

# Распределённые алгоритмы

[mk.cs.msu.ru](http://mk.cs.msu.ru) → Лекционные курсы → Распределённые алгоритмы

## Блок 6

Основные соглашения о псевдокоде

Лектор:

**Подымов Владислав Васильевич**

E-mail:

**[valdus@yandex.ru](mailto:valdus@yandex.ru)**

ВМК МГУ, 2025, февраль–май

В курсе будут часто требоваться описания (представления) р.а.,

- ▶ достаточно строгие, чтобы можно было рассуждать об их вычислениях в строгих терминах **систем переходов**,
- ▶ и при этом достаточно лаконичные и наглядные, чтобы было «содержательно» понятно, как выполняется алгоритм

Описание р.а., как правило, состоит из двух частей:

- ▶ Описание того, какого вида узлы могут содержаться в системах этого алгоритма
- ▶ Описание устройства узла каждого вида

Узел в «реальной» р.с., как правило, устроен просто: это

- ▶ либо последовательная программа с элементами недетерминизма,
- ▶ либо набор действий набора небольших детерминированных программ, параллельно недетерминированно выполняющихся при наступлении подходящих условий

Описывать узел будем при помощи **псевдокода** этого узла

Описание узла будем начинать с (локальных) **переменных** и их **начальных значений**

Будем использовать, в числе прочих, следующие множества переменных, считая их типами:

- ▶ булев тип  $\mathbb{B}$ :  $\{\text{t}, \text{f}\}$ , он же  $\{0, 1\}$
- ▶  $\mathbb{Z}$ : все целые числа
- ▶  $\mathbb{N}$ : все натуральные (целые положительные) числа
- ▶  $\mathbb{N}_0$ : все целые неотрицательные числа (натуральные с нулём)
- ▶  $\mathcal{T}^\perp$ : тип  $\mathcal{T}$  с дополнительным значением  $\perp$ , обозначающим, что значения нет
- ▶  $ARR[\mathcal{T}]$ : множество неограниченных массивов (бесконечных последовательностей) элементов типа  $\mathcal{T}$

Будем использовать следующие обозначения:

- ▶  $x/v$ : переменная  $x$  имеет значение  $v$
- ▶  $x : \mathcal{T}$ : переменная  $x$  имеет тип  $\mathcal{T}$
- ▶  $x : \mathcal{T}/v$ : переменная  $x$  имеет тип  $\mathcal{T}$  и начальное значение  $v$

Состоянием узла будем считать набор значений его переменных вместе с **состоянием управления**: номером текущей выполняемой команды

Среди команд будут выделяться простейшие, отвечающие действиям узла

В описании действия, отвечающего команде, будет говориться,

- ▶ как при выполнении перехода изменяются
  - ▶ значения переменных узла и его состояния управления и
  - ▶ состояние коммуникационной подсистемы
- ▶ в каких конфигурациях допустимо соответствующее действие (в действие не входят переходы из конфигураций, в которых оно недопустимо)

`nop;` — простейшая тривиальная команда:

- ▶ Управление передаётся следующей команде
  - ▶ (здесь и дальше — если следующей команды нет, то выполнение узла завершается, т.е. узел переходит в заключительное состояние)
- ▶ Больше ничего не происходит

`x := expr;` — простейшая команда присваивания:

- ▶ В `x` записывается текущее значение выражения `expr`
- ▶ Управление передаётся следующей команде

`send(m);` — простейшая команда отправки сообщения `m`:

- ▶ Переменные не изменяются
- ▶ Управление передаётся следующей команде
- ▶ В коммуникационную подсистему добавляется одно сообщение `m` для текущих значений записанных в `m` выражений

$\text{recieve}(v_1, \dots, v_n)$ ; — простейшая команда приёма сообщения:

- ▶ Если  $v_i$  в записи команды подчёркнуто, то это константа, иначе — переменная
- ▶ Действие допустимо  $\Leftrightarrow$  в коммуникационной подсистеме содержится сообщение, представляющее собой набор из  $n$  элементов с такими константами, как указано в команде, в соответствующих местах
- ▶ Из коммуникационной подсистемы удаляется одно сообщение, устроенное как описано выше, и в переменные, записанные в команде, присваиваются соответствующие значения сообщения
- ▶ Управление передаётся следующей команде

**Например:** « $\text{receive}(\underline{\mathbf{d}}, x)$ ;» означает, что следует

- ▶ принять сообщение, представляющее собой пару элементов, первый из которых равен  $\mathbf{d}$ , и
- ▶ присвоить второй элемент принятого сообщения в  $x$

**Псевдокод узла** — это последовательность простейших и составных команд

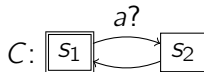
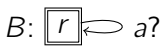
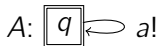
В ближайшее время понадобится только одна составная команда — цикл **do-until**:

**do** { *последовательность команд* } **until** *условие* ;

*Условие* — это либо «естественное» выражение со значением булевого типа, либо «неестественное», смысл которого будет поясняться при введении

Выполняется как *последовательность команд* без цикла, но если после выполнения *условие* не выполнено, то управление передаётся в начало *последовательности*

## Пример



Эти узлы можно представить как не содержащие ни одной переменной и имеющие такой псевдокод

A: **do** { *send*(*a*); } **until**  $\mathbb{f}$ ;

B: **do** { *receive*(*a*); } **until**  $\mathbb{f}$ ;

C: **do** { *receive*(*a*); **nop**; } **until**  $\mathbb{f}$ ;