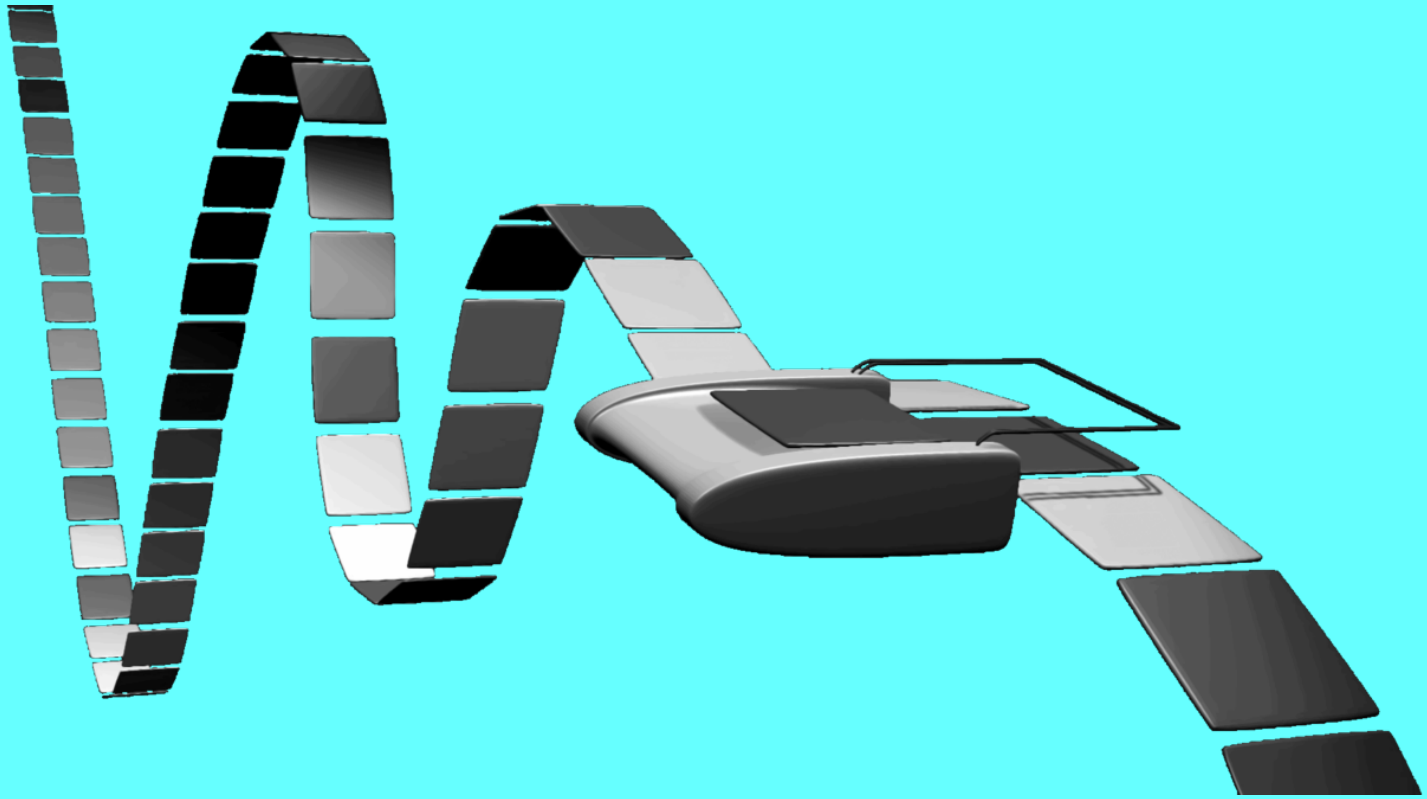
Неклассическая теория алгоритмов и алгоритмический язык ДРАКОН

Владимир Пароннджанов

Презентация доклада на Семинаре в Институте системного программирования РАН 19 мая 2017 года

Теория алгоритмов
Theory of computation



1. Теория формальных языков
2. Теория автоматов
3. Теория вычислимости
4. Теория сложности вычислений

The field is divided into three major branches: automata theory and language, computability theory, and computational complexity theory, which are linked by the question: "What are the fundamental capabilities and limitations of computers?“

Algorithm engineering

До Бога высоко, до царя далеко

- Алгоритмы
- Медицинские алгоритмы Тавровский
- Потоки работ (workflows)
- Бизнес-процессы, стартапы
- Алгоритмические предписания

Тезис Ланды-Непейводы

«Необходимо различать алгоритм и алгоритмическое предписание, имеющее внешнюю форму алгоритма, но включающее не до конца определенные шаги»

«Только через грустный опыт отстаивается золотой фонд медицины»

Кардиохирург

Николай Амосов

По рекомендации Института медицины в конгрессе США были проведены слушания и принят **Закон о безопасности пациентов,** подписанный президентом Джорджем Бушем младшим 29 июля 2005 года

Patient Safety and Quality Improvement Act of 2005

Владимир Пароджанов

Почему врачи убивают и калечат пациентов

или Зачем врачу блок-схемы алгоритмов?

Предисловие доктора медицинских наук,
профессора, члена-корреспондента РАН
Геннадия Порядина

Алгоритмы для эффективного клинического мышления

Пневмонэктомия
Чек-лист

Проблема

Классическая теория алгоритмов

1. **Не имеет удобного алгоритмического языка**
2. **Не приложима к медицинским алгоритмам и алгоритмическим предписаниям**
3. **Не пригодна для целостного описания человеческой деятельности**

ALGOL 60: The Death of a Programming Language and the Birth of a Science

Huub de Beer
Eindhoven School of Education
Eindhoven, February 12, 2010

http://datamuseum.dk/site_dk/20100213/Huub_de_Beer-ALGOL_anniversary.pdf

1955-1965: The Era of the Algorithmic Language

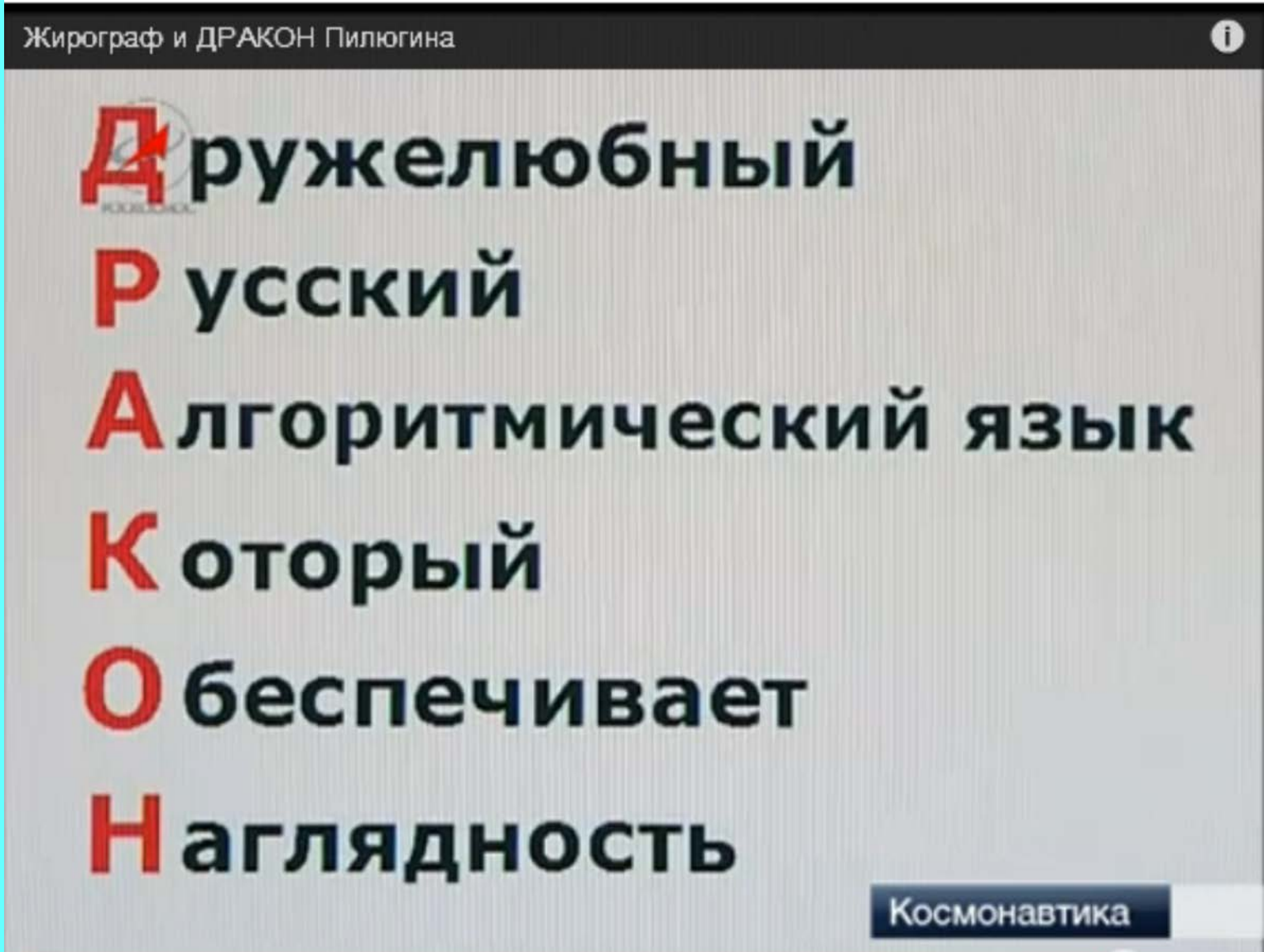
ALGOL became a hammer, and a bad one at that: a new ALGOL was needed.
Why Did ALGOL Die?

Solution: A General Purpose Programming Language

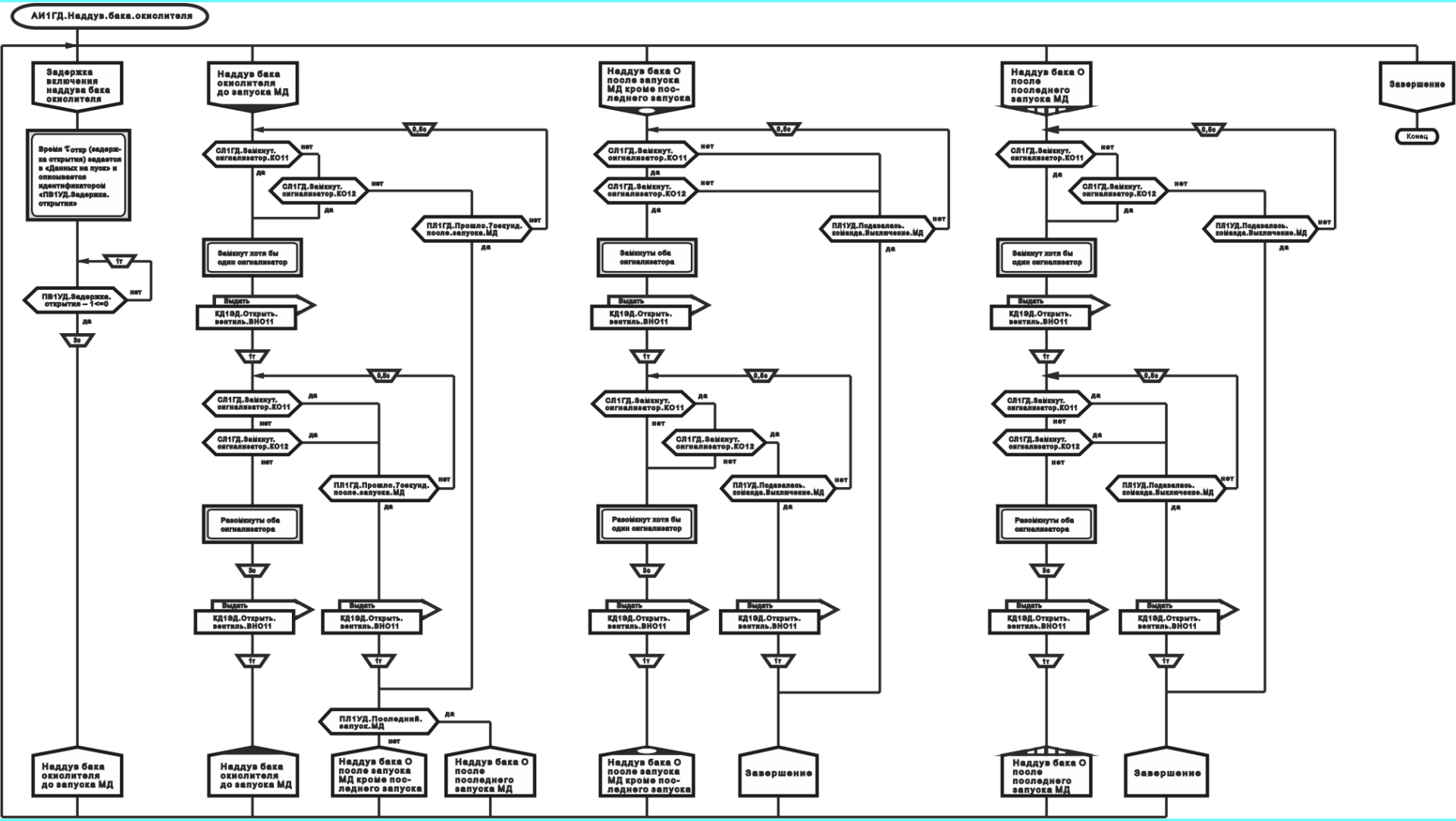
Кадр из фильма к 100-летию Пилюгина

ВЕСТИ

17 мая 2008 года
ЖИРОГРАФ И ДРАКОН ПИЛЮГИНА



Наддув бака окислителя



1

Алгоритмизация и программирование

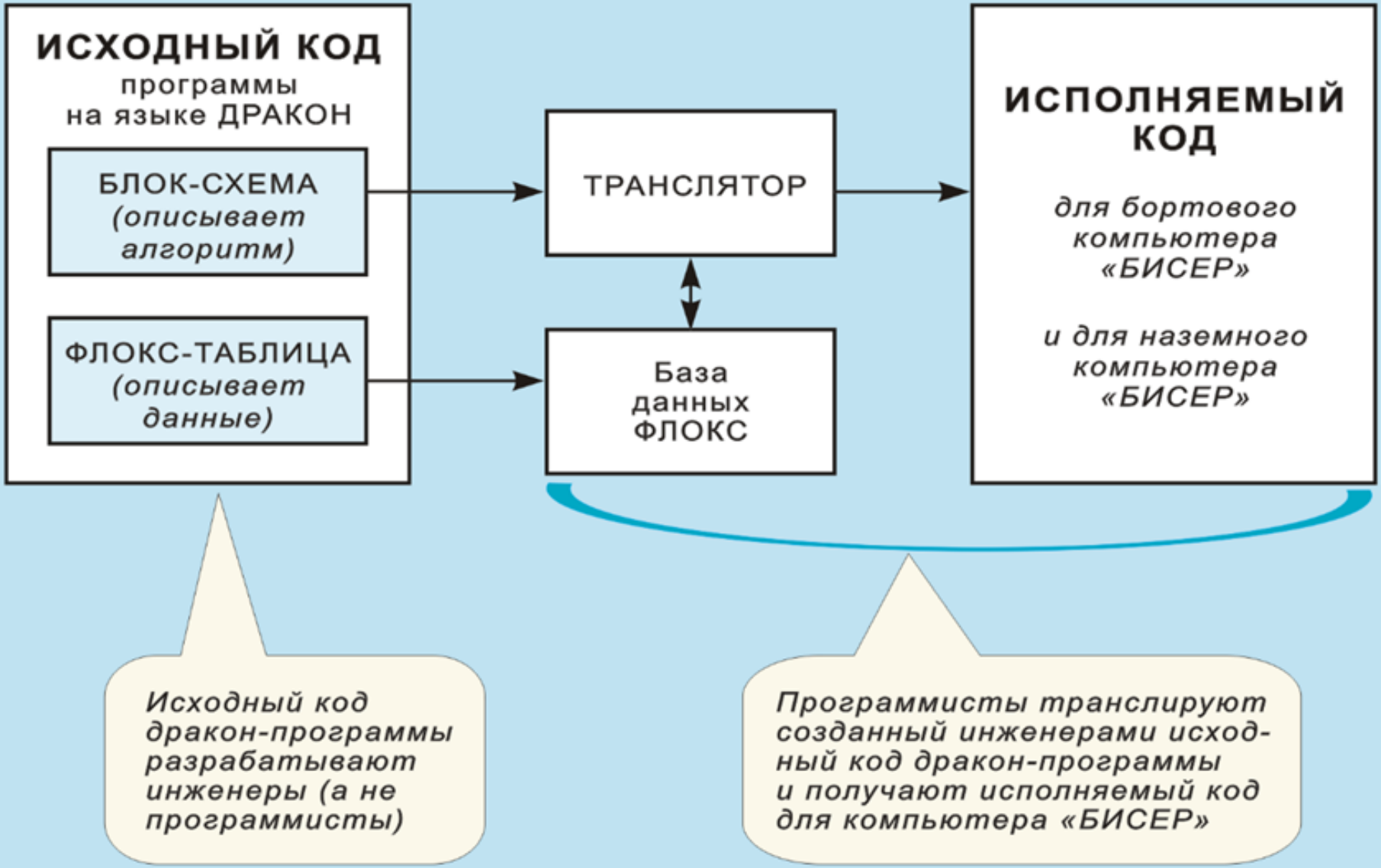
Джеймс Мартин (1933—2013)
автор идеи «Программирование без программистов»

ДРАКОН-технология в НПЦАП — это

Программирование без программистов

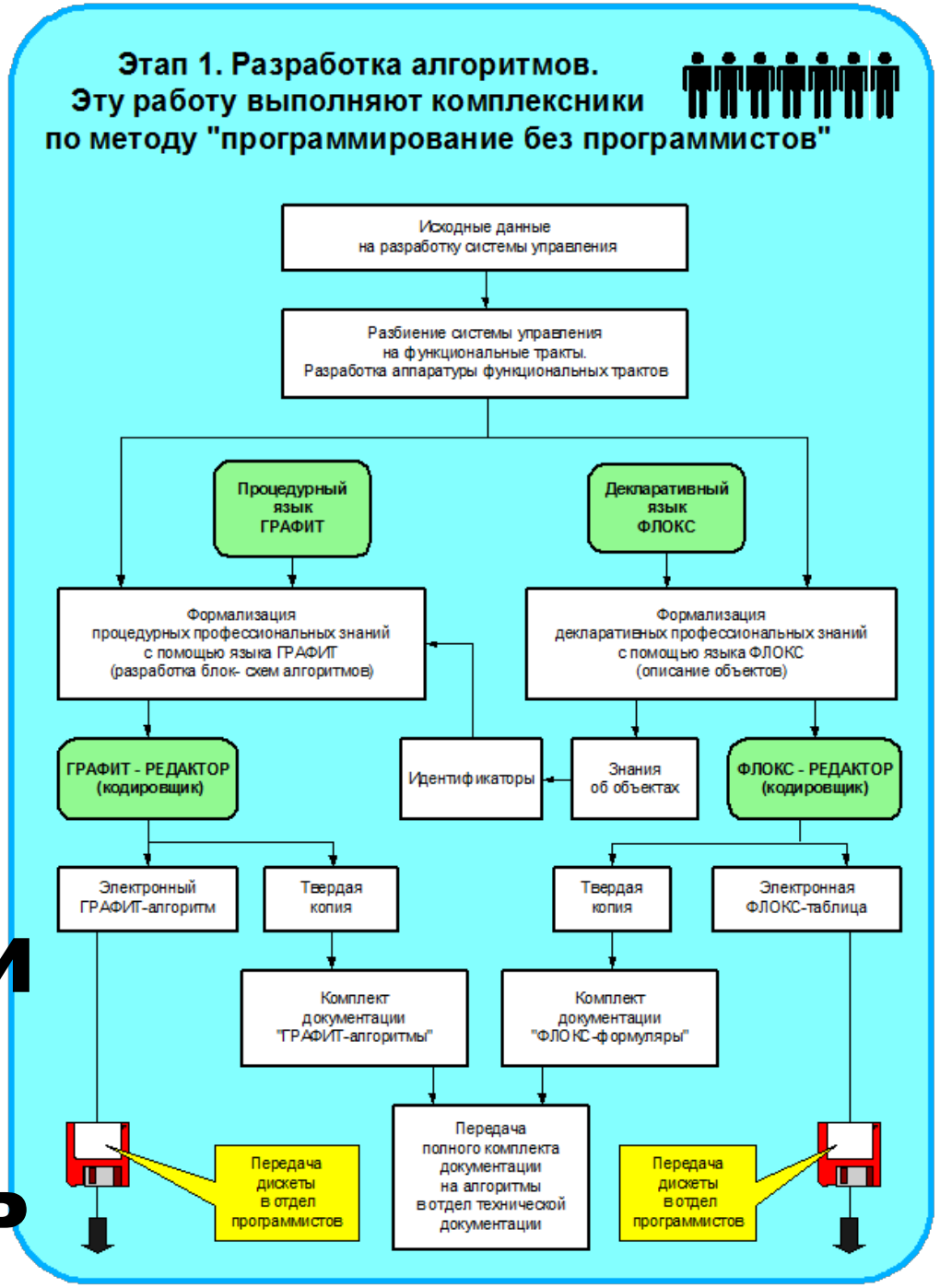
1996 — 2017

ДРАКОН-технология

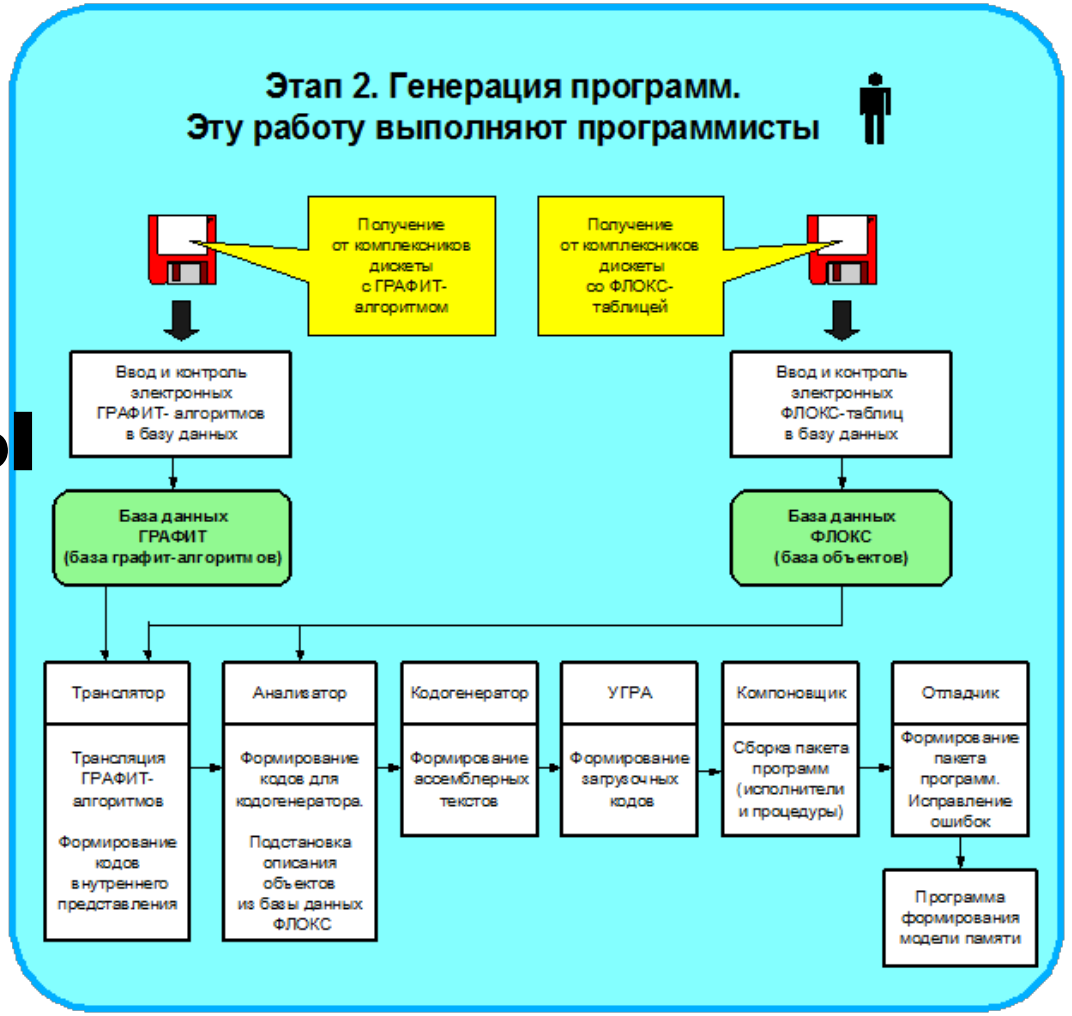


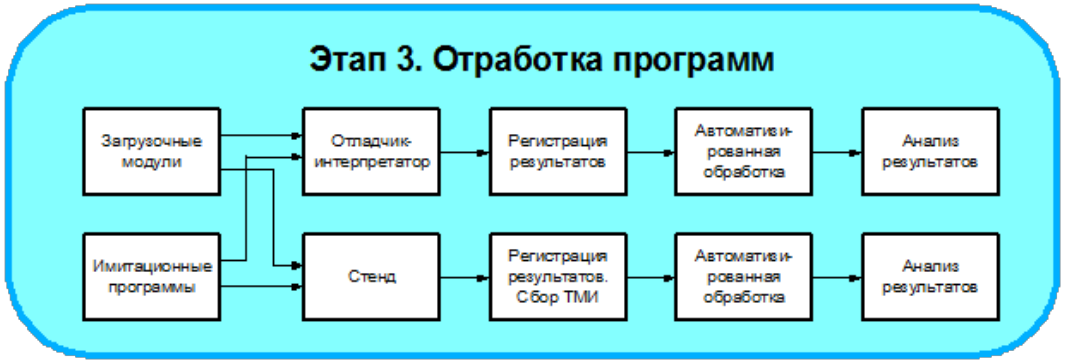
Инженеры

Кто обладает знаниями, тот и должен их формализовать



Программисты





Ц Е Л Ь

- **Облегчить и ускорить понимание алгоритмов Ч Е Л О В Е К О М**
- **Исключить ошибки в алгоритмах**
- **Заменить стандарт на блок-схемы ГОСТ 19.701-90 и ISO 5807-85 на стандарт языка ДРАКОН**

2

Формальная управляющая графика

Ц Е Л Ь

- **Заменить управляющие ключевые слова на управляющую графику**
- **Дополнить три базовые управляющие конструкции (последовательность, ветвление, цикл)**

Подробности

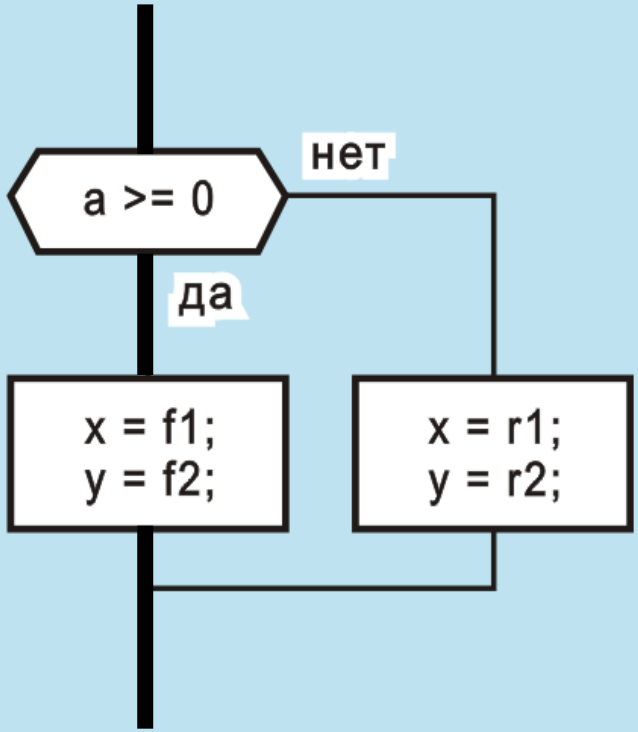
- Отказаться от управляющих ключевых слов:
  if, then, else, case, switch, break, while, do, repeat, until, for, foreach, continue, loop, exit, when, last и т. д.
- Заменить их на управляющую графику, то есть использовать двумерное структурное программирование

Как удалить
Оператор if else

Программа на языке Си

```
if ( a >= 0 )
{
    x = f1;
    y = f2;
}
else
{
    x = r1;
    y = r2;
}
```

Программа на языке Дракон–Си



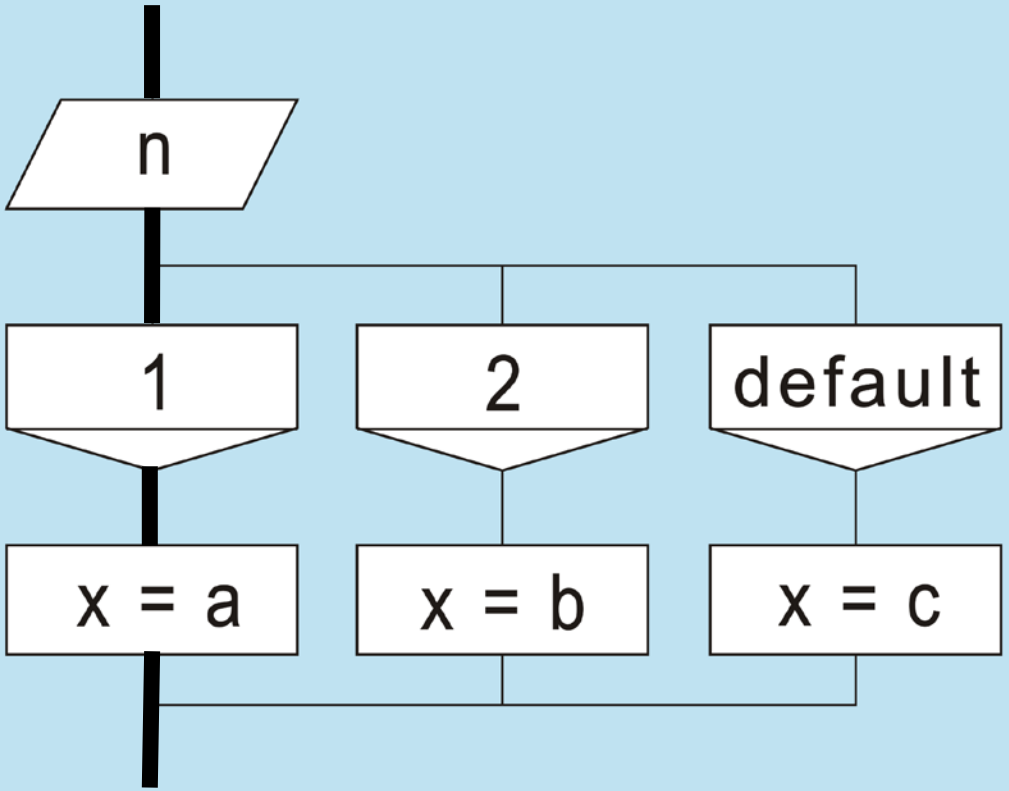
Как удалить

Операторы switch case break

Программа на языке Си

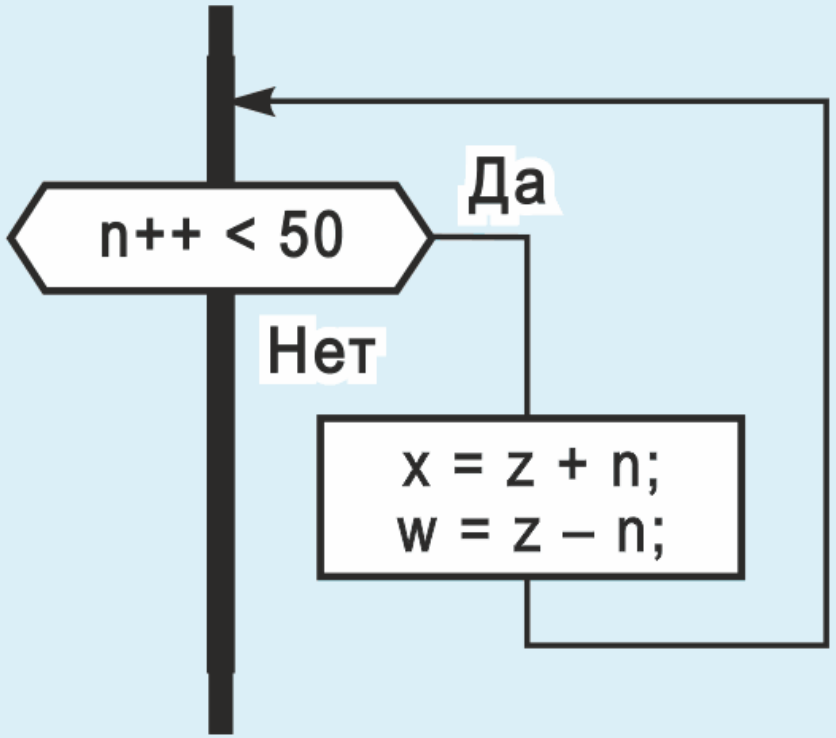
```
switch (n)
{
    case 1:
        x = a;
        break;
    case 2:
        x = b;
        break;
    default:
        x = c;
} /* switch */
```

Программа на языке Дракон–Си



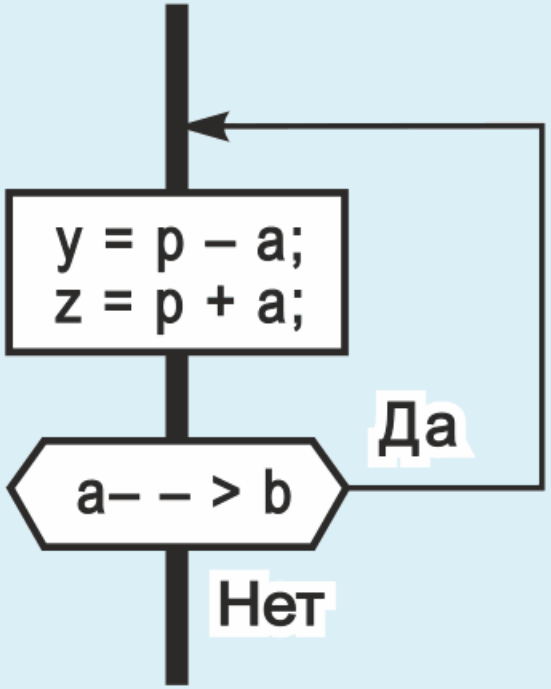
Как удалить
Оператор while

Программа на языке Си	Программа на языке Дракон–Си

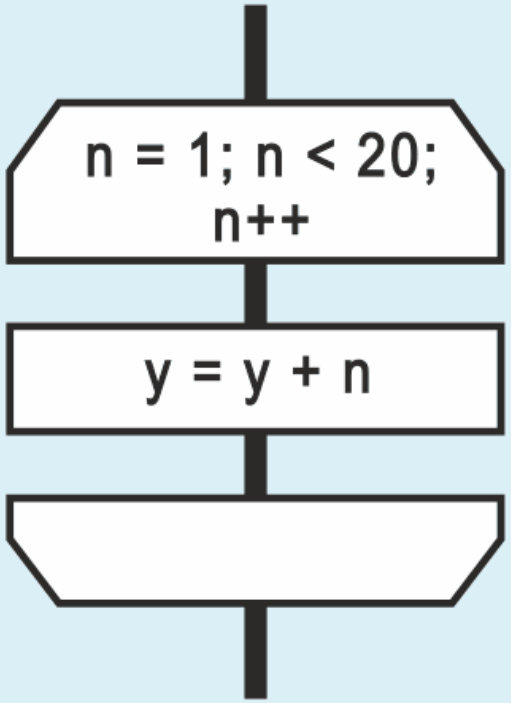
while (n++ < 50)
{
x = z + n;
w = z – n;
} /* while */
	

Как удалить
Оператор do while

Программа на языке Си	Программа на языке Дракон–Си

do
{
y = p – a;
z = p + a;
} while (a– – > b);
/* do while */
	

Как удалить
Оператор for

Программа на языке Си	Программа на языке Дракон–Си
for (n=1; n<20; n++) y = y + n;	

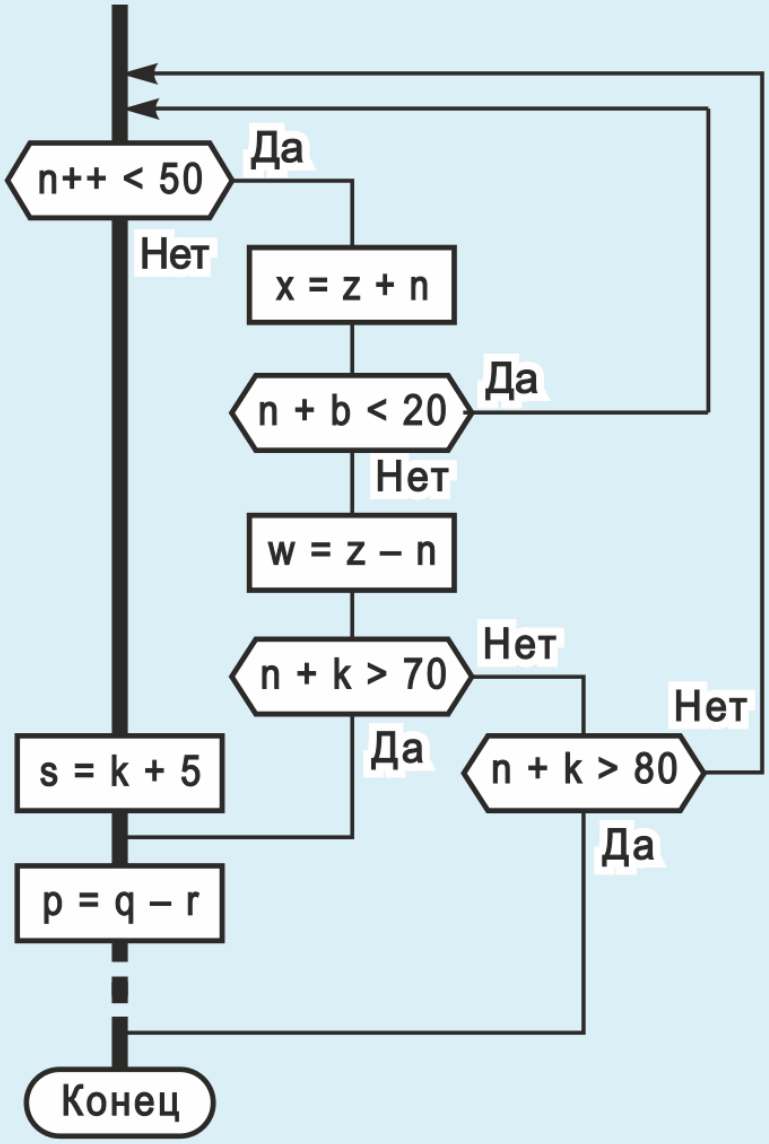
Как удалить Оператор continue

Оператор return

Программа на языке Си

```
while (n++ < 50)
{
    x = z + n;
    if (n + b < 20)
        continue;
    w = z – n;
    if (n + k > 70)
        go to m1;
    if (n + k > 80)
        return;
} /* while */
s = k + 5;
m1:  p = q – r;
.....
} /* end */
```

Программа на языке Дракон–Си



Главная И Д Е Я

- **Объединить АЛГОРИТМЫ и ЭРГОНОМИКУ**
- **Сделать алгоритмы эргономичными**

GUI, WYSIWYG это не то

3

Когнитивная эргономика

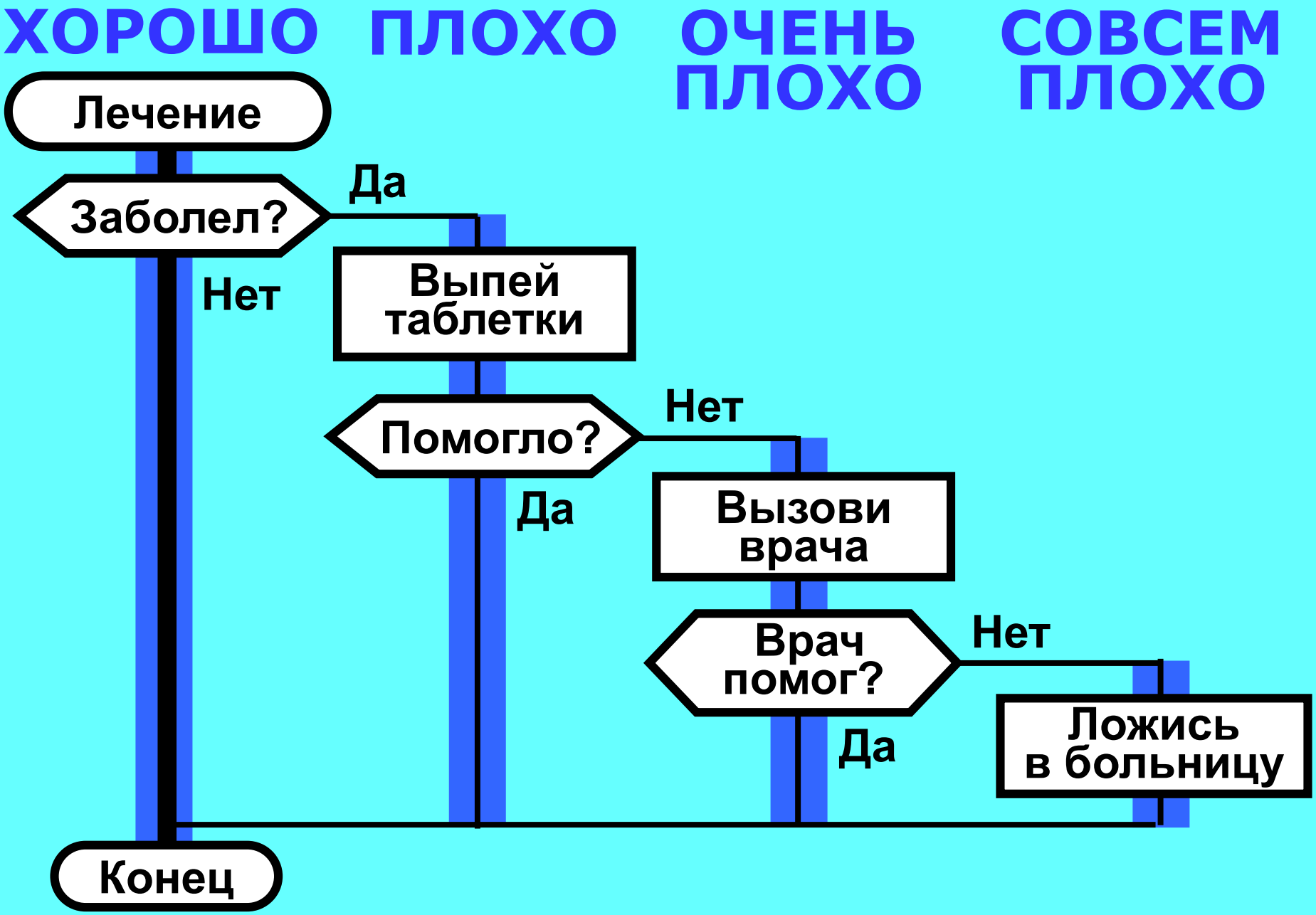
Эргономика интеллекта

Когнитивно-эргономический анализ алгебры Диофанта Александрийского III век

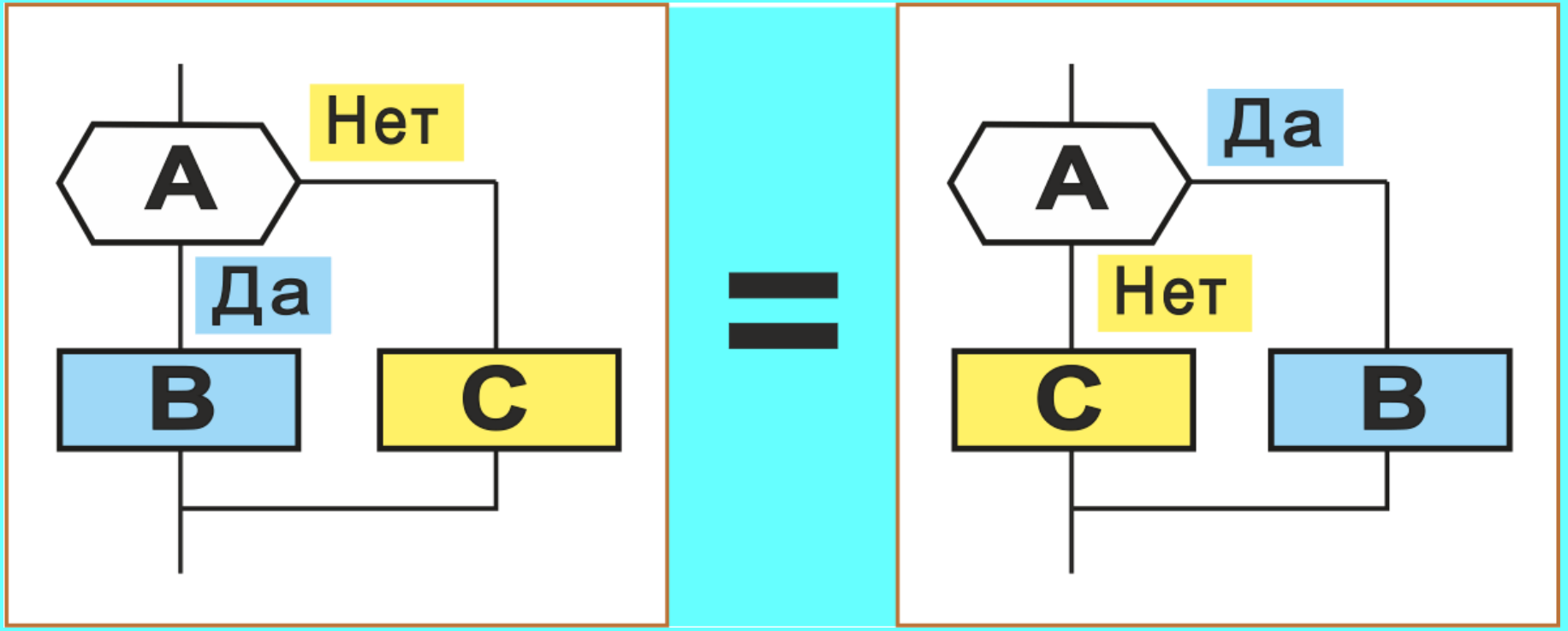
Картографический принцип языка ДРАКОН

Проекция Меркатора

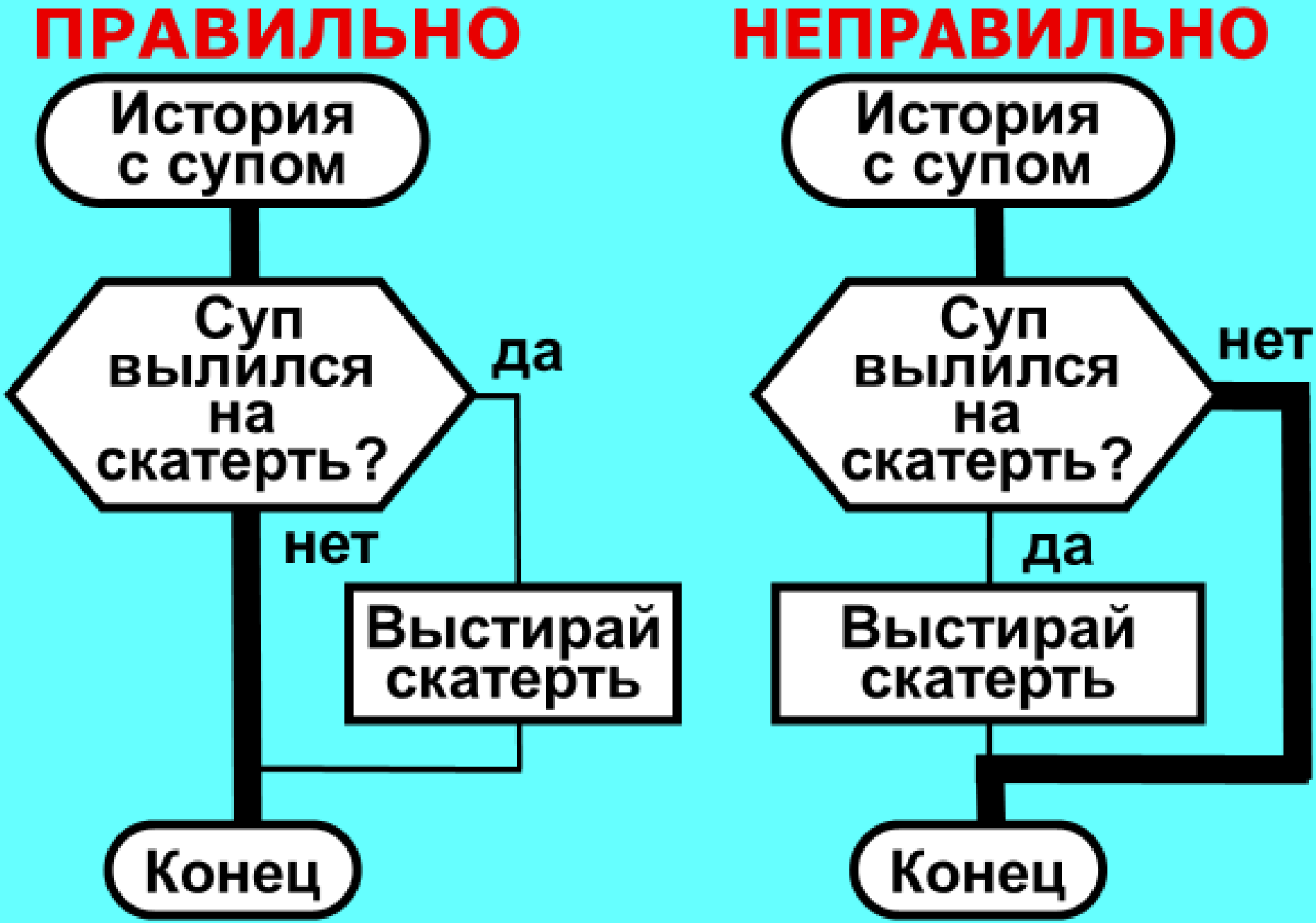
ЧЕМ ПРАВЕЕ, ТЕМ ХУЖЕ



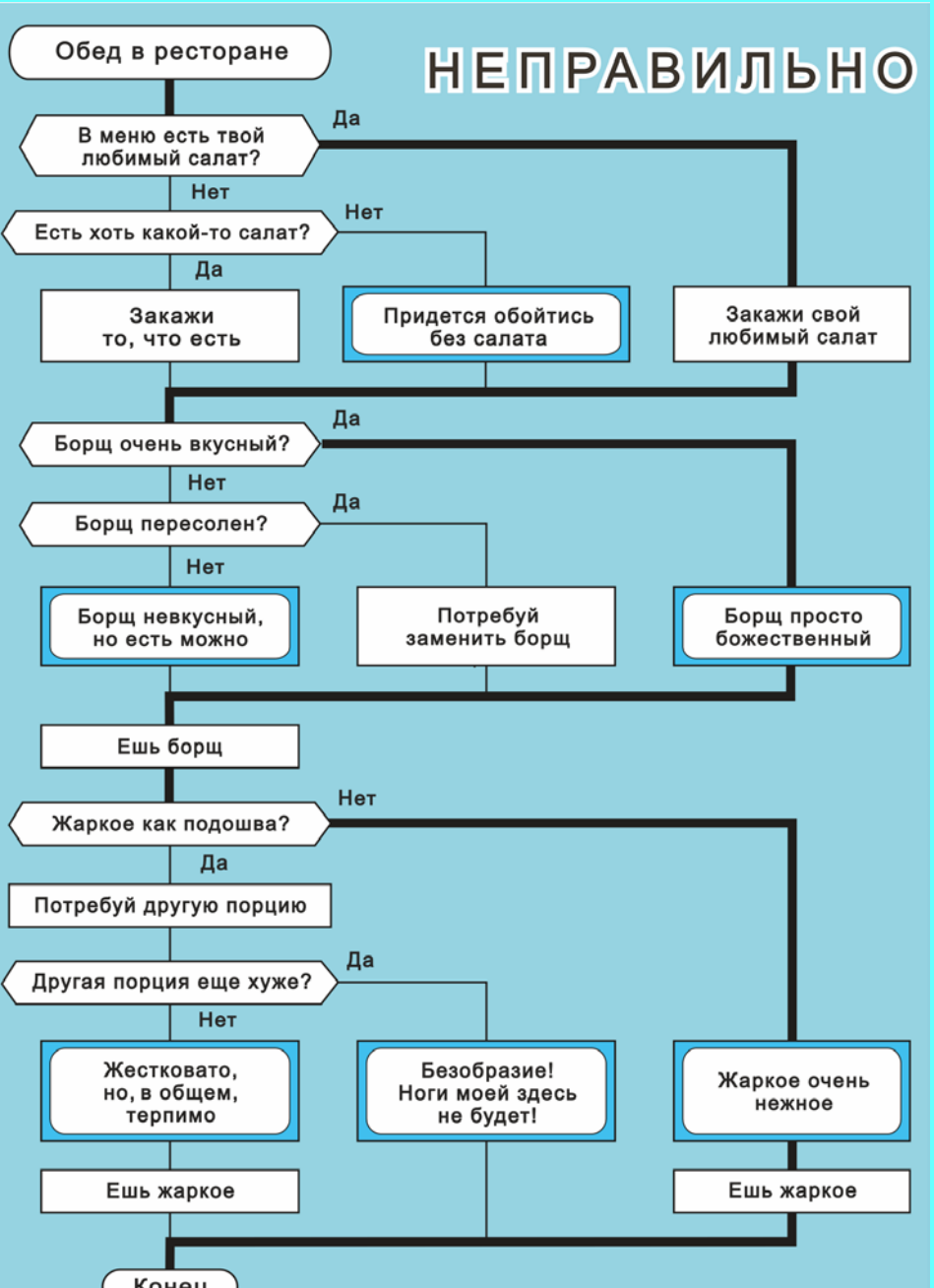
РОКИРОВКА

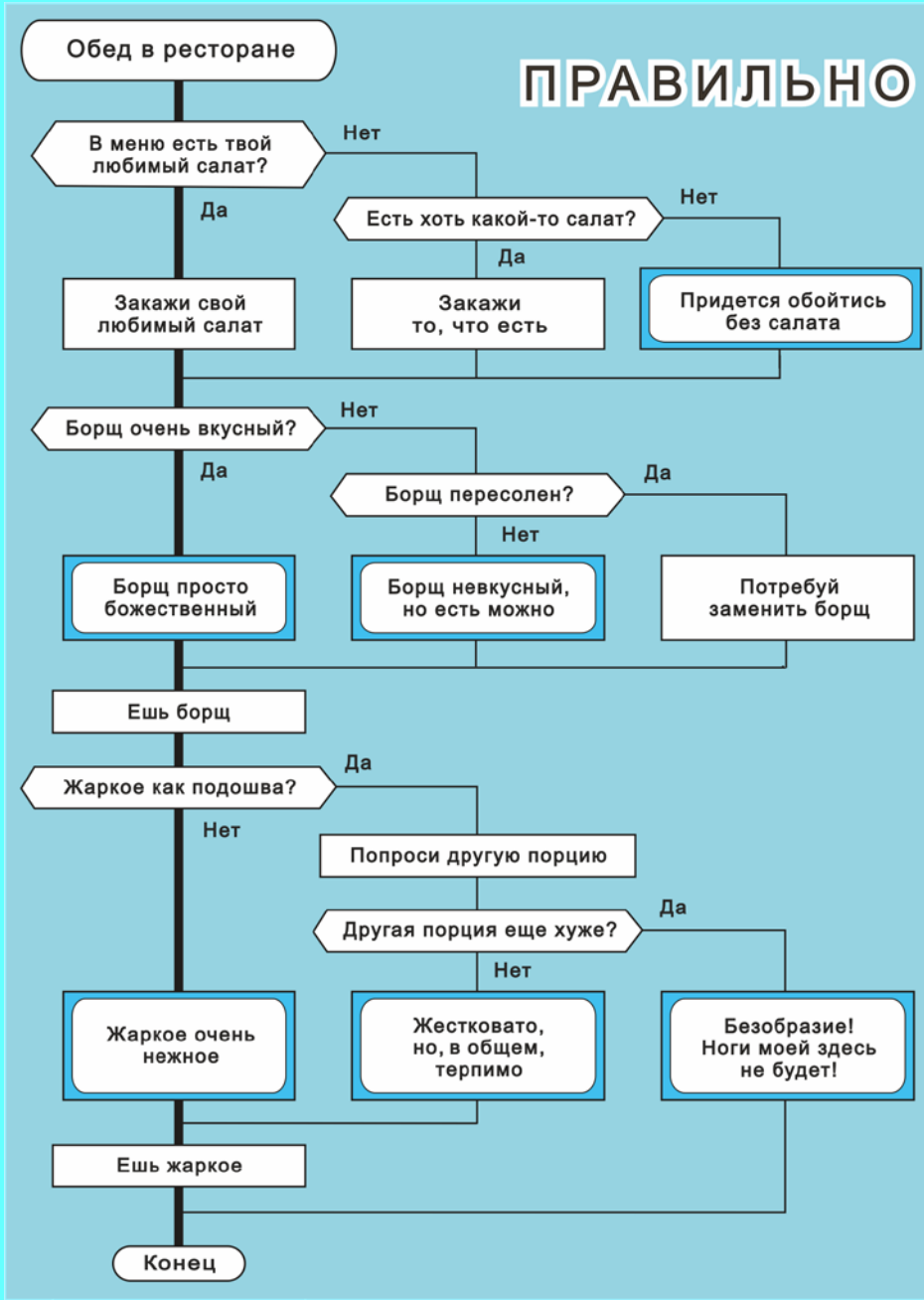


ГЛАВНЫЙ МАРШРУТ



Шесть рокировок





Язык ДРАКОН имеет два синтаксиса:

- графический синтаксис
- текстовый синтаксис

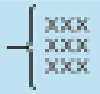
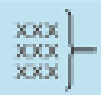
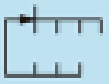
SDL Specification and Description Language

Графический алфавит языка ДРАКОН

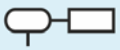
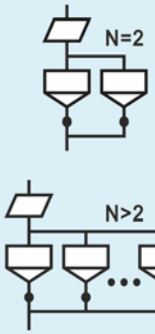
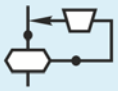
Графические буквы

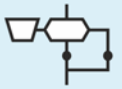
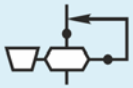
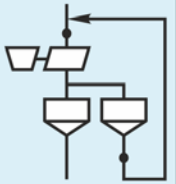
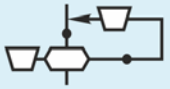
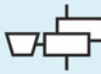
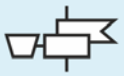
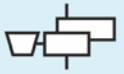
Графический алфавит языка ДРАКОН

	Икона	Название иконы
И1		Заголовок
И2		Конец
И3		Действие
И4		Вопрос
И5		Выбор
И6		Вариант
И7		Имя ветки
И8		Адрес
И9		Вставка
И10		Полка
И11		Формальные параметры
И12		Начало цикла ДЛЯ
И13		Конец цикла ДЛЯ

	Икона	Название иконы
И14		Вывод
И15		Ввод
И16		Пауза
И17		Период
И18		Пуск таймера
И19		Синхронизатор (по таймеру)
И20		Параллельный процесс
И21		Комментарий
И22		Правый комментарий
И23		Левый комментарий
И24		Петля цикла
И25		Петля силуэта
И26		Соединитель

Макроиконы языка ДРАКОН

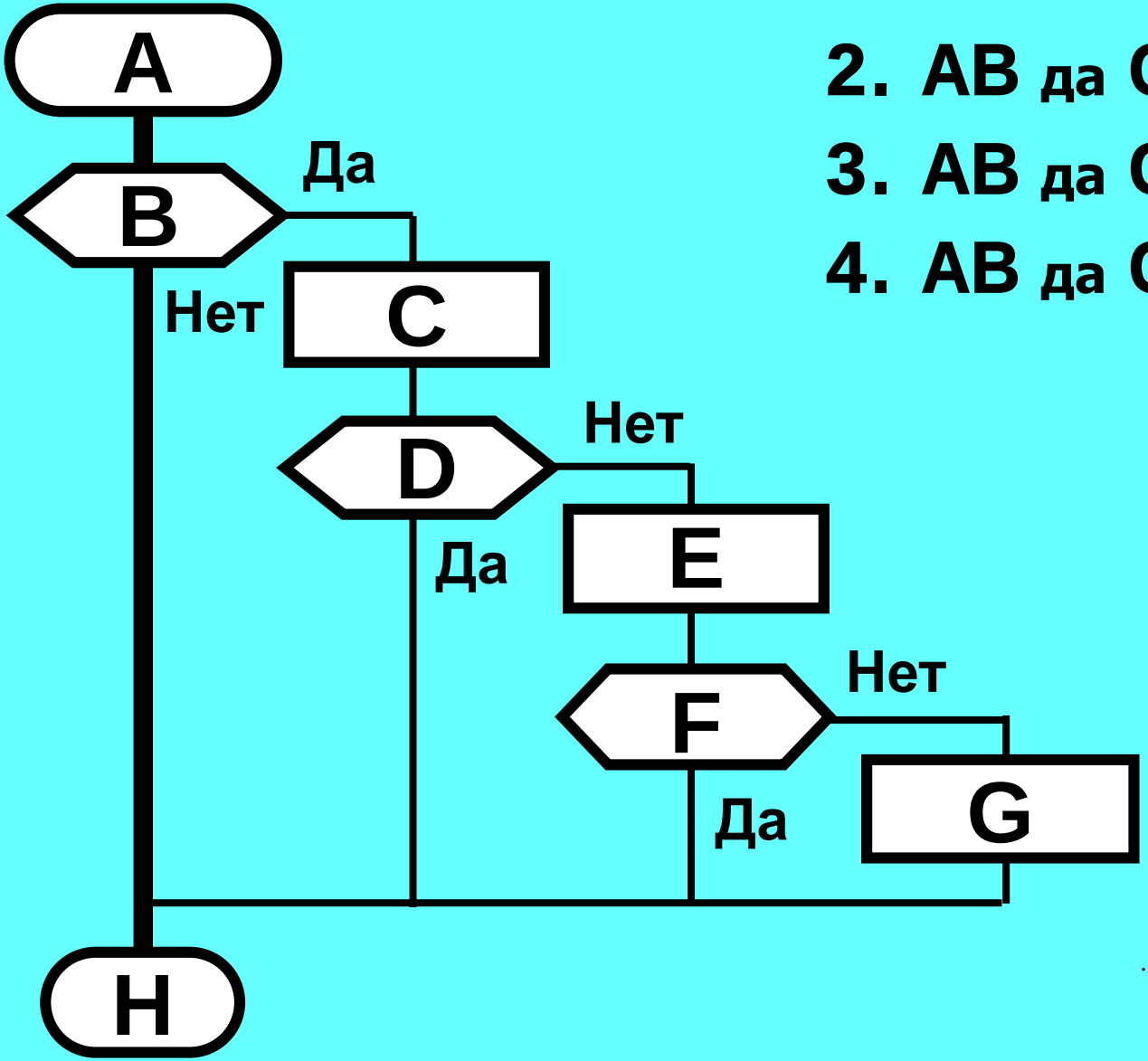
	Макроикона	Название макроиконы
1		Заголовок с параметрами
2		Развилка
3		Переключатель (число вариантов N>2)
4		Обычный цикл
5		Переключающий цикл
6		Цикл ДЛЯ
7		Цикл ЖДАТЬ
8		Действие по таймеру
9		Полка по таймеру

	Макроикона	Название макроиконы
10		Развилка по таймеру
11		Переключатель по таймеру
12		Обычный цикл по таймеру
13		Переключающий цикл по таймеру
14		Цикл ДЛЯ по таймеру
15		Цикл ЖДАТЬ по таймеру
16		Вставка по таймеру
17		Вывод по таймеру
18		Ввод по таймеру
19		Пуск таймера по таймеру
20		Параллельный процесс по таймеру

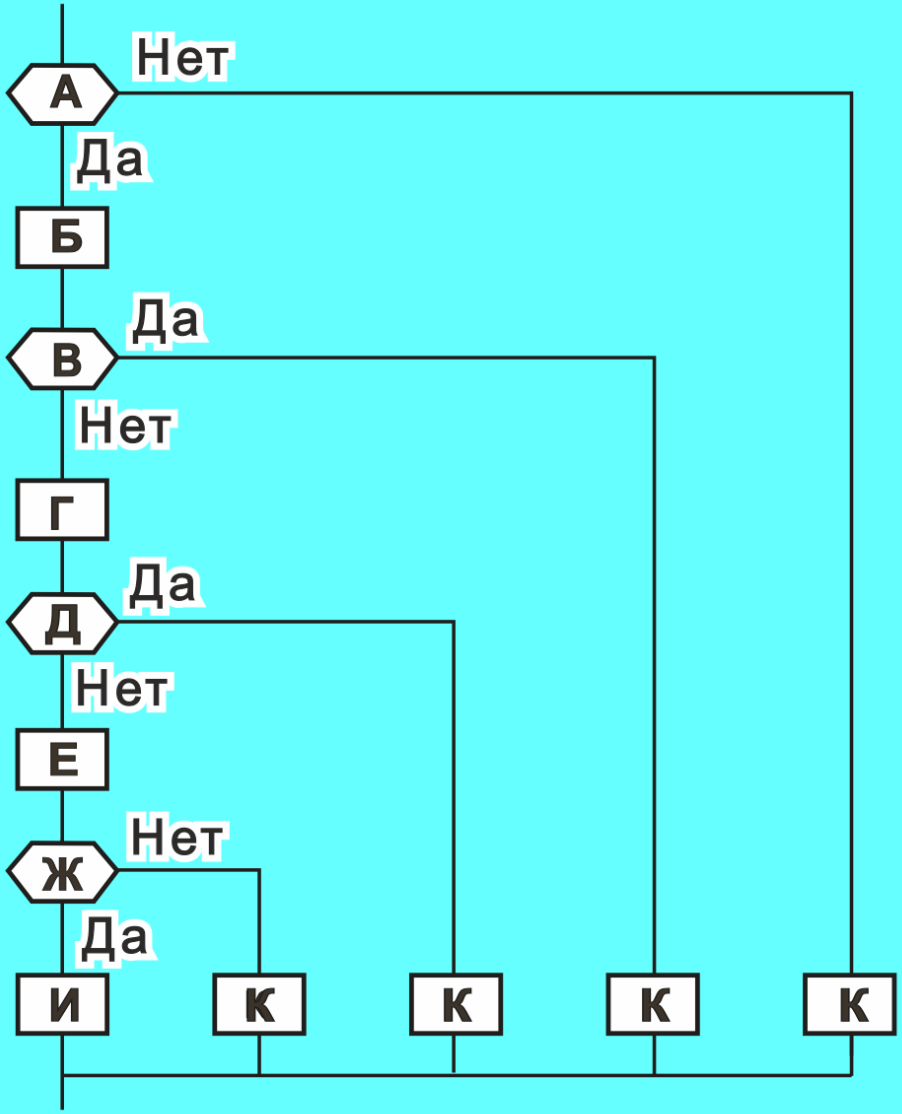
Формализация маршрутов алгоритма

Инвариант

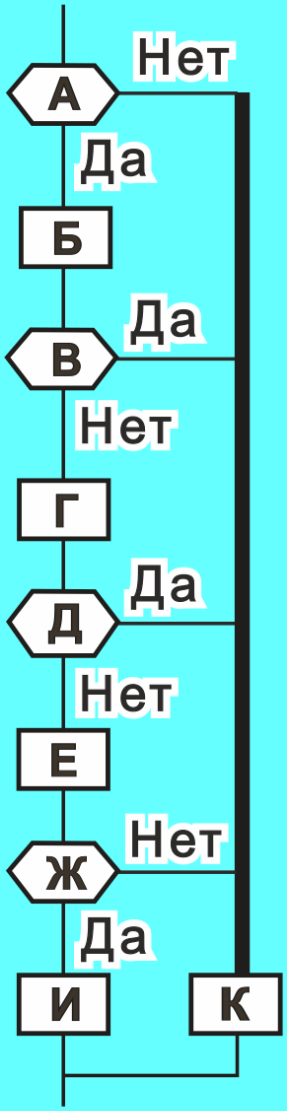
1. AB нет H
2. AB да CD да H
3. AB да CD нет EF да H
4. AB да CD нет EF нет GH



НЕПРАВИЛЬНО



ПРАВИЛЬНО



4

Визуальная логика

«Если это возможно, избегайте отрицаний в булевых выражениях. Представляется, что их понимание представля-ет трудность для многих программистов».

