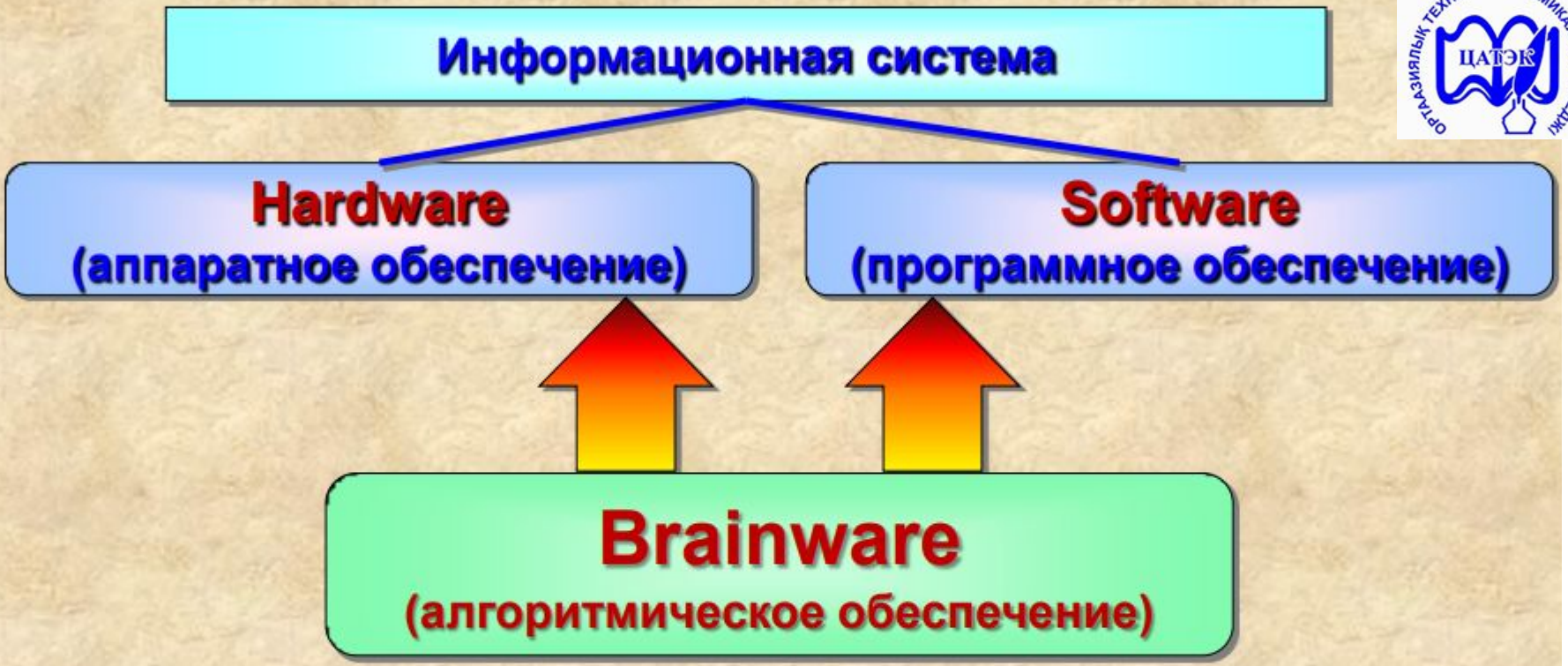
Структуры данных и алгоритмы

Тема: Элементы теории алгоритмов

Преподаватель Аршидинова М. Т.

План лекции

- Понятие и свойства алгоритма
- Виды алгоритмов и их реализация
- Базовые канонические структуры алгоритмов
- Полное построение алгоритма
- Принципы создания эффективных алгоритмов
- Выводы по теме
- Рекомендуемая литература
- Учебно-методическое обеспечение для самостоятельной работы



Разработка алгоритмов предшествует процессу практического программирования

Различают

язык алгоритмический и язык программирования

Понятие и свойства алгоритма

В IX веке аль-Хорезми формализовал 4 действия десятичной арифметики

Алгоризм, алгорисмус (1747 г.), алгорифм (1950 г. – Марков), алгоритм нестрого определяются как конечная совокупность точно сформулированных правил для решения некоторого класса задач

С 30-х годов XX века – Гильберт, Гёдель, Чёрч, Клини, Пост, Тьюринг, Марков

Алгоритм – система формальных правил, четко и однозначно определяющая процесс решения поставленной задачи в виде конечной последовательности действий или операций

Виды алгоритмов и их реализация

Классификация способов представления алгоритмов

Словесный – содержание этапов вычислений задается на естественном языке в произвольной форме с требуемой детализацией

Задан массив чисел. Требуется проверить, все ли числа принадлежат заданному отрезку, который задается границами *A* и *B*

п. 1 Выбираем первый элемент массива - к **п. 2**.
п. 2 Выбранный элемент массива принадлежит интервалу?
Если «да», то к **п. 3**, если «нет» – к **п. 6**.
п. 3 Все элементы массива просмотрены?
Если «да», то к **п. 5**, если «нет», то к **п. 4**.
п. 4 Выбираем следующий элемент - к **п. 2**.
п. 5 Печать сообщения: все элементы принадлежат интервалу - к **п. 7**.
п.6 Печать сообщения: не все элементы принадлежат интервалу - к **п. 7**.
п.7 Конец

Отсутствует наглядность вычислительного процесса, нет достаточной формализации

Формульно-словесный – задание инструкций с использованием математических символов и выражений в сочетании со словесными пояснениями

Вычислить площадь треугольника по трем сторонам (*a*, *b*, *c*)

п. 1 – вычислить полупериметр треугольника p = (a + b + c) / 2

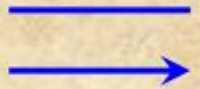
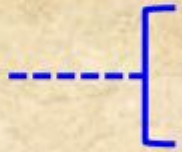
п. 2 – вычислить $S = \sqrt{p(p-a)(p-b)(p-c)}$

п. 3 – напечатать результат S и прекратить вычисления

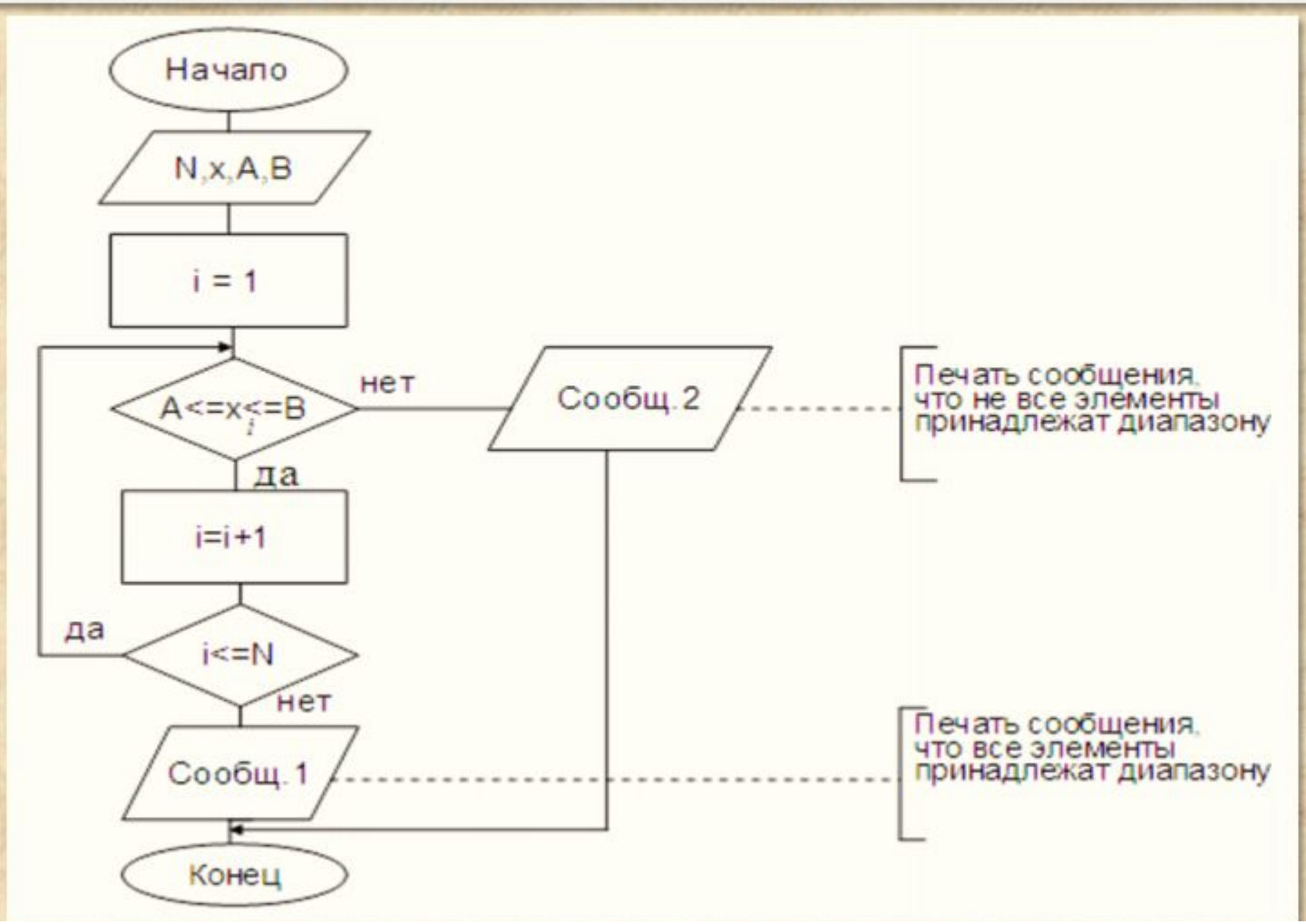
Может быть достигнута любая степень детализации, более наглядно, но не строго формализовано

Блок-схемный – графическое изображение алгоритма, в котором каждый этап процесса обработки данных представляется в виде геометрических фигур (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций

Процесс		Действие (последовательность действий)
Решение		Проверка условия
Модификация (цикл)		Модификация команды или группы команд с целью воздействия на последующую функцию
Предопределенный процесс (модуль)		Процесс, состоящий из одной или нескольких операций (шагов) подпрограммы или модуля
Ввод-вывод		Ввод и вывод информации, при котором не надо описывать носитель данных

Пуск-останов		Начало или конец алгоритма, вход (выход) подпрограммы
Линии потока данных и управления		Отображает поток данных и управления
Соединитель		Используется для обрыва линии и продолжения ее в другом месте блок-схемы
Предопределенный процесс (модуль)		Применяется для добавления комментариев или пояснительных записей

Задан массив чисел.
Требуется проверить, все ли числа принадлежат заданному отрезку, который задается границами *A* и *B*



Псевдокод – совокупность операторов языка программирования и естественного языка

Задан массив чисел. Требуется проверить, все ли числа принадлежат заданному отрезку, который задается границами *A* и *B*

Выбираем первый элемент (*i* = 1).
IF $A > x_i$ или $x_i > B$ **THEN** печать сообщения и выход на конец
ELSE переход к следующему элементу.
IF массив не кончился **THEN** переход на проверку интервала
ELSE печать сообщения, что все элементы входят в интервал.
Конец

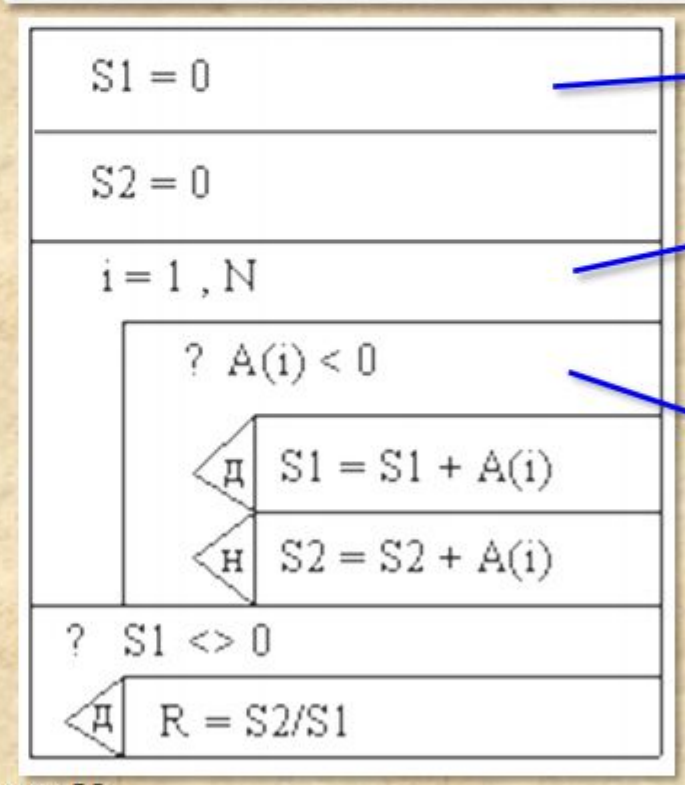
Вычислить

$$Y = \begin{cases} x^2, \text{если } x < 0; \\ x + 1, \text{если } x \geq 0 \end{cases}$$

* Ввод (*x*)
* * Если *x* < 0,то
* * * *y*: = *x***x*
* * иначе
* * * *y*: = *x* +1
* * конец проверки условия
* вывод (*y*)
* конец алгоритма

Структурные диаграммы – могут использоваться в качестве структурных блок-схем, для показа межмодульных связей, для отображения структур данных и систем обработки данных
(диаграммы Насси–Шнейдермана, Варнье, Джексона, МЭСИД и др.)

Задан одномерный массив из положительных и отрицательных чисел. Требуется определить частное от деления суммы положительных элементов на сумму отрицательных элементов этого массива



Следование

Повторение (цикл)

Развилка (решение)

```
. . . S1:=0;
    S2:=0;
FOR I :=1 TO N DO
    IF A[I] < 0 THEN
            S1:=S1 + A[I]
        ELSE
            S2:=S2 +A[I] ;
IF S1 <> 0 THEN
            R:= S2/S1; . . .
```

Базовые канонические структуры алгоритмов

Базовые структуры:

- **следование или последовательность операторов**
- **ветвление или условный оператор**
- **повторение или оператор цикла**

Программа, составленная из канонических структур, называется регулярной программой, т.е. имеющей один вход и один выход

Блоки, связанные с обработкой данных

Простые

Имеют один вход и один выход
Простое действие не означает, что оно единственное – это может быть последовательность действий

Условные

Обладают двумя выходами в зависимости от того, истинным ли окажется условие

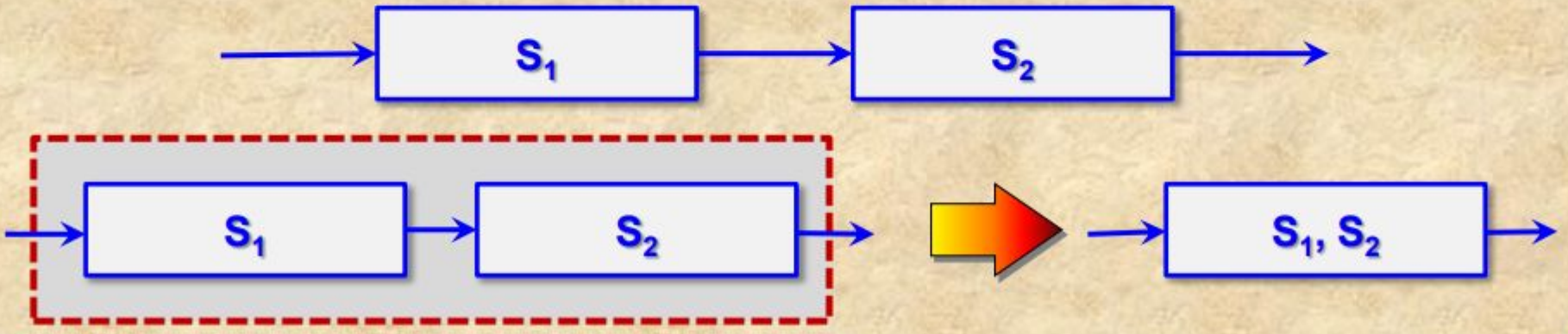
Часть алгоритма, организованная как **простое действие**, т. е. имеющая **один вход** (выполнение начинается всегда с одного и того же действия) и **один выход** (после завершения данного блока всегда начинает выполняться одно и то же действие), называется **функциональным блоком**

Поток управления может обладать следующими свойствами:

- **каждый блок выполняется не более одного раза**
- **выполняется каждый блок**

1 *Линейный поток управления*

Поток управления, в котором выполняются оба свойства, называется линейным – в нем несколько функциональных блоков выполняются последовательно



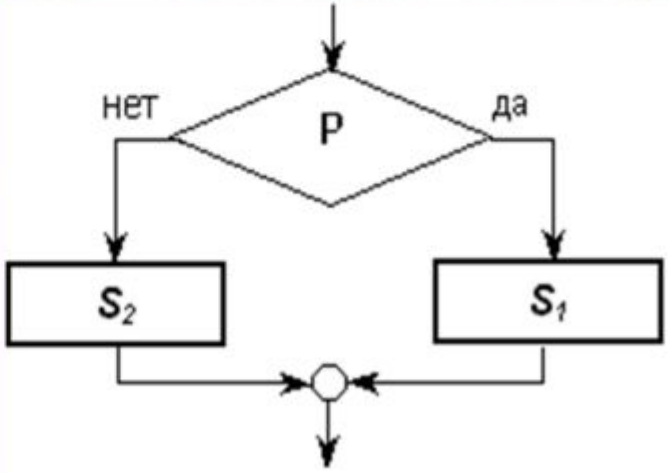
2 Ветвление

Ветвление организует выполнение одного из двух функциональных блоков **в зависимости от значения проверяемого логического условия**

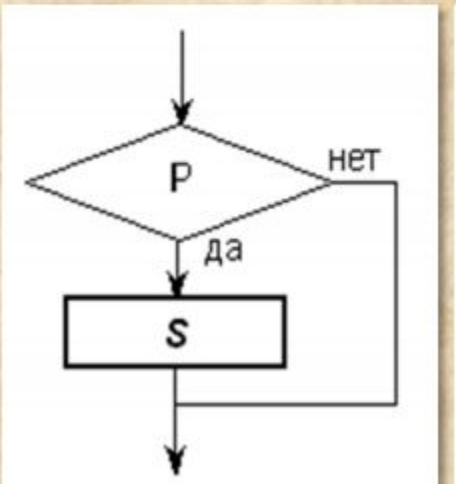
Проверка P выполняется **предикатом** – функцией, задающей логическое выражение или условие, значением которого может быть **истина** или **ложь**

Переключатель организует выбор одного варианта **из множества возможных**

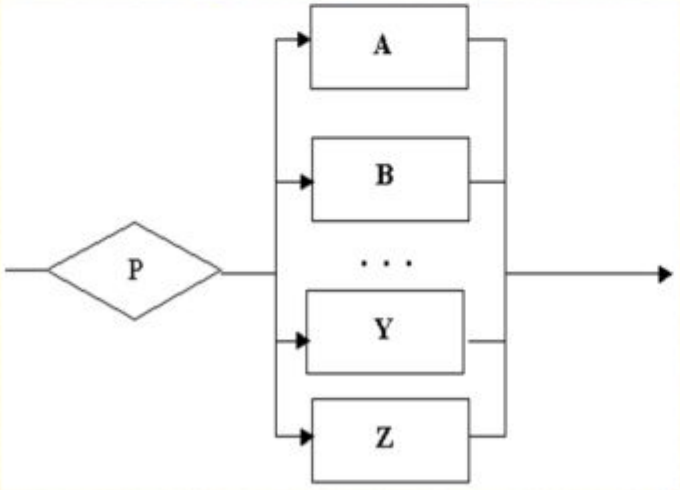
Полное ветвление



Неполное ветвление



Переключатель

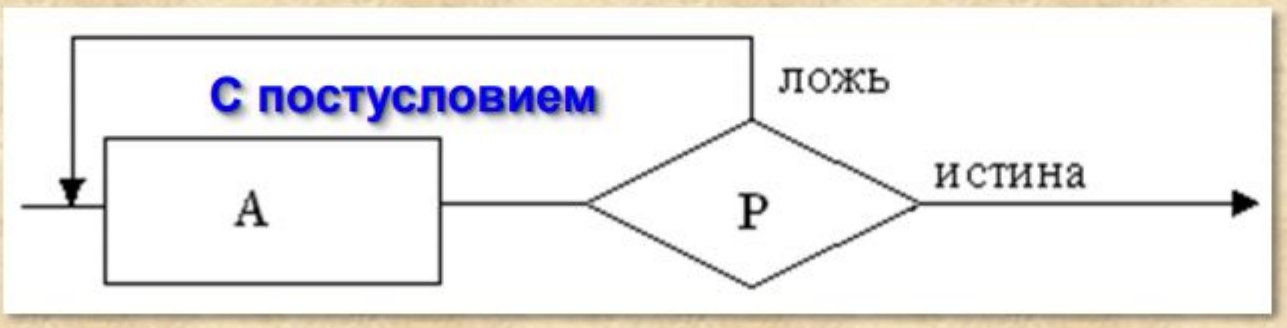


3 Повторение

Повторение – **многократное** выполнение фрагментов алгоритма (программы).

Такой тип потока управления называется **циклическим** – он организует многократное повторение функционального блока, **пока логическое условие его выполнения является истинным.**

Действие *A* будет повторяться, пока значение предиката *P* будет оставаться истинным



Действие A будет выполняться хотя бы один раз

Алгоритм называется структурным, если он представляет собой комбинацию трех рассмотренных выше структур

Преимущества структурных алгоритмов

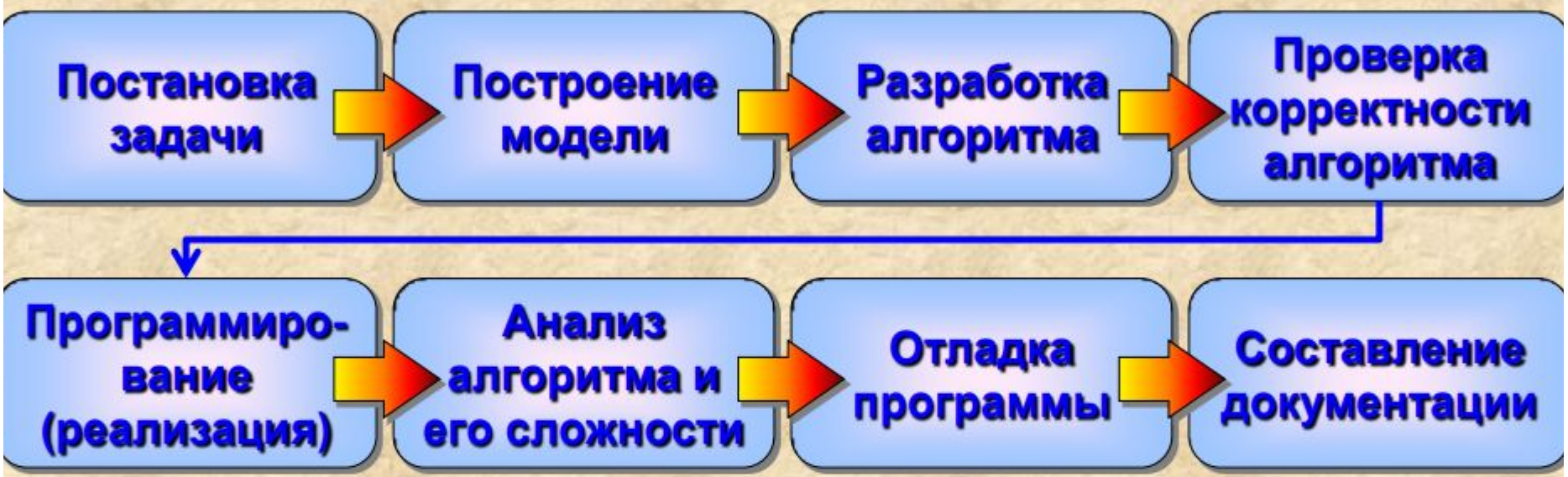
- **Понятность и простота восприятия алгоритма** (поскольку невелико число исходных структур, которыми он образован)
- **Проверяемость** (для проверки любой из основных структур достаточно убедиться в правильности входящих в нее функциональных блоков)
- **Модифицируемость** (состоит в простоте изменения структуры алгоритма, поскольку составляющие блоки относительно независимы)

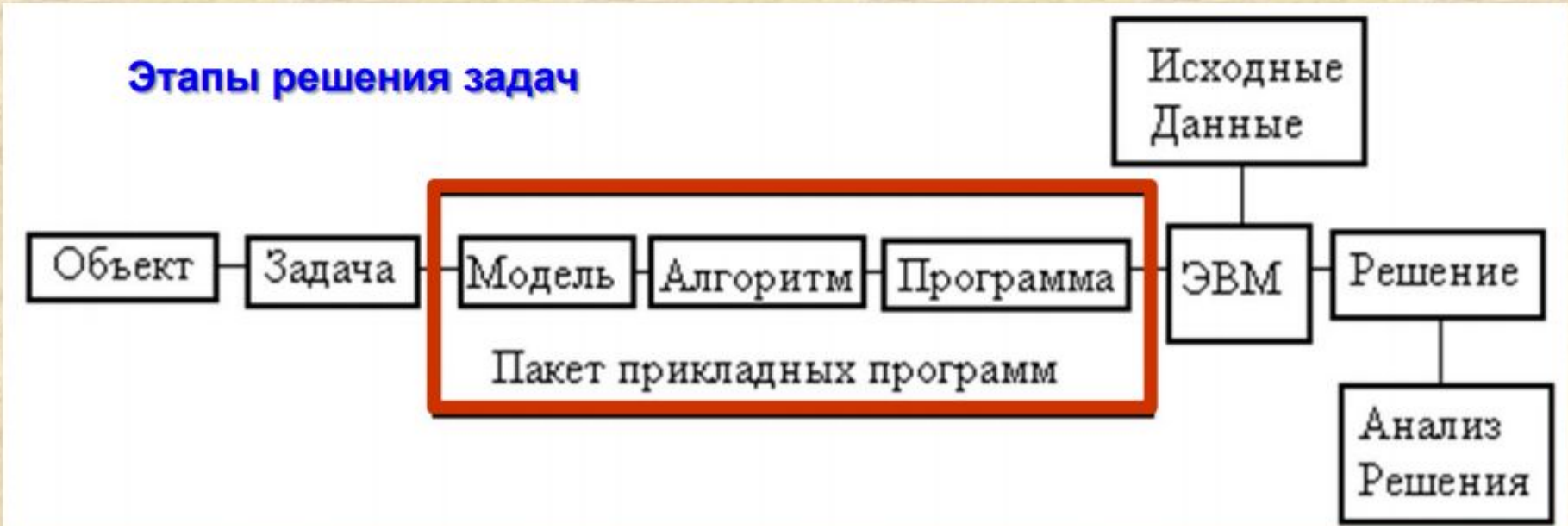
Структурная теорема Бома – Джакопини

Любой алгоритм может быть сведен к структурному

Иными словами: любому неструктурному алгоритму может быть построен эквивалентный ему структурный алгоритм

Полная разработка алгоритма





Постановка задачи

1. Понятна ли терминология, используемая в предварительной формулировке?
2. Что дано?
3. Что нужно найти?
4. Как определить решение?
5. Каких данных не хватает, или, наоборот, все ли перечисленные в формулировке задачи данные используются?
6. Какие сделаны допущения?

Построение модели

1. Какие математические структуры больше всего подходят для задачи?
2. Существуют ли решенные аналогичные задачи?

На выбор соответствующей структуры будут влиять следующие факторы:

- ограниченность наших знаний относительно небольшим количеством структур
- удобство представления
- простота вычислений
- полезность различных операций, связанных с рассматриваемой структурой или структурами

1. Вся ли важная информация задачи хорошо описана математическими объектами?
2. Существует ли математическая величина, ассоциируемая с искомым результатом?
3. Выявили ли мы какие-нибудь полезные отношения между объектами модели?
4. Можем ли мы работать с моделью? Удобно ли с ней работать?

Разработка алгоритма

Выбор метода разработки алгоритма существенно зависит от выбора модели и может в значительной степени повлиять на эффективность алгоритма решения. Два различных алгоритма могут быть правильными, но сильно отличаться по эффективности

Корректность алгоритма

- **Пусть алгоритм описан в виде последовательности шагов от шага 0 до шага *m***
- **Может потребоваться формулировка утверждения об условиях, действующих до и после пройденного шага**
- **Затем постараемся предложить доказательство конечности алгоритма, при этом будут проверены все подходящие входные данные и получены все подходящие выходные данные**

Другой способ проверки корректности

- **Для каждого цикла вручную подсчитываются две контрольные точки**
- **Если контрольные точки совпадают со значениями, выданными программой, можно быть уверенным, что все циклы подобного типа в программе работают правильно**
- **Два контрольных вычисления должны быть сделаны для каждого цикла программы**

Реализация алгоритма (программирование)

При написании программы могут возникать проблемы:

- Очень часто отдельно взятый шаг алгоритма может быть выражен в форме, которую трудно перевести непосредственно в конструкции языка программирования
- Перед разработкой программы необходимо построить целую систему структур данных для представления важных аспектов используемой модели

1. Каковы основные переменные?
2. Каких они типов?
3. Сколько нужно массивов и какой размерности?
4. Имеет ли смысл пользоваться связными списками?
5. Какие нужны подпрограммы (возможно, уже записанные в памяти)?
6. Каким языком программирования пользоваться?

Необходимо **тщательно следить**, чтобы процесс преобразования правильного алгоритма (в словесной форме в или форме схемы алгоритма) в программу, написанную на алгоритмическом языке, **«заслуживал доверия»**

Принципы создания эффективных алгоритