
        call    _ZNSolsEl
```

Машинный код x86-64

```
48 C7 45 F0 00 00 00 00
48 8D 45 E8
48 89 C6
BF 00 00 00 00
E8 00 00 00 00
48 8B 45 E8
48 85 C0
78 2E
48 C7 45 F8 01 00 00 00
48 C7 45 F0 01 00 00 00
EB 12

48 8B 45 F0
48 0F AF 45 F8
48 89 45 F0
48 83 45 F8 01

48 8B 45 E8
48 39 45 F8
7E E4

48 8B 45 F0
48 89 C6
BF 00 00 00 00
E8 00 00 00 00
```

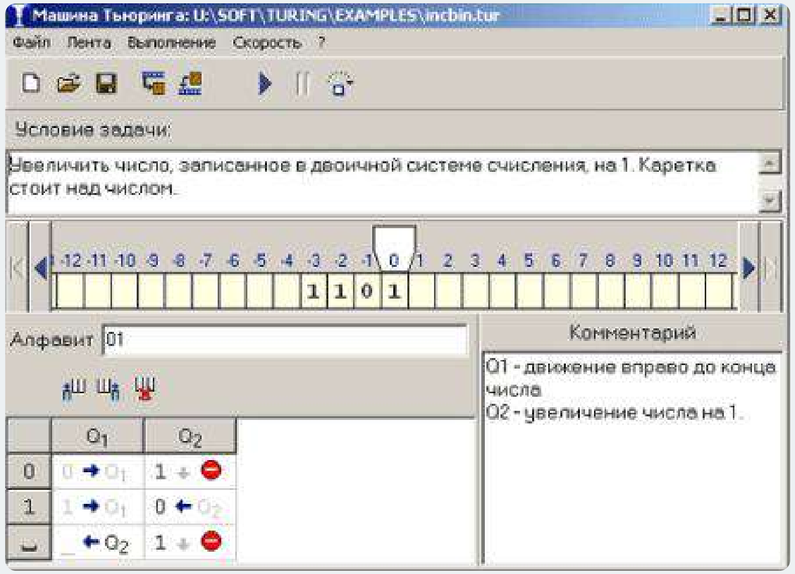
Программа на языке C++

```
// Вычисление факториала
long i, n, x = 0;
std::cin >> n;
if (n >= 0)
  for (i = 1, x = 1; i <= n; ++i)
    x *= i;
std::cout << x;
```

Машина Тьюринга

Тезис Тьюринга: для любой (в интуитивном понимании) алгоритмической вычислимой функции существует машина Тьюринга, её вычисляющая

Реализации машины Тьюринга с конечной лентой



Программный эмулятор

© 2010 Константин Поляков, Россия

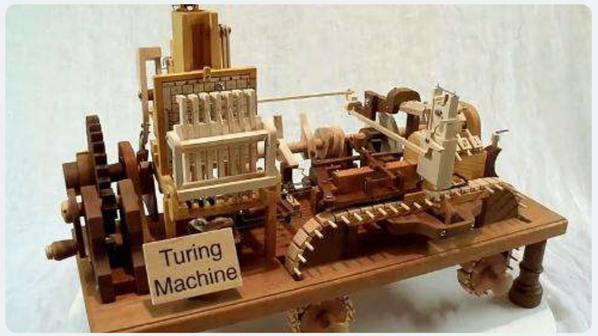
https://kpolyakov.spb.ru/prog/turing.htm

Электромеханическая модель

© 2010 Mike Davey, США

https://aturingmachine.com

https://www.computerhistory.org/collections/catalog/102682867



Механическая модель из дерева

© 2015 Richard J. Ridel

https://hackaday.com/2018/03/08/mechanical-wooden-turing-machine

Электромеханическая модель

© 2012 Jeroen van den Bosand & Davy Landman, Нидерланды

https://www.ecalpemos.nl/filmmaking/lego-turing-machine

education

Зачем программисту знать алгоритмы и структуры данных

Зачем нам это?

- Хранение данных
- Обработка данных
- Понимание сложности и ресурсоёмкости
- Часть профессиональной гигиены

education

Ещё один аспект

– алгоритмическая часть собеседований

education

О курсе

От массивов до динамического программирования, от списков до графов: широкий охват тем, от простого к сложному

- 5 модулей
- 30 занятий
- Более 50 разборов фундаментальных задач различной сложности
- Около 20 задач для самостоятельного решения с автопроверкой

Стандартный разбор темы на курсе

1. Теория
2. Разбор типичных задач на псевдокоде
3. Закрепление знаний: тестирование, самостоятельное решение задач с автопроверкой на платформе All Cups.

education

Что будет в курсе и чего не будет

Чего не будет в курсе

Наш курс призван дать в первую очередь
практическое понимание АиСД

- Языки программирования
- Математическое обоснование

Что будет в курсе

- Теория об основных структурах данных и алгоритмах
- Подходы к решению задач
- Будет много практики

Фишка курса

Практико-ориентированный подход: освойте алгоритмы и структуры данных на практике, решая реальные задачами, которые часто встречаются на собеседованиях ведущих IT-компаний

education

Успешной вам учёбы на курсе!

Ждём вас на следующих занятиях