Эдвард Йодан

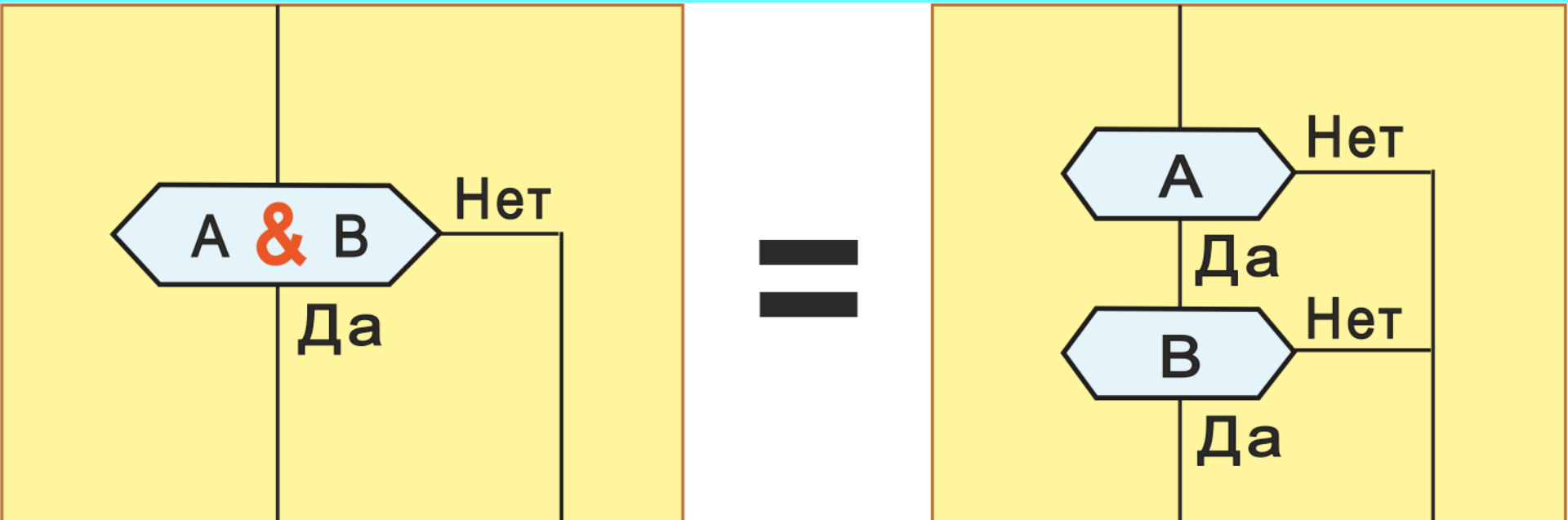
Использование символов алгебры логики

& V ¬

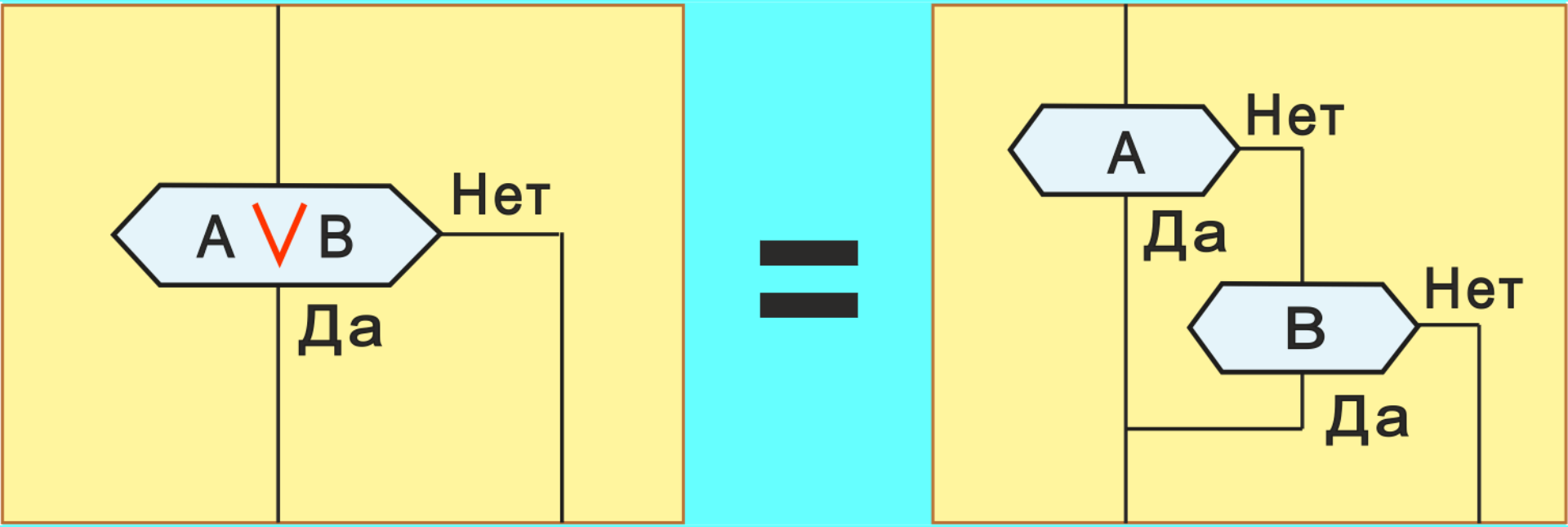
(И, ИЛИ, НЕ)

провоцирует появление ошибок

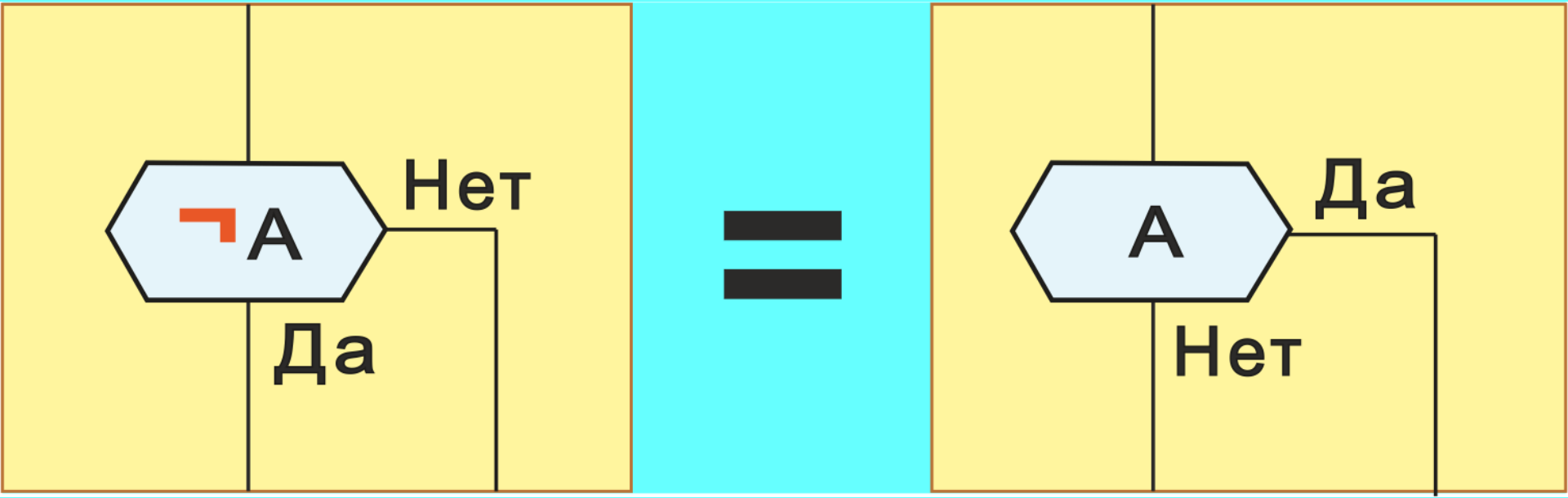
Как удалить знак конъюнкции & который провоцирует появление ошибок?



Как удалить знак дизъюнкции ∨ который провоцирует появление ошибок?



Как удалить знак отрицания ¬ который провоцирует появление ошибок?



В языке ДРАКОН логические связки

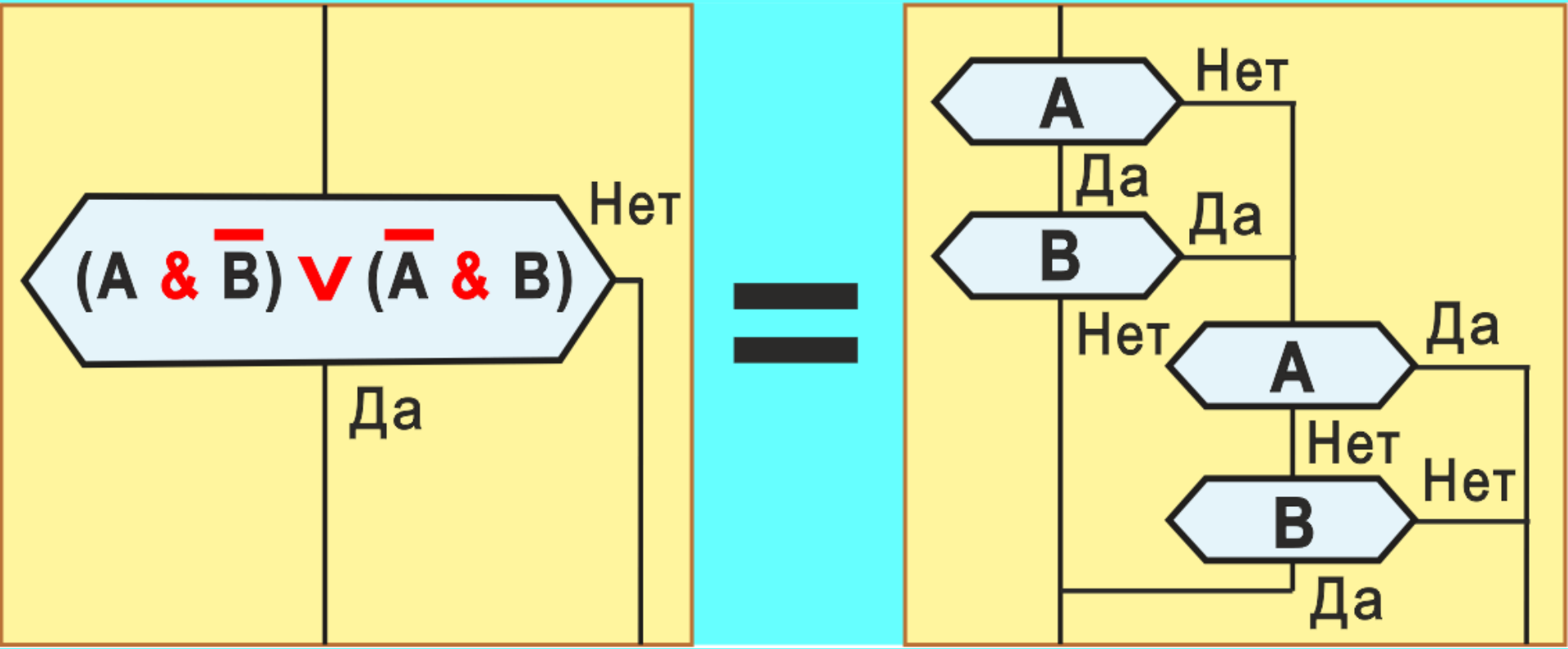
& ∨ ¬

(И, ИЛИ, НЕ)

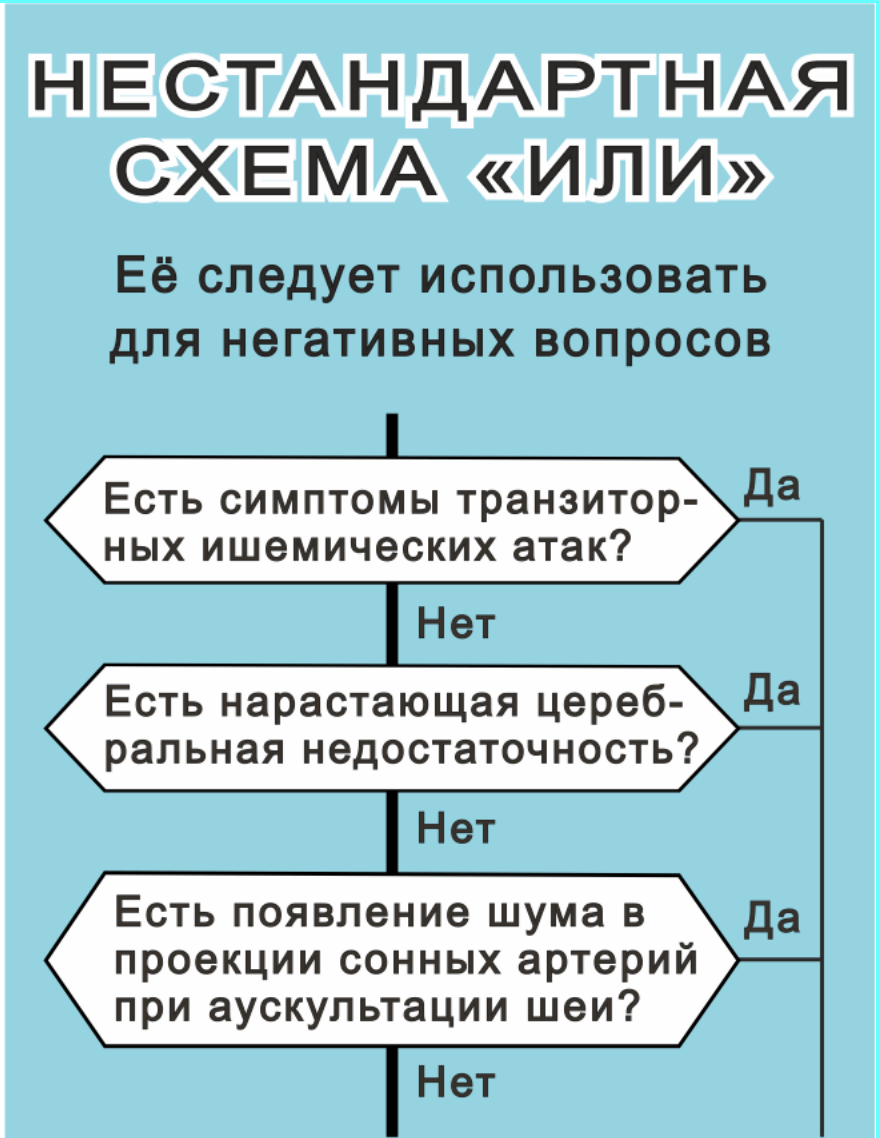
становятся ненужными.

Вместе с ними исчезают и вызванные ими ошибки

Исключающее «ИЛИ»



Запрещается использовать пропозициональные переменные



Неклассическая логика высказываний

Запрещено использовать:

- **Пропозициональные переменные**
- **Пропозициональные связки & ∨ ¬ (И, ИЛИ, НЕ)**
- **Слова «Истина» и «Ложь». Вместо них «Да» и «Нет»**

Недостаток линейной Булевой алгебры

Она скрывает часть проблем

Теорема. Если один выход логического фрагмента есть результат вычисления логической функции Z, то другой выход вычисляет ¬Z.

Пропозициональная формула – выражение, построенное из пропозициональных переменных с помощью пропозициональных связок & ∨ ¬ (Это общеизвестно)

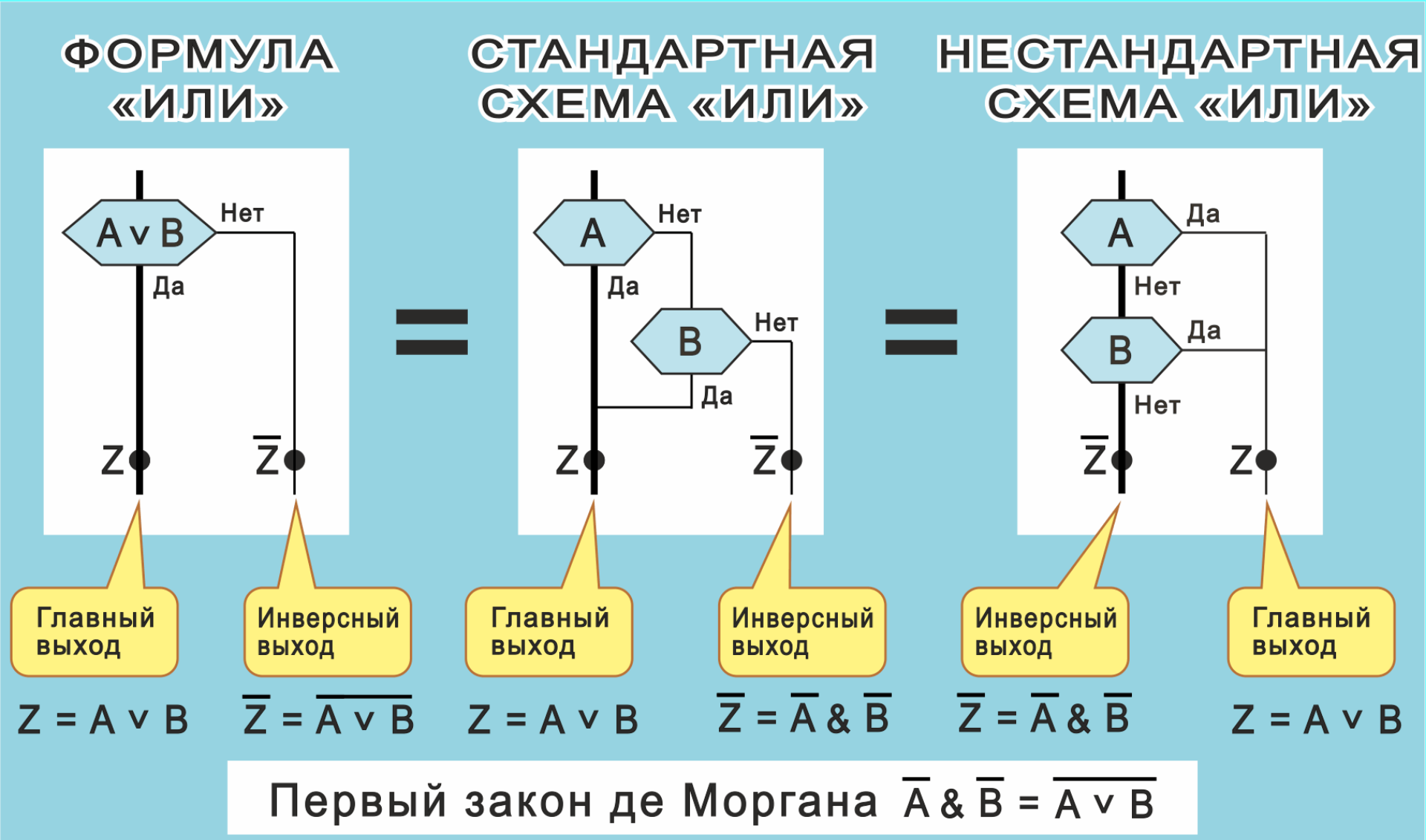
Линейная пропозициональная формула Z не показывает свою инверсию ¬Z

Визуальная пропозициональная формула Z показывает разом оба инверсных выхода:

1) выход Z

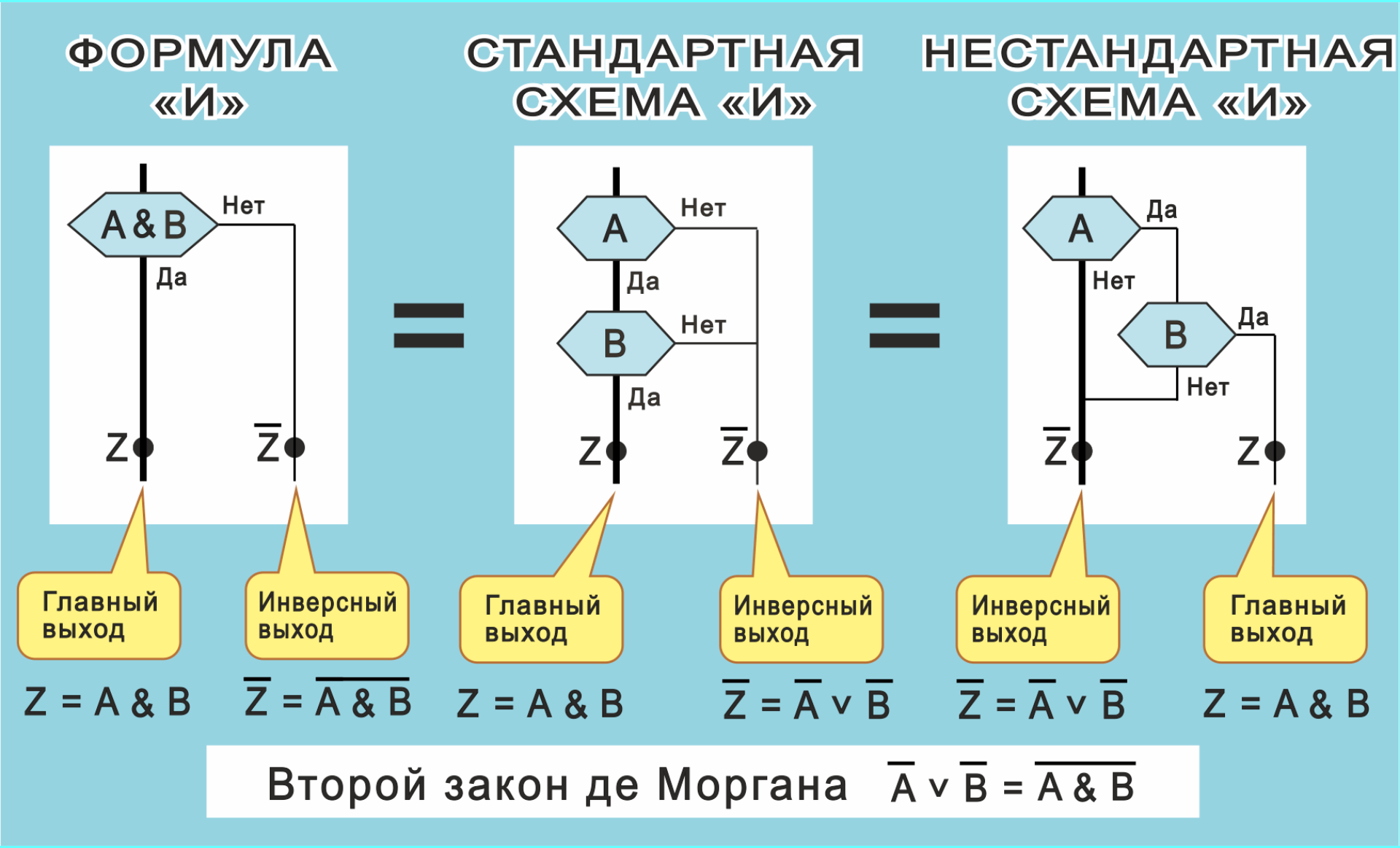
2) выход ¬Z

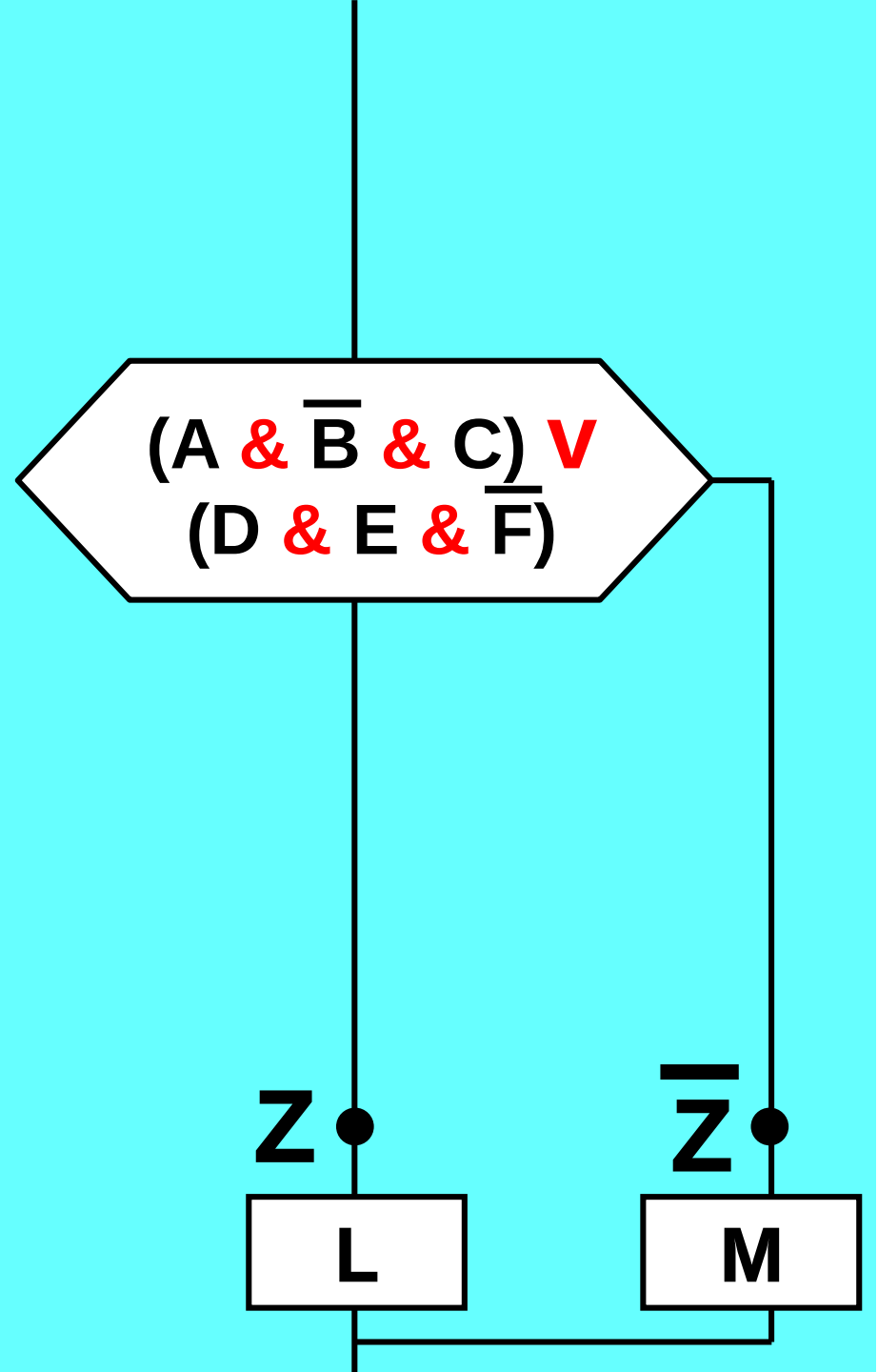
Три белых квадрата — три визуальных пропозициональных формулы. У каждой формулы два выхода: главный и инверсный

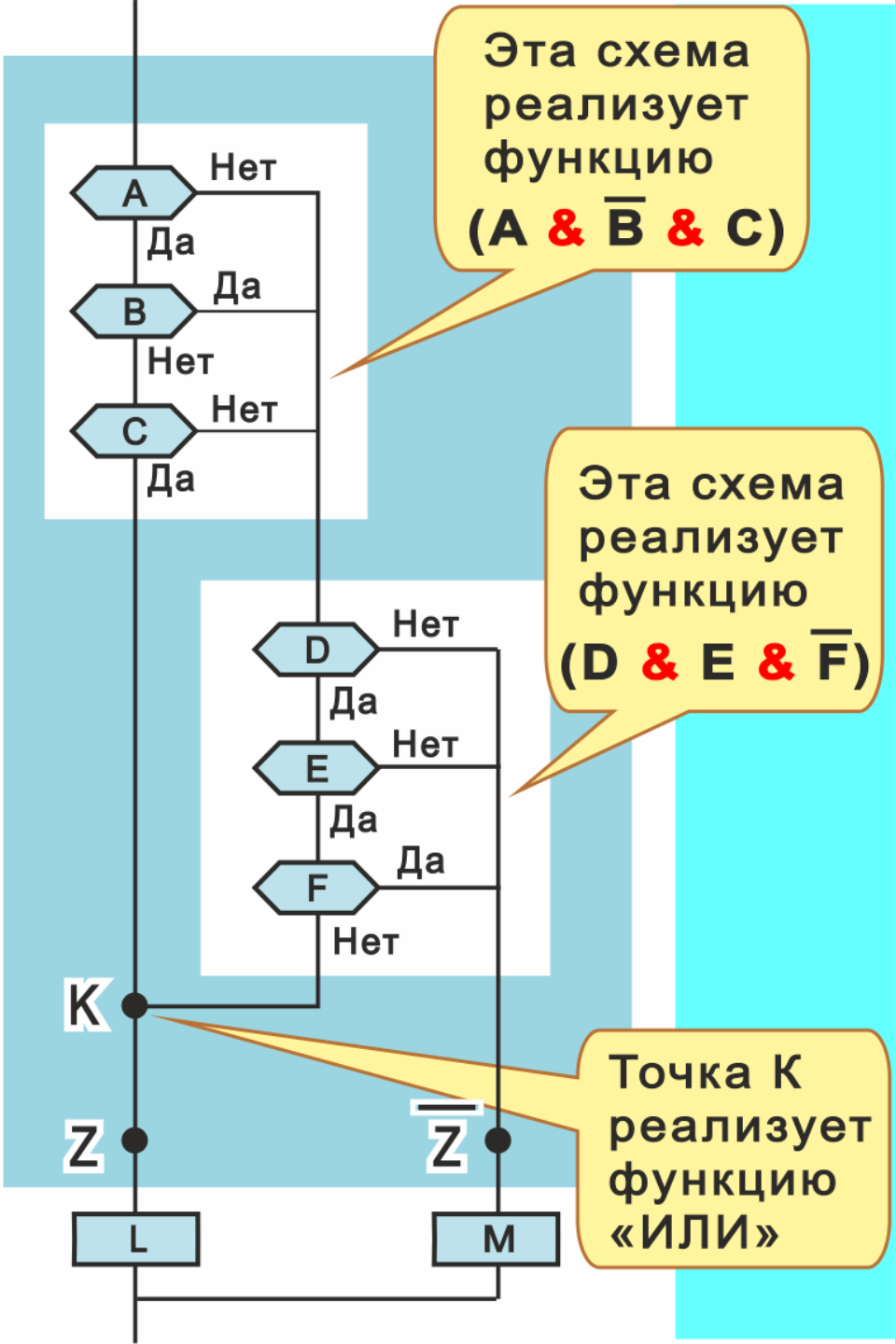


Первый закон де Моргана $\overline{A}$ & $\overline{B}$ = $\overline{A \vee B}$

То же самое, но для конъюнкции &




(A & $\overline{B}$ & C) V
(D & E & $\overline{F}$)
Z
$\overline{Z}$
L
M


A
Нет
Да
B
Да
Нет
C
Нет
Да
D
Нет
Да
E
Нет
Да
F
Да
Нет
K
Z
$\overline{Z}$
L
M
Эта схема реализует функцию (A & $\overline{B}$ & C)
Эта схема реализует функцию (D & E & $\overline{F}$)
Точка К реализует функцию «ИЛИ»

Недостаток линейной логики высказываний состоит в том, что:

Любая формула классической логики высказываний демонстрирует лишь одно высказывание из инверсной пары высказываний и игнорирует парное (инверсное к нему) высказывание.

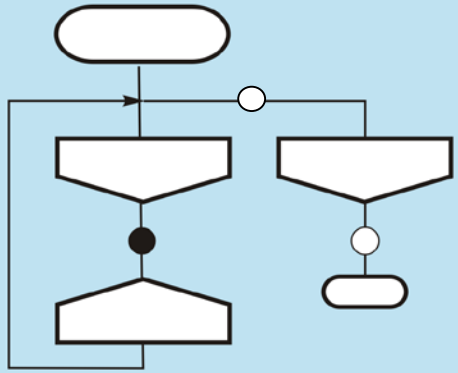
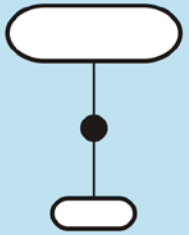
Математическая логика

- Визуальное логи-ческое исчисление
- Метод визуального логического вывода

5
Визуальное логическое исчисление

«Исчисление икон»

Валентные точки языка ДРАКОН

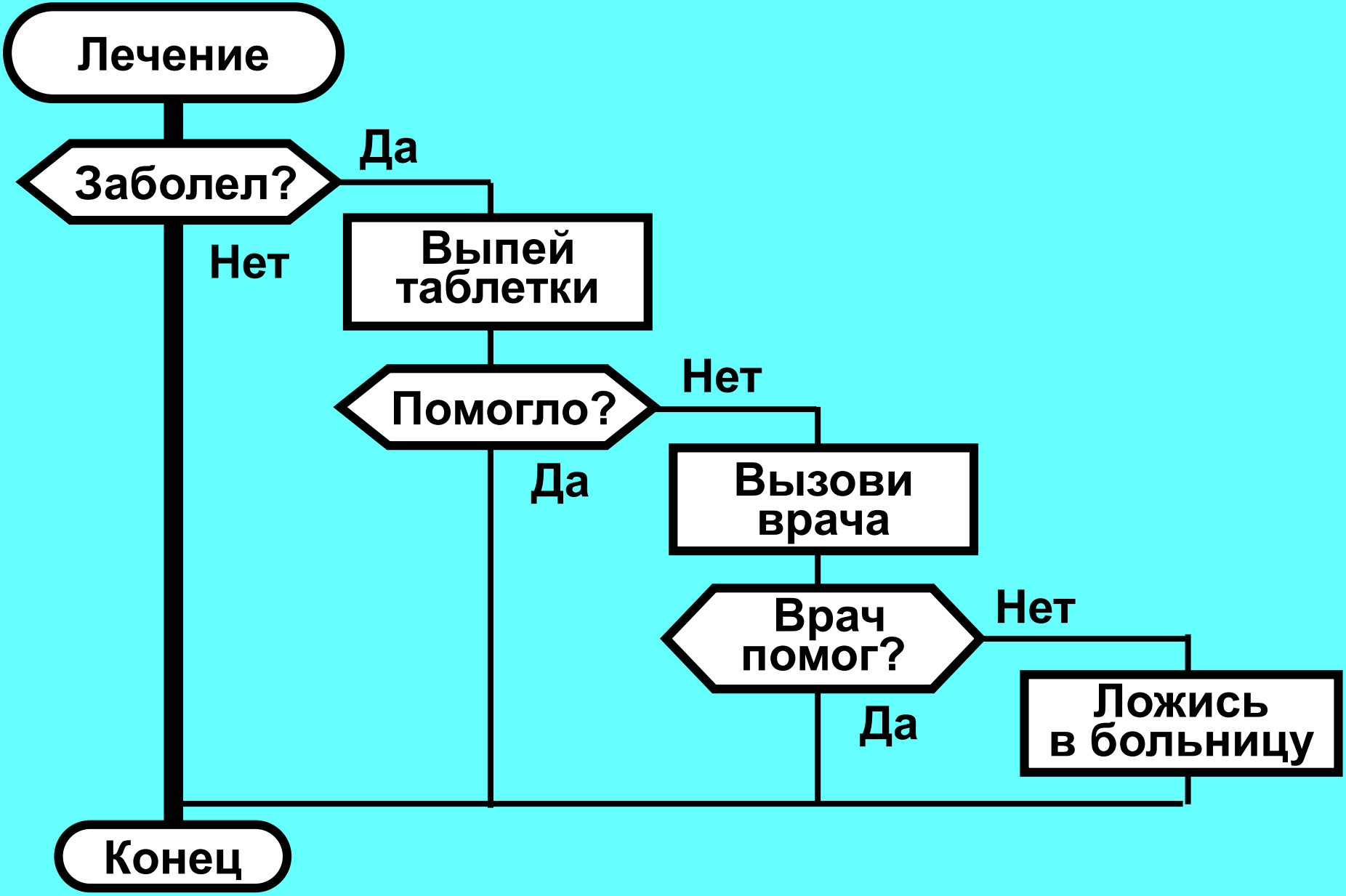
Аксиома-силуэт	Аксиома-примитив
	

Из этих аксиом методом визуального логического вывода выводятся строго доказанные теоремы (шампур-схемы)

«Открытие простых символических обозначений, которые сами приводят к манипуляциям по формальным правилам, явилось одним из путей, на которых развивалась мощь современной математики».

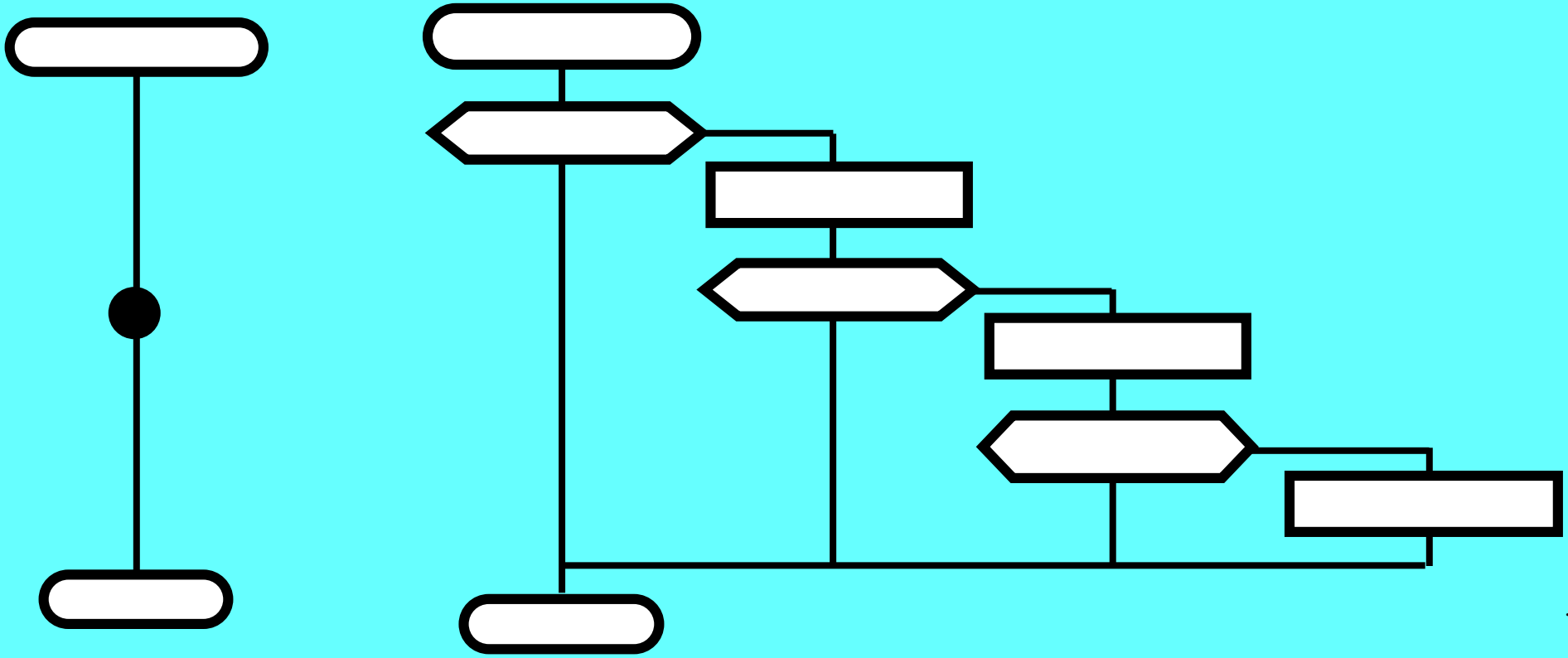
Стефен Клини

ЧЕМ ПРАВЕЕ, ТЕМ ХУЖЕ

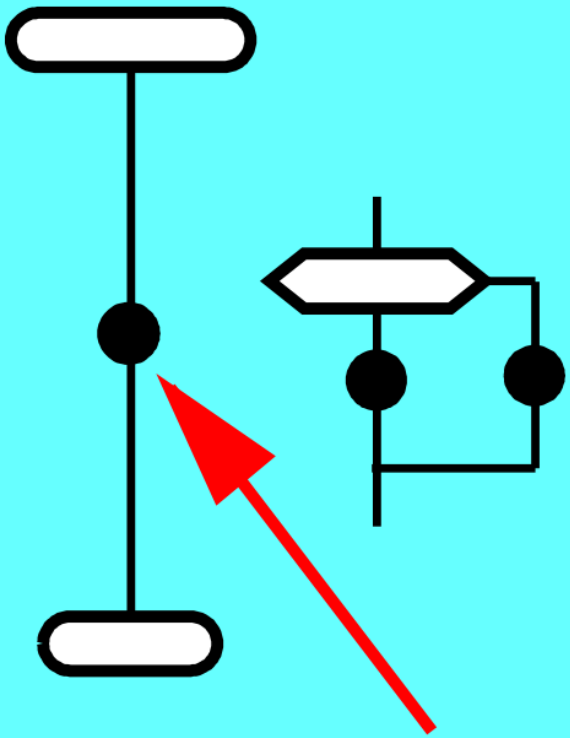


Аксиома-примитив

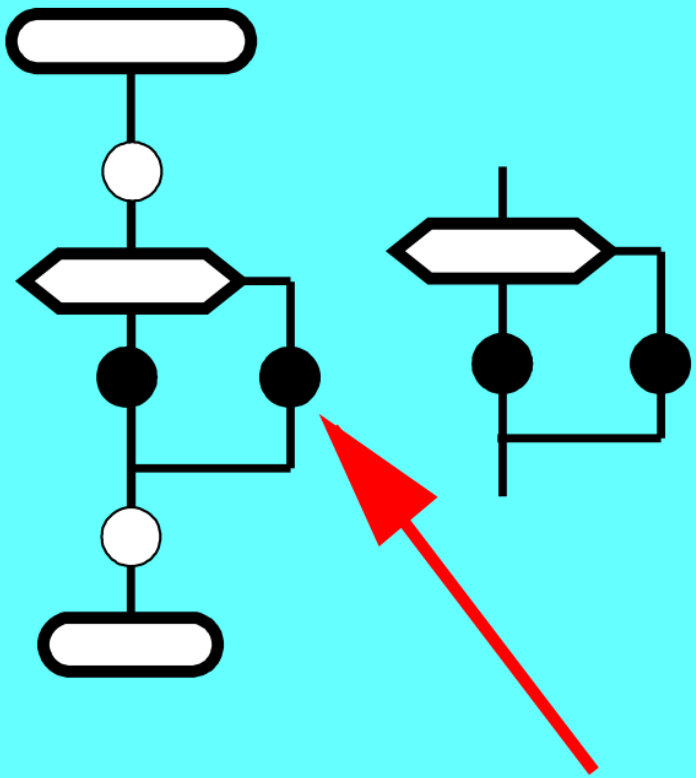
Теорема-примитив



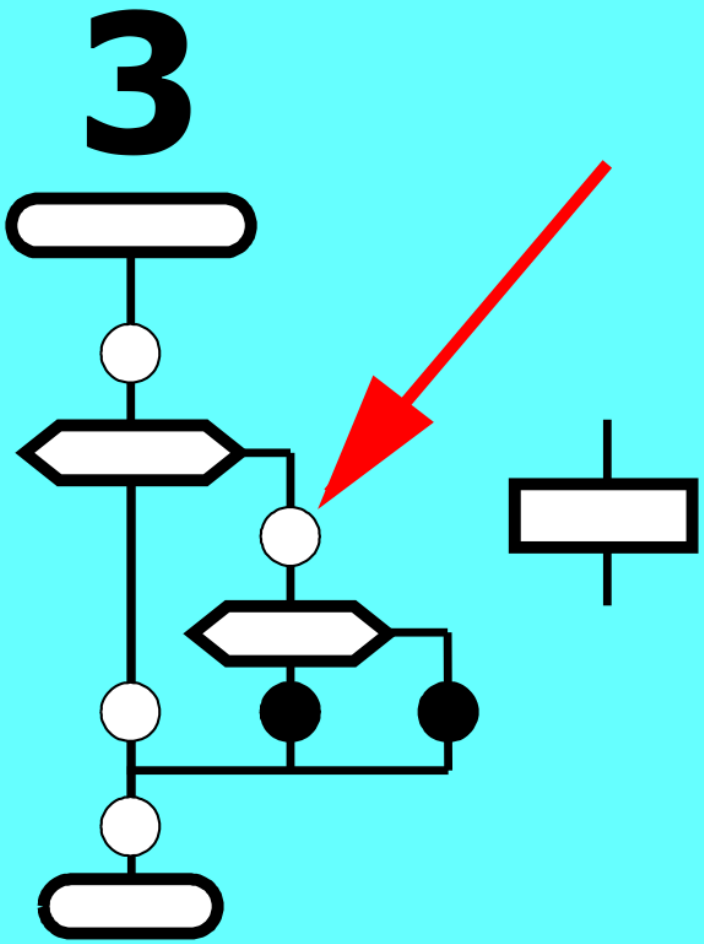
1

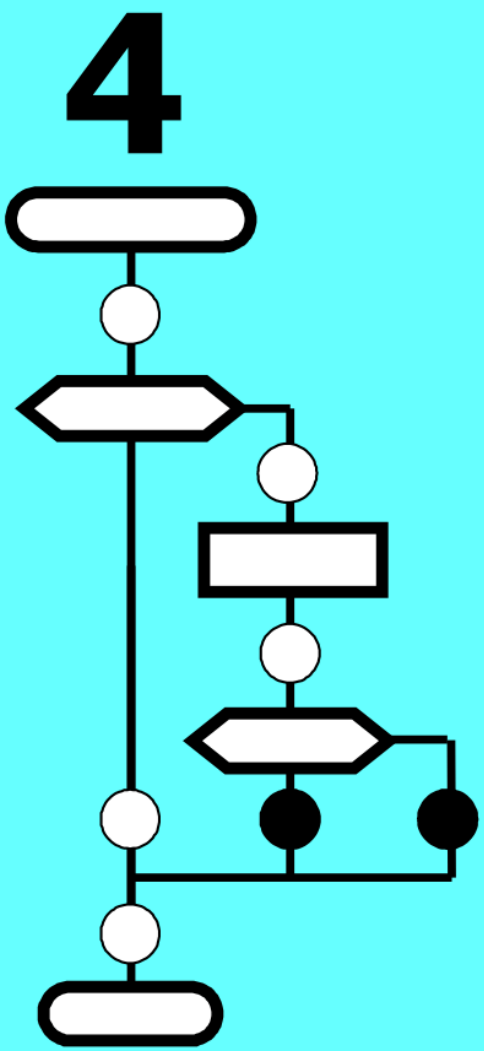


2



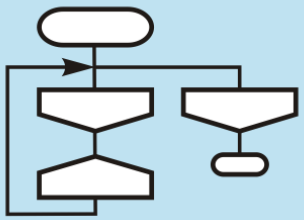
Теорема 2 получена из аксиомы 1 методом визуального логического вывода





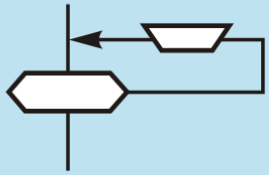
Теоремы 3 и 4 получены из аксиомы 1 методом визуального логического вывода

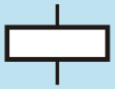
меню

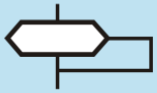

Силуэт


Примитив


Пауза

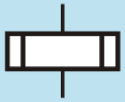

Цикл ЖДАТЬ

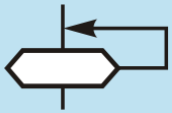

Действие


Развилка

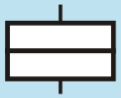

Таймер

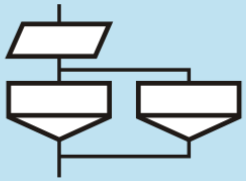

Формальные параметры

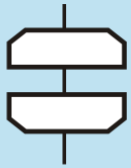

Вставка

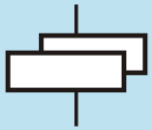

Обычный цикл

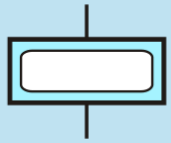

Синхронизатор

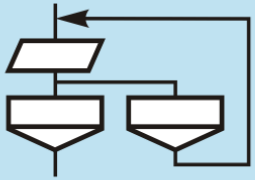

Полка

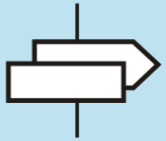

Переключатель

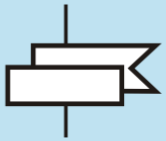

Цикл ДЛЯ


Параллельный процесс


Комментарий


Переключающий цикл


Вывод


Ввод


Соединитель

- **ДРАКОН-конструктор**

  Геннадия Тышова
- **Онлайн ДРАКОН-конструктор**

  Степана Митькина
- **Офлайн ДРАКОН-конструктор**

  Степана Митькина
- **ДРАКОН-конструктор**

  Эдуарда Ильченко
- **ДРАКОН-конструктор**

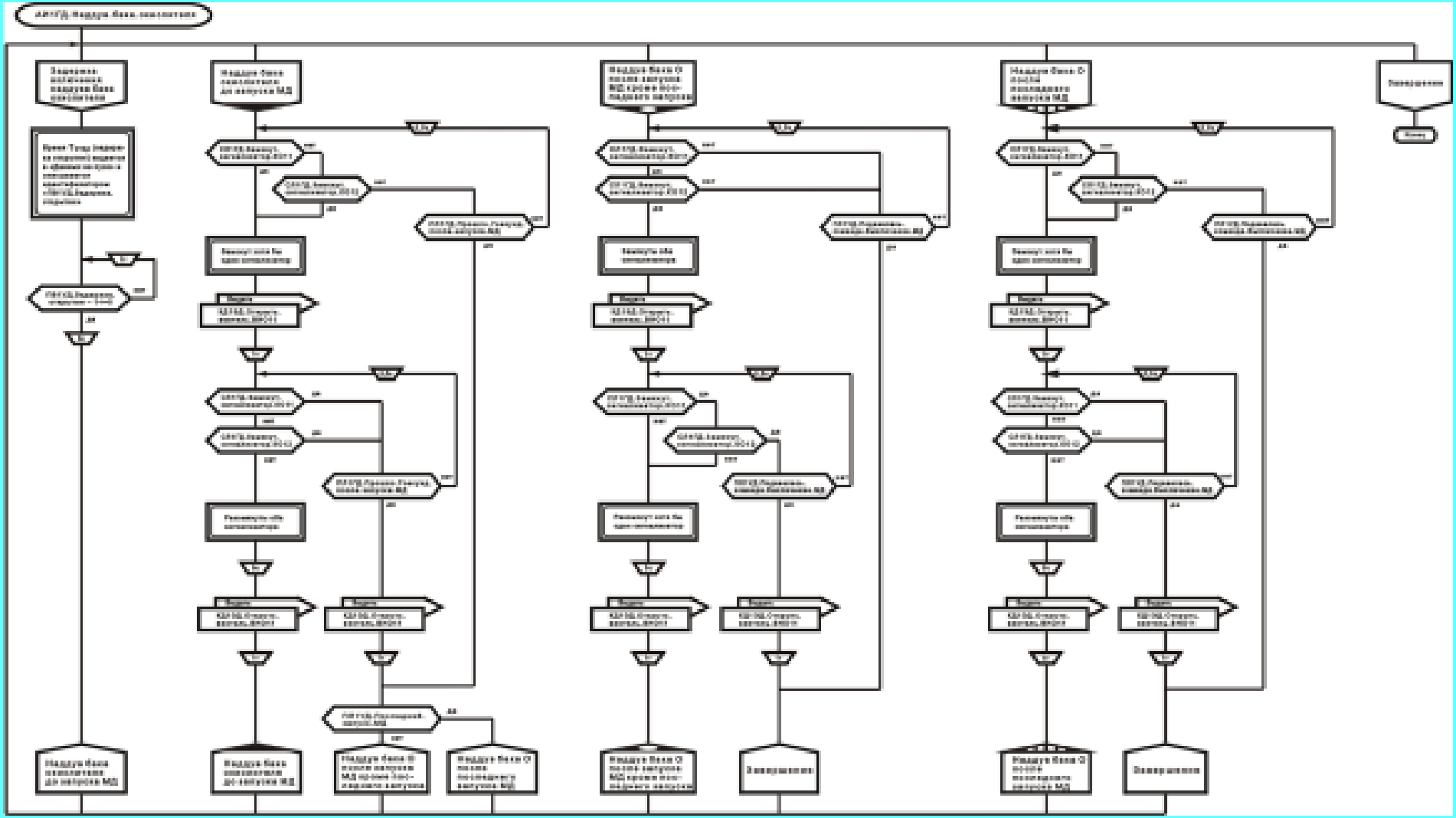
  Артема Бразовского
- **ДРАКОН-конструктор**

  Леонида Эйсымонта

6

Алгоритмическая конструкция «Силуэт»

Алгоритм «Наддув бака окислителя»



ДРАКОН-СХЕМА

Первая помощь при химическом ожоге глаза жидкостью

Промывание глаза водой

Закапать в конъюнктивальный мешок
- 0,25% р-р дикаина;
- или 2% р-р новокаина

Промывать глаз водой

При промывании надо выворачивать веки или оттягивать их от глазного яблока

Промывать глаз можно:
- свободно текущей водой из крана
- из резинового баллона объемом 10--20 мл
- из шприца объемом 20 мл
- из ундинки
- из кружки Эсмарха, подвешенной на высоте 1,5–2 м над больным

Промывание глаза нейтрализатором

Промывание глаза нейтрализатором

Опустите в жидкость лакмусовую бумажку

Удалось определить жидкость?

нет

да

Какая жидкость

Щелочь

Кислота

Промывать глаз 2,5% раствором борной кислоты

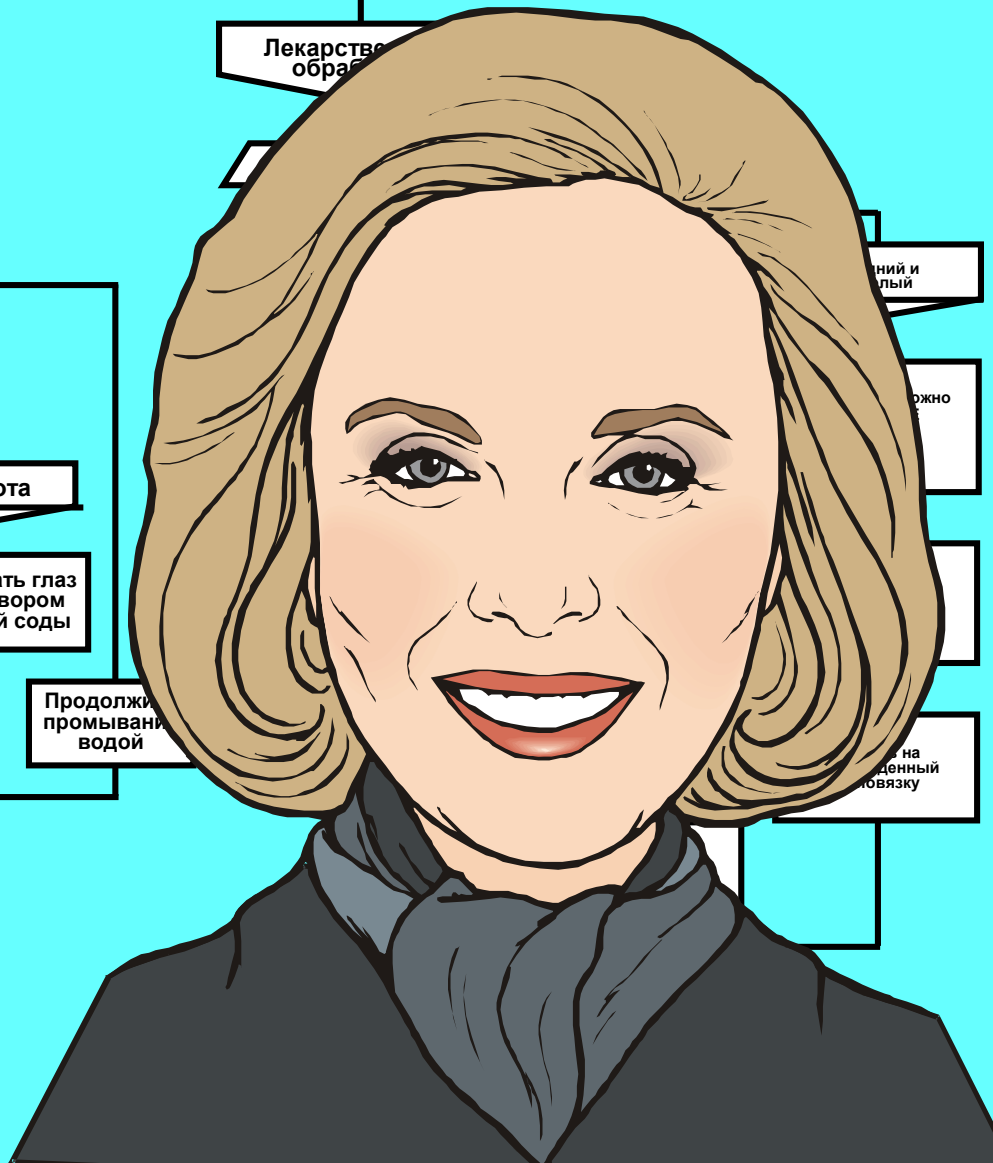
Промывать глаз 2% раствором питьевой соды

Продолжи... промывани... водой

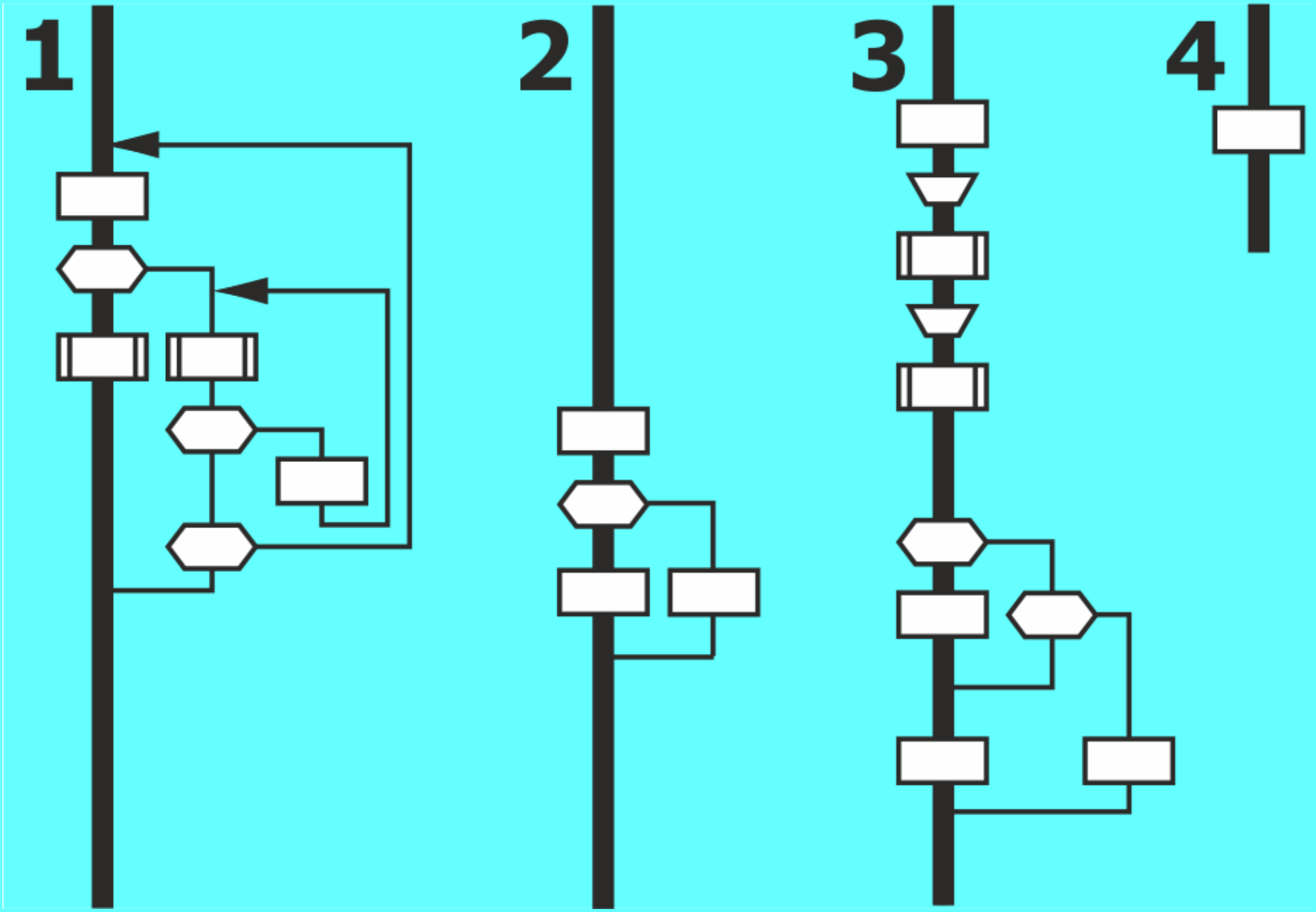
10–15 минут

Прекратить промывание

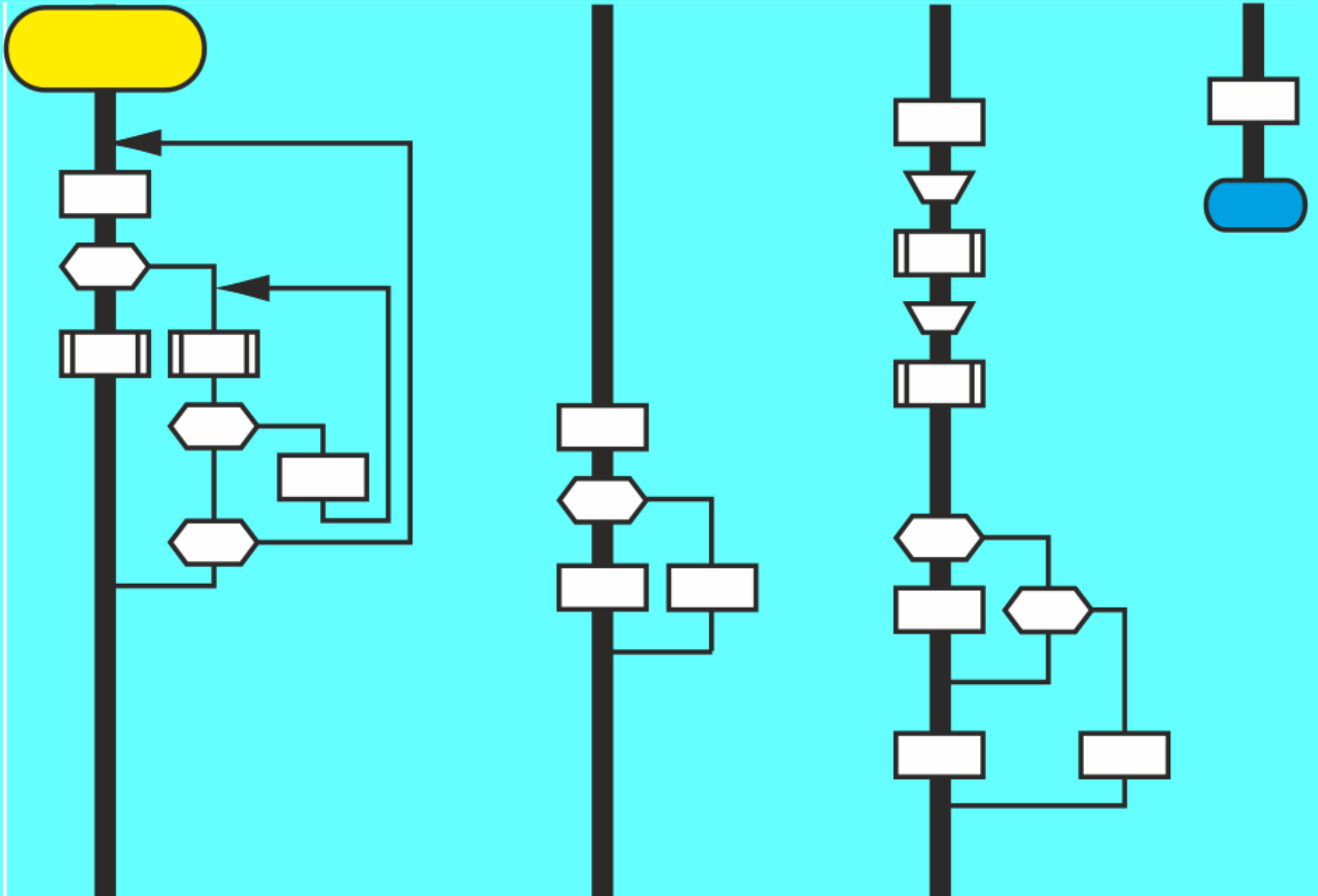
Лекарственная обработка



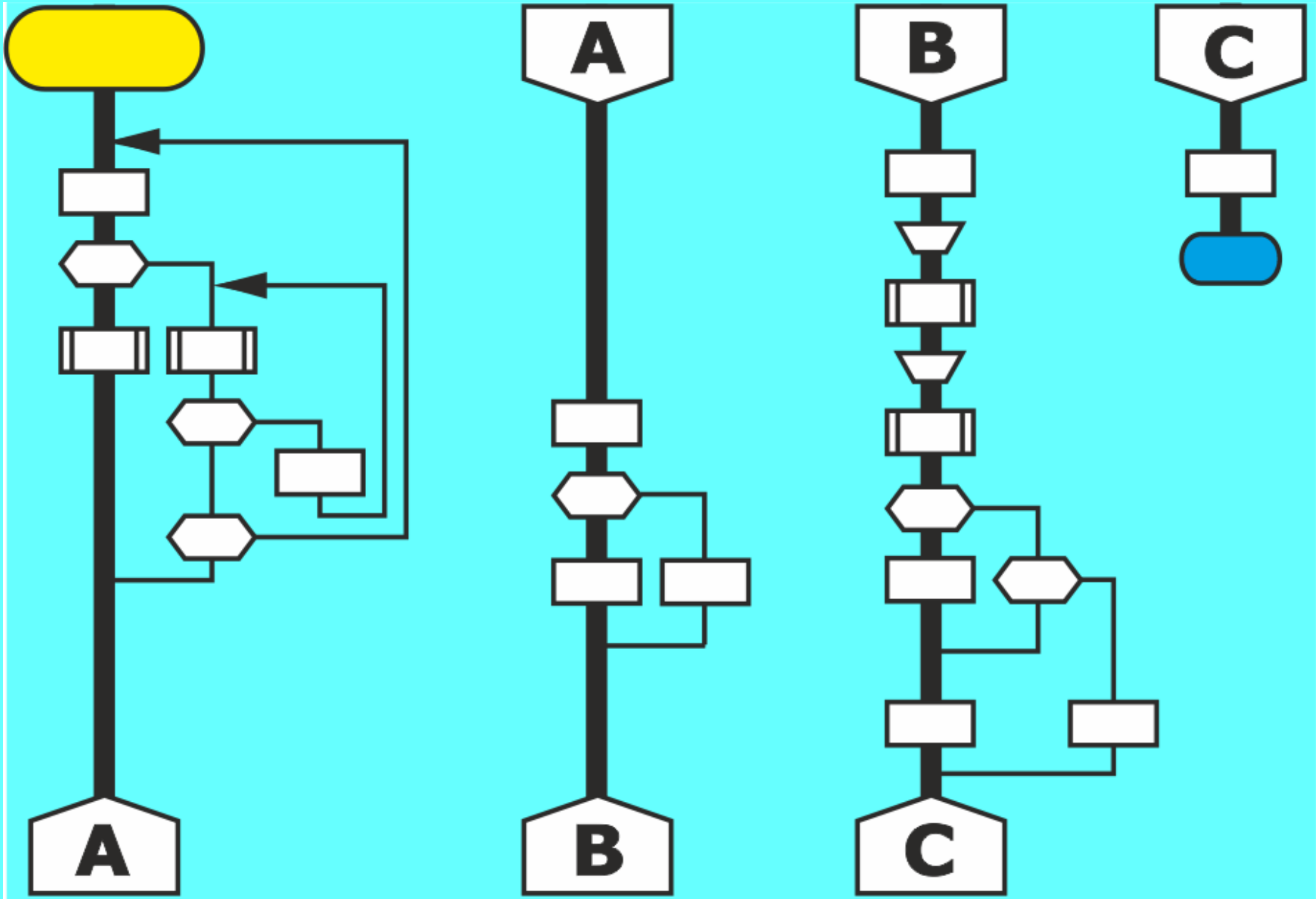
Что такое СИЛУЭТ. *Шаг 1*



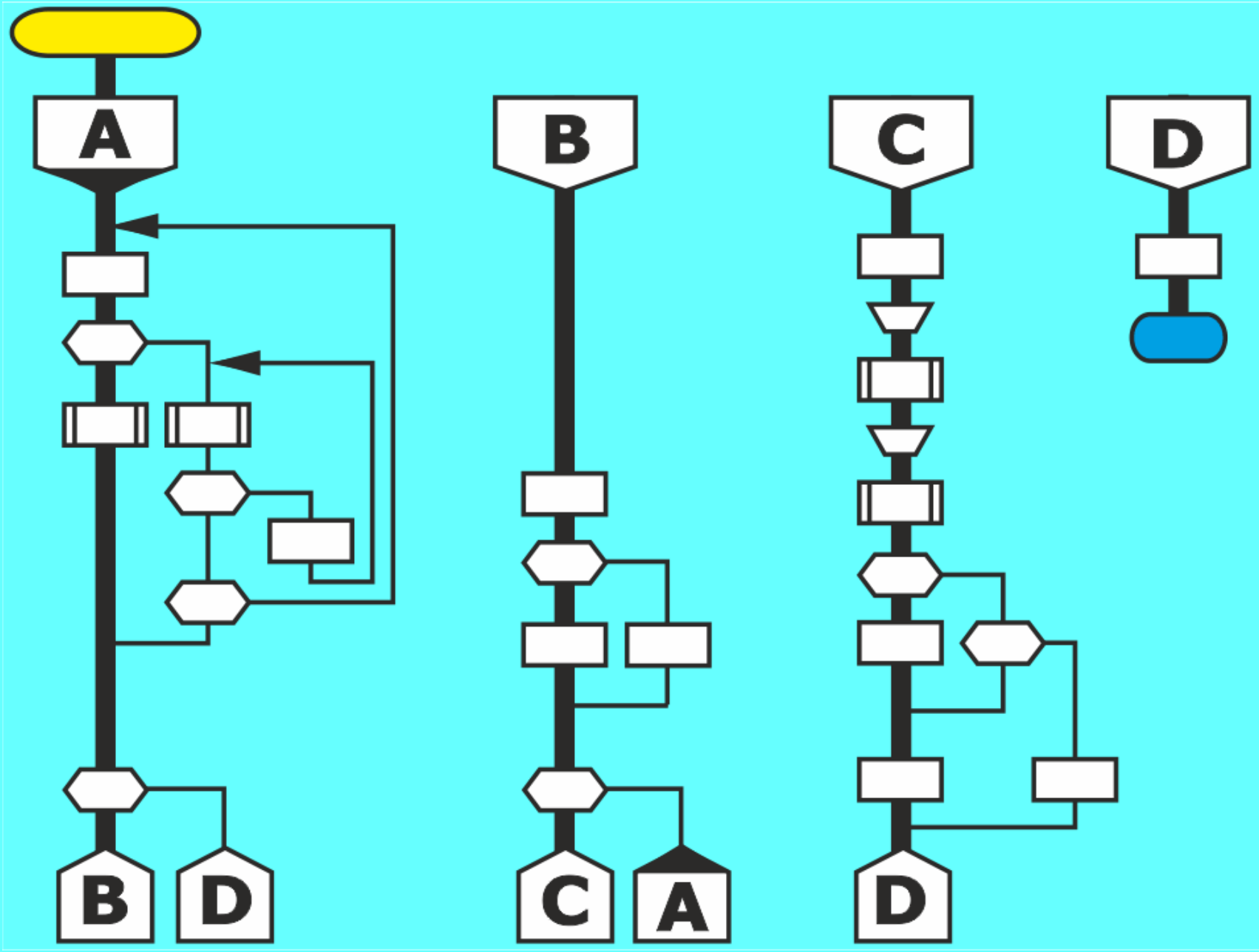
Что такое СИЛУЭТ. *Шаг 2*



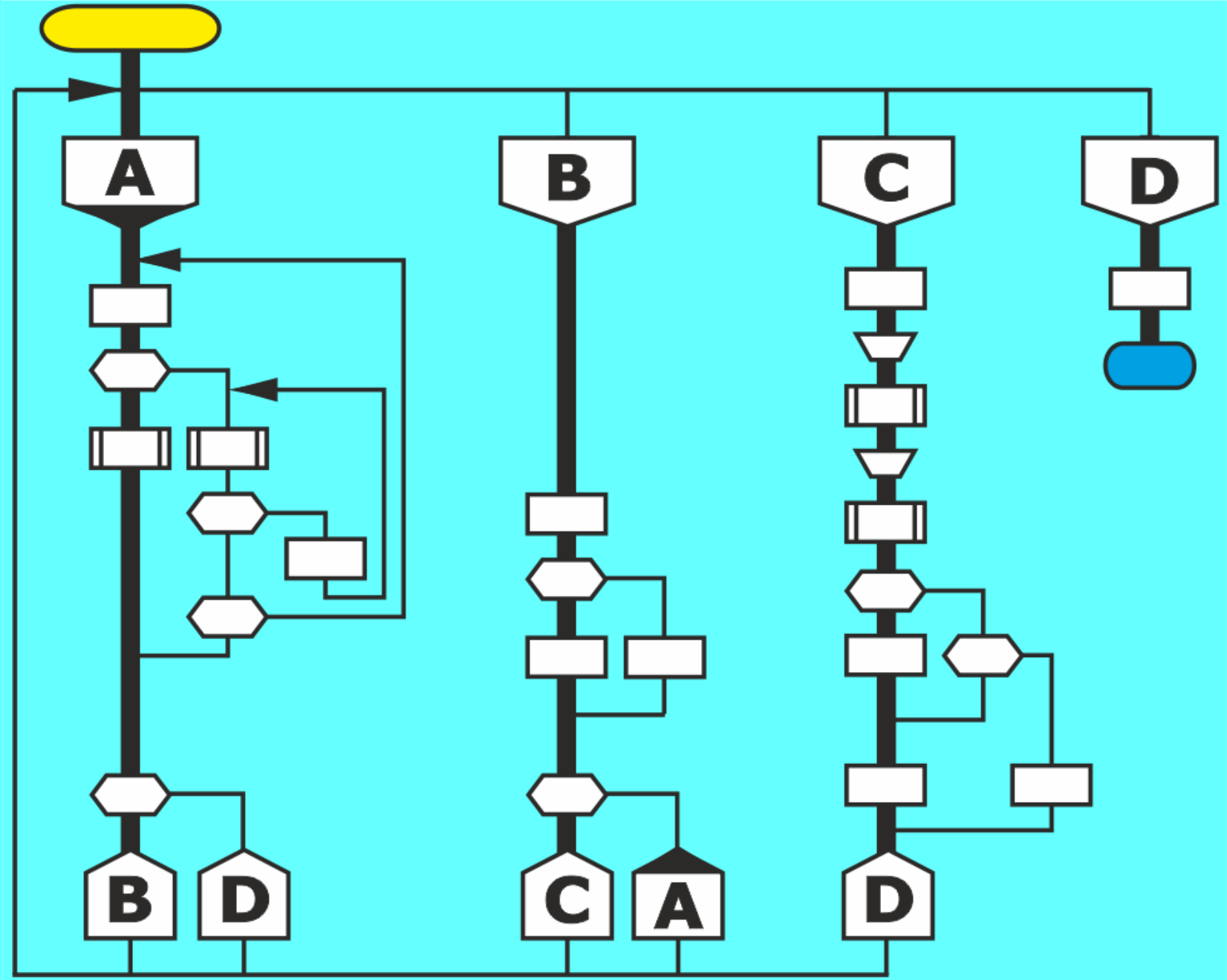
Что такое СИЛУЭТ. *Шаг 3*



Что такое СИЛУЭТ. *Шаг 4*



Что такое СИЛУЭТ. *Шаг 5*



7

Метод Ашкрофта-Манны

преобразования неструктурной программы в структурную

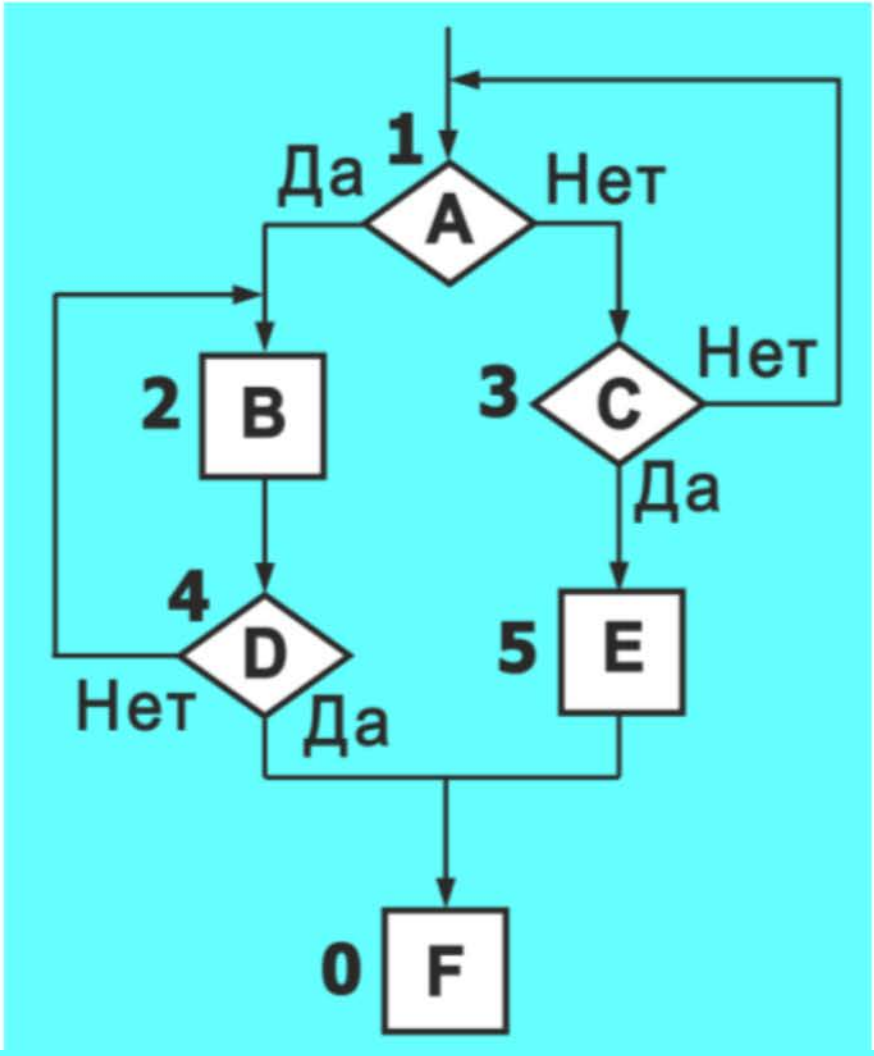
«Хотя метод [Ашкрофта-Манны] представляет теоретический интерес, его никогда не следует использовать на практике».

Гленфорд Майерс

Метод введения переменной состояния

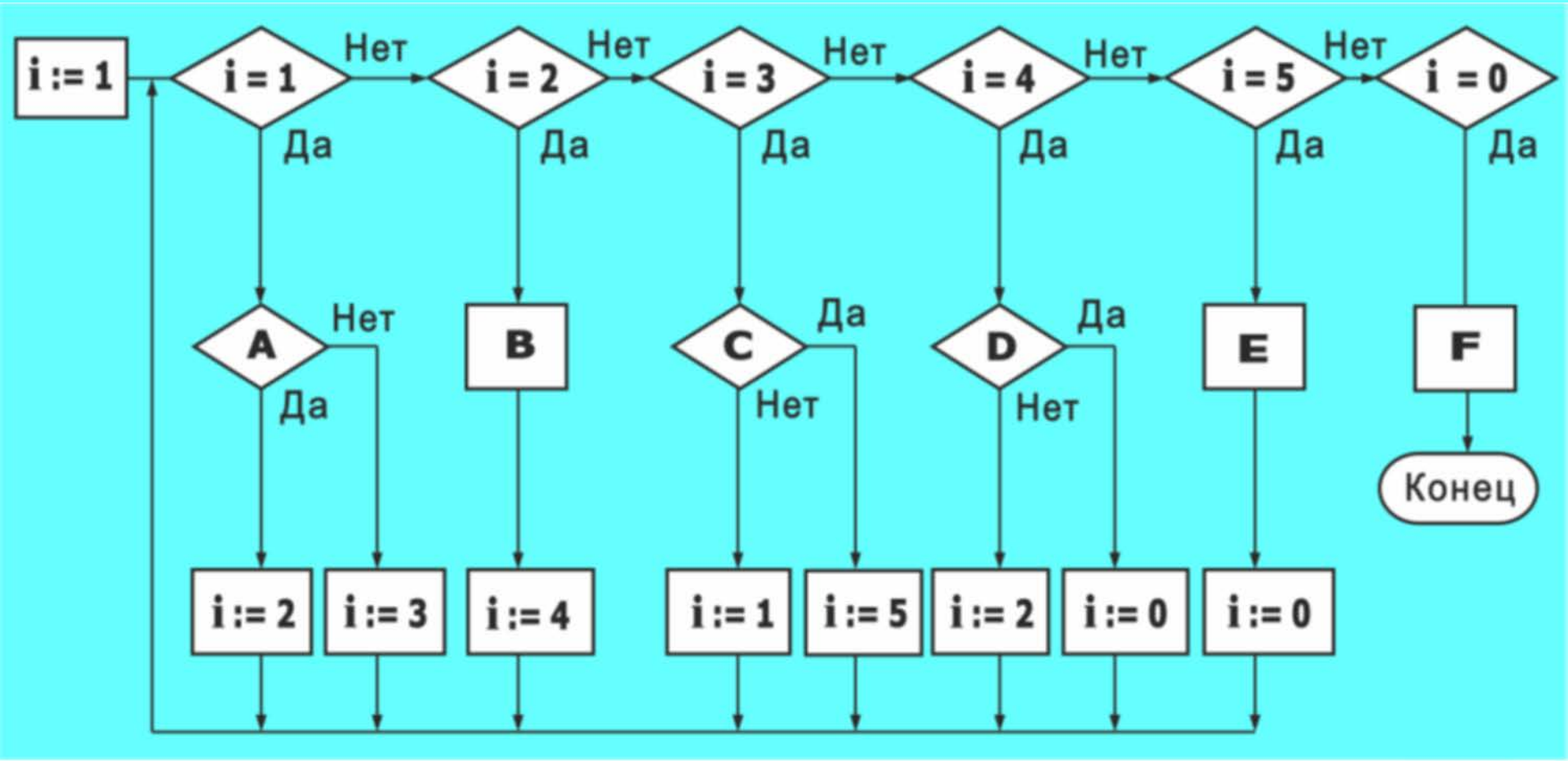
По Эдварду Йодану

Пронумеруем блоки неструктурной программы

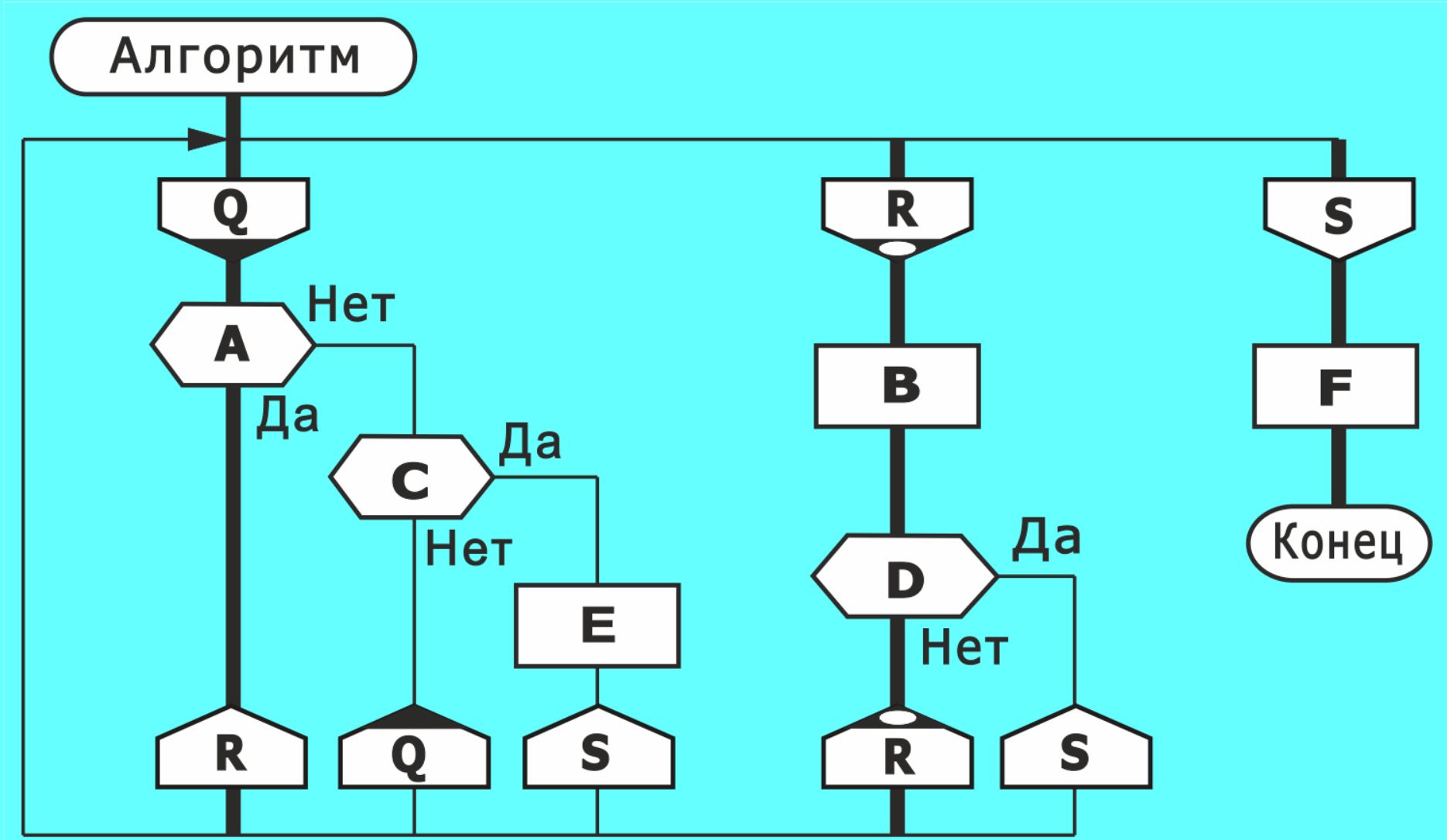


Добавим переменную состояния
i = 1, 2, 3, 4, 5
и служебные блоки.

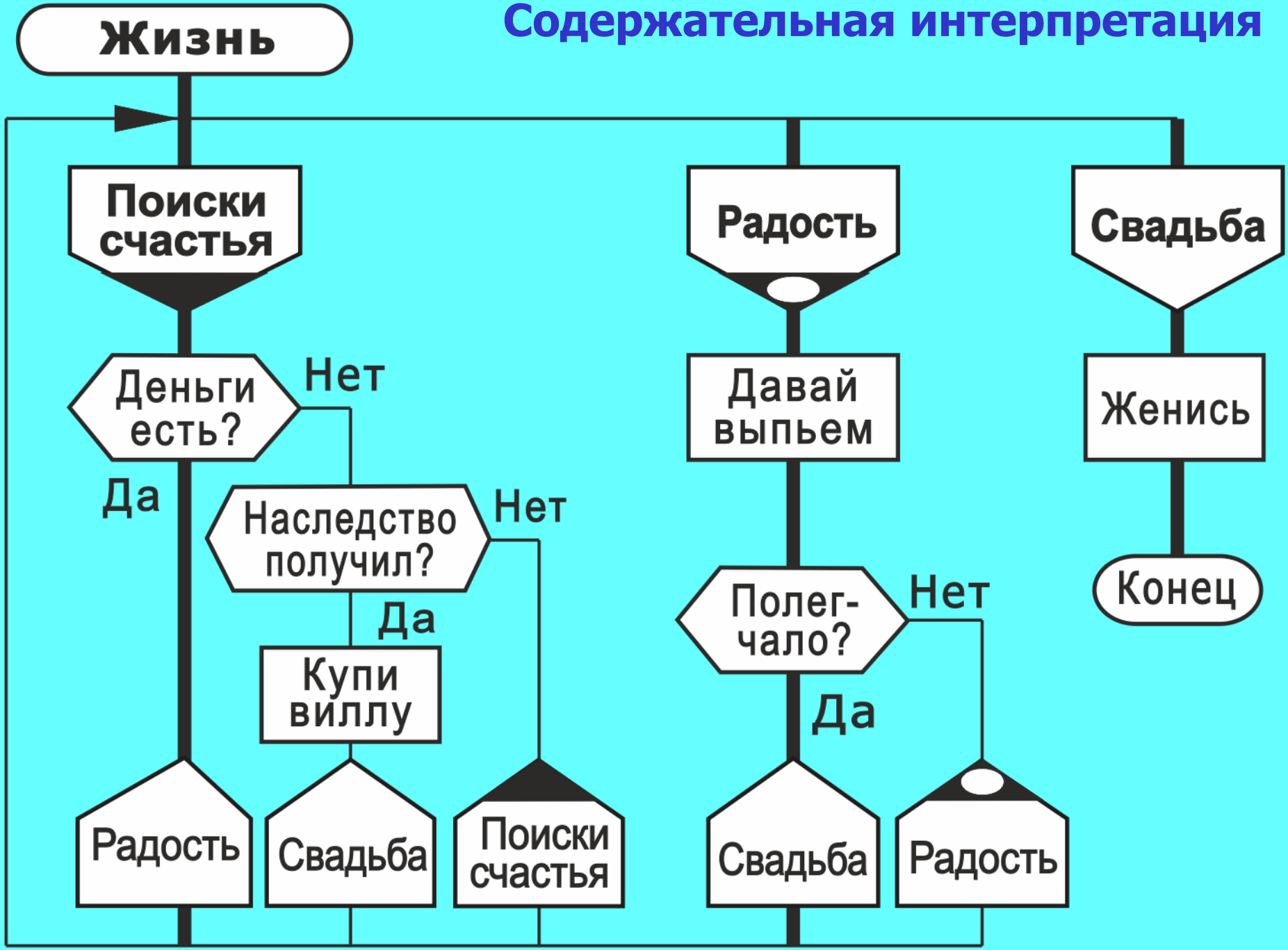
После этого повернем блок-схему на 90°
и зеркально отразим


i := 1
i = 1
Нет
Да
i = 2
Нет
Да
i = 3
Нет
Да
i = 4
Нет
Да
i = 5
Нет
Да
i = 0
Да
A
Нет
Да
B
C
Да
Нет
D
Да
Нет
E
F
Конец
i := 2
i := 3
i := 4
i := 1
i := 5
i := 2
i := 0
i := 0

Структурный силуэт, полностью соответствующий исходной неструктурной программе



Содержательная интерпретация

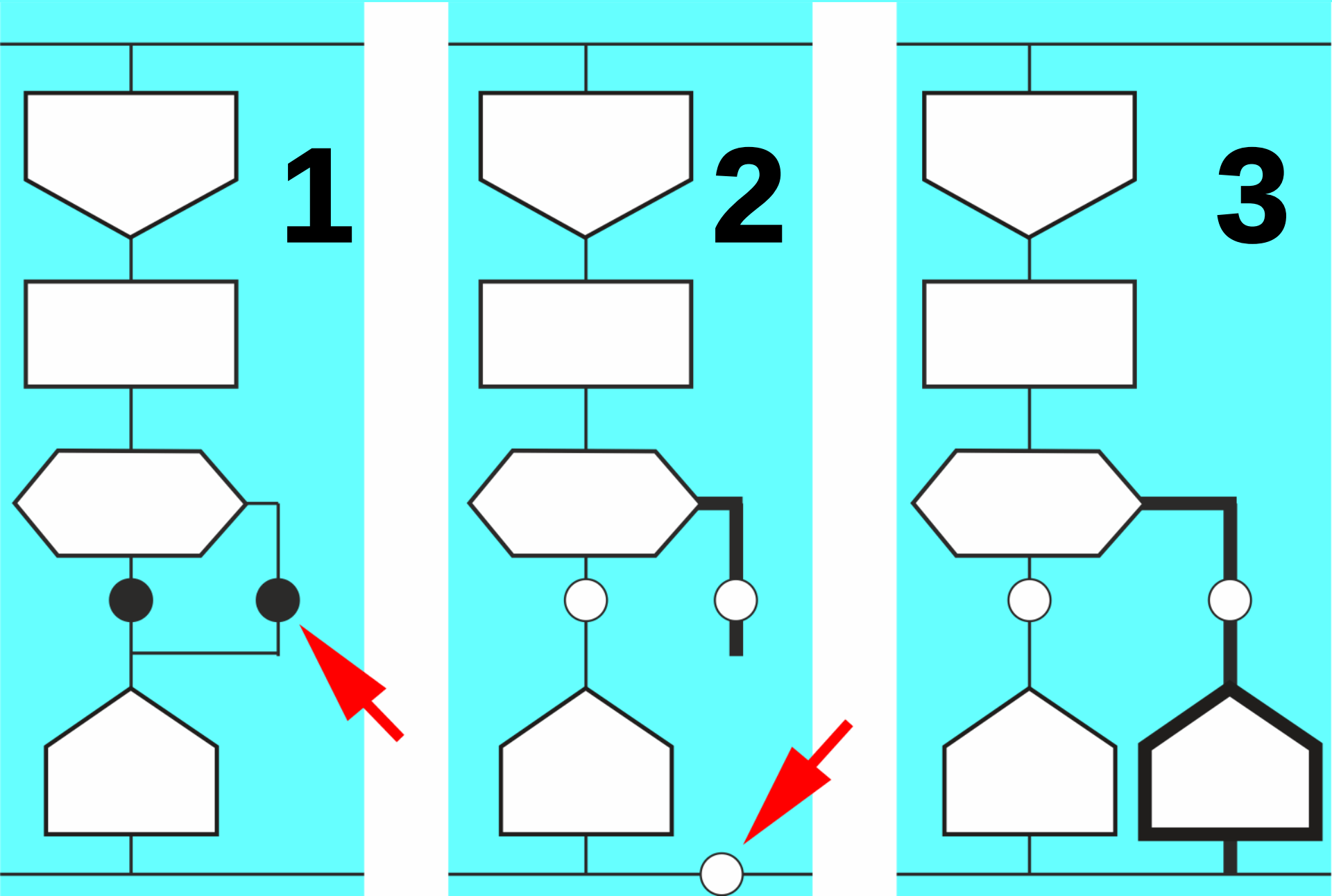


«Язык программирования кодирует объекты предметной области задачи, а наше знание об этих объектах остается за пределами программного текста».

Андрей Петрович Ершов

Операции с лианой

Заземление лианы



8

Двумерное структурное программирование

КРИТИКА

«Мое мнение, что Дракон также способствует катастрофе в ИТ. Так как уже около 50 лет назад было открыто и изучено структурное программирование, а в Драконе это не нашло отражение до сих пор (или нашло отражение частично, но не так, как оно было открыто 50 лет назад)»

Принцип вертикальной ориентации входов и выходов блок-схемы.

Имея в виду шесть типовых блок-схем *(if-do*, *if-then-else*, *case-of*, *while-do*, *repeat-until и сочленение)*
Дейкстра пишет:

«Общее свойство всех этих блок-схем состоит в том, что у каждой из них один вход вверху и один выход внизу»

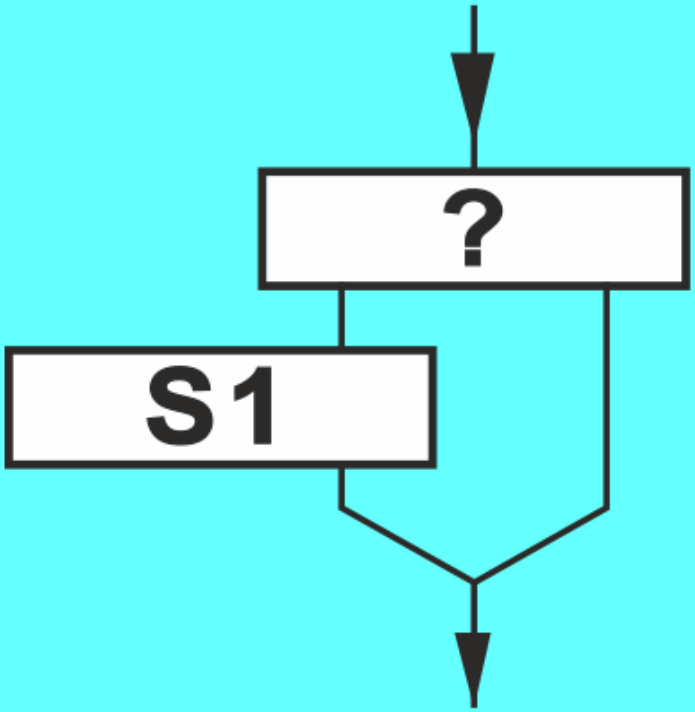
[Структурный подход приводит] «к ограничению топологии блок-схем по сравнению с различными блок-схемами, которые могут быть получены, если разрешить проведение стрелок из любого блока в любой другой блок. Отказавшись от большого разнообразия блок-схем и ограничившись ... тремя типами операторов управления, мы следуем тем самым некоей последовательностной дисциплине».

Эдсгер Дейкстра

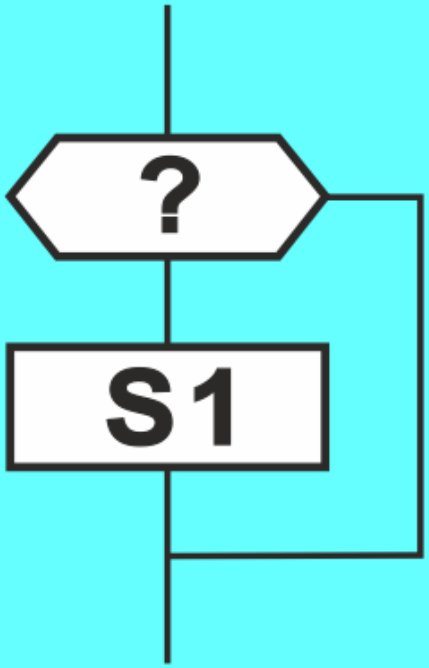
Оператор Дейкстры

if ? do S1

Блок-схема Дейкстры



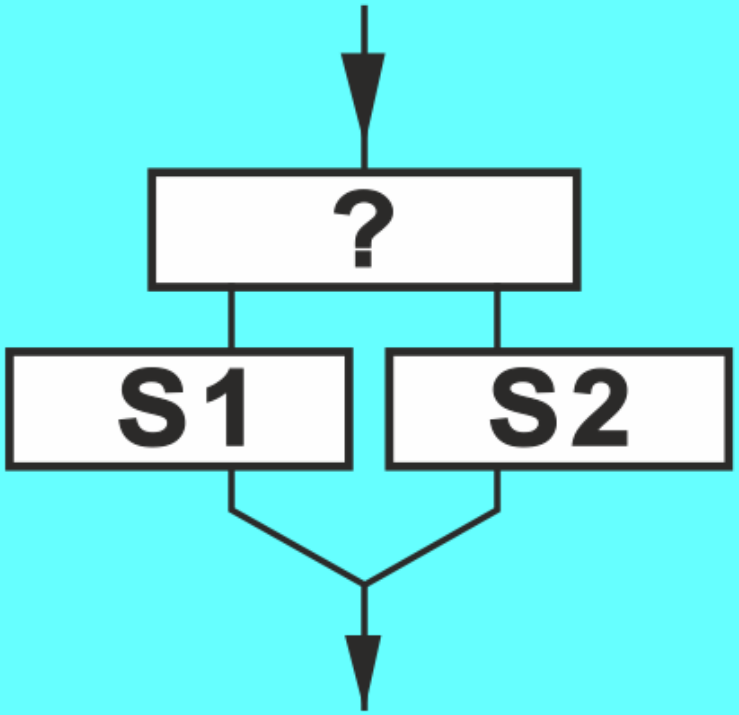
Дракон-схема



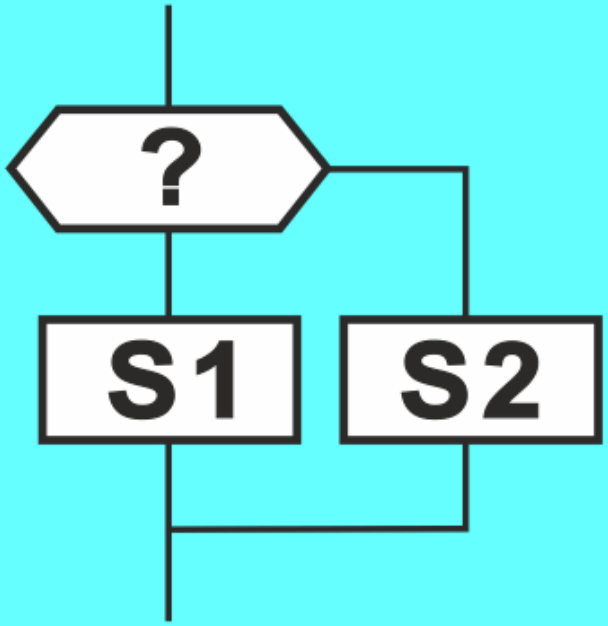
Оператор Дейкстры

if ? then S1 else S2

Блок-схема Дейкстры



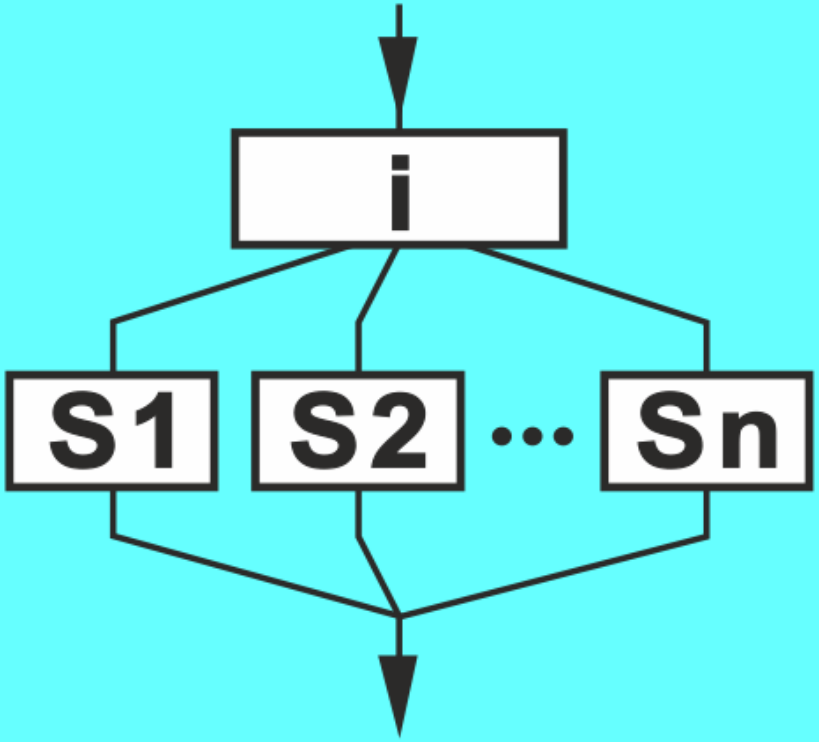
Дракон-схема



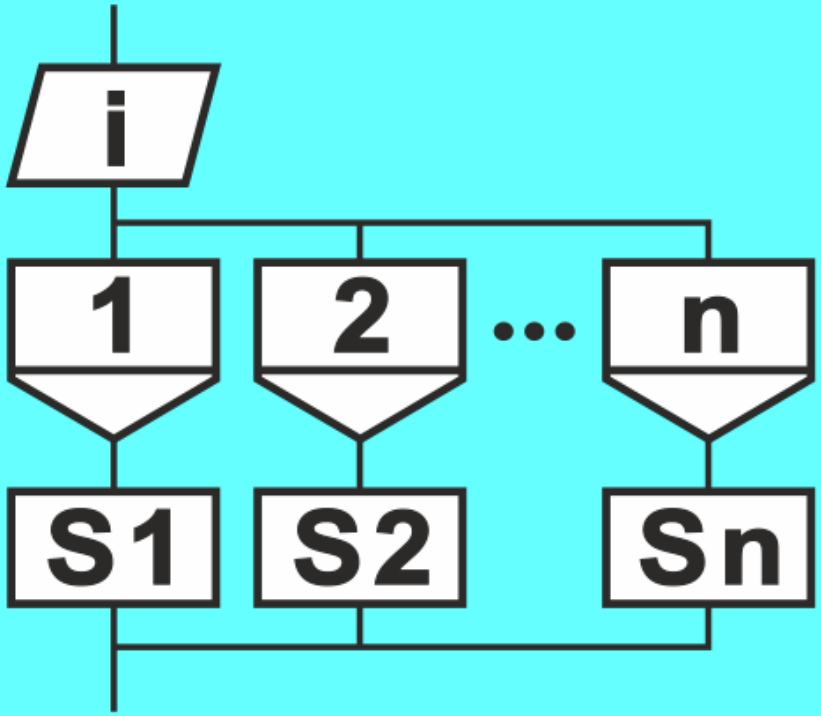
Оператор Дейкстры

case i of (S1; S2; ... Sn)

Блок-схема Дейкстры



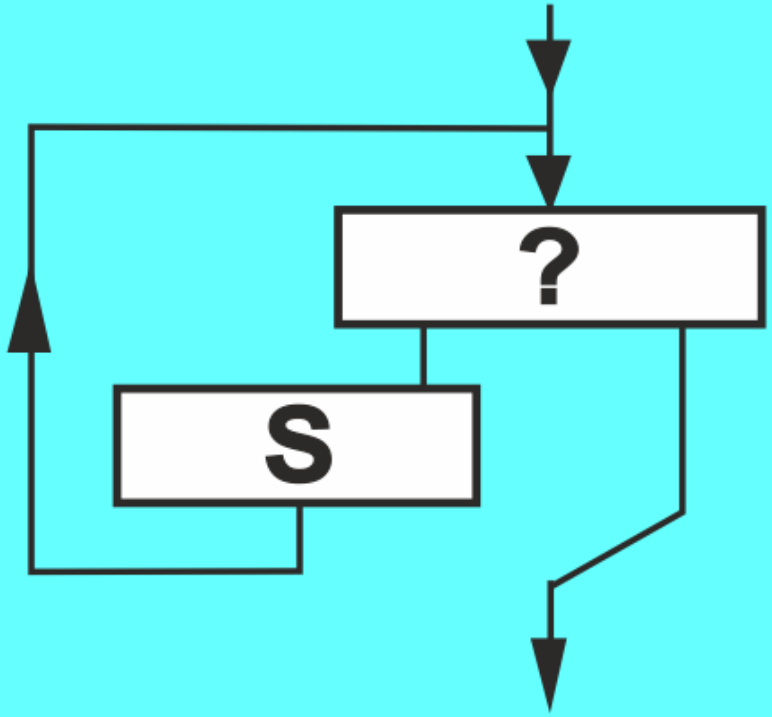
Дракон-схема



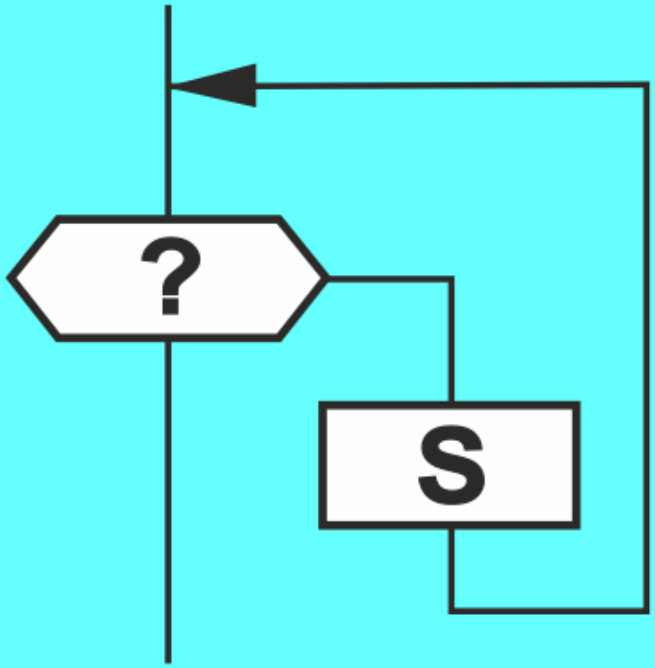
Оператор Дейкстры

while ? do S

Блок-схема Дейкстры



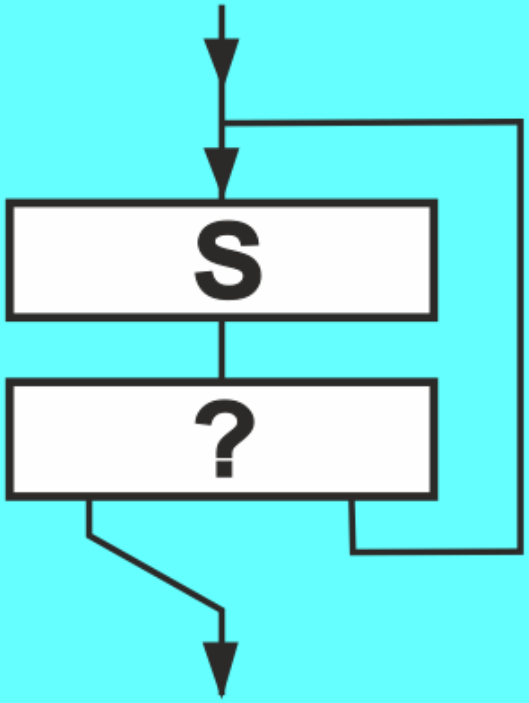
Дракон-схема



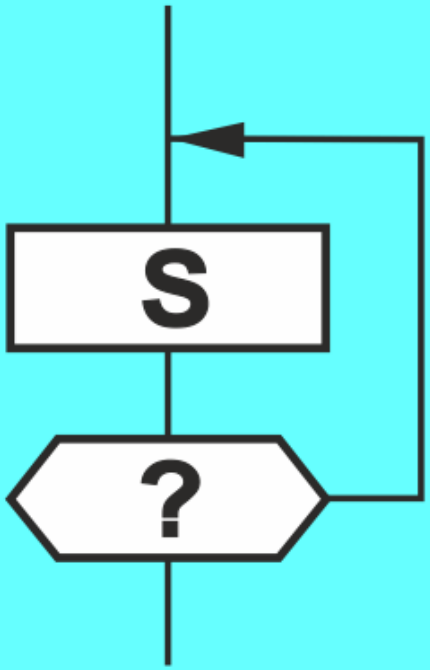
Оператор Дейкстры

repeat S until ?

Блок-схема Дейкстры



Дракон-схема



Семейство ДРАКОН-языков

язык Дракон + язык Си	= гибридный язык Дракон-Си
язык Дракон + язык Java	= гибридный язык Дракон-Java
язык Дракон + язык Си#	= гибридный язык Дракон-Си#
язык Дракон + язык Питон	= гибридный язык Дракон-Питон
язык Дракон + язык Tcl	= гибридный язык Дракон-Tcl
язык Дракон + язык Javascript	= гибридный язык Дракон-Javascript
язык Дракон + язык Lua	= гибридный язык Дракон-Lua
язык Дракон + язык Ада	= гибридный язык Дракон-Ада
язык Дракон + язык Erlang	= гибридный язык Дракон-Erlang
язык Дракон + язык Оберон	= гибридный язык Дракон-Оберон
язык Дракон + язык Ассемблер	= гибридный язык Дракон-Ассемблер

и т. д.

Пуски ракет-носителей и разгонных блоков, при создании которых использовался язык ДРАКОН, производятся или производились с шести космодромов мира:

- **Плесецк**
- **Байконур**
- **Kourou (Французская Гвиана)**
- **Плавучий космодром «Морской старт»**
- **Naro (Южная Корея)**
- **Восточный**

В 1996 году Государственный комитет Российской Федерации по высшему образованию включил изучение языка ДРАКОН в программу курса «Информатика»

Примерная программа дисциплины «Информатика». Издание официальное. — М.: Госкомвуз, 1996. — 21 с.

http://drakon.su/_media/biblioteka/progr_drakon.pdf

Журавлев Юрий Иванович

Тезисы для обсуждения

(Перспективы применения языка ДРАКОН)

1. **Принять язык ДРАКОН как стандарт для медицинских алгоритмов в России**
2. **Принять язык ДРАКОН как стандарт для Вооруженных сил России**
3. **«Скрестить» язык ДРАКОН и 1С**
4. **Заменить стандарт на блок-схемы ГОСТ 19.701-90 и ISO 5807-85 на стандарт языка ДРАКОН**
5. **Ввести изучение языка ДРАКОН в систему образования России**

Пароджанов

vdp2007@bk.ru

Тел. 8-916-111-91-57

Сайт http://drakon.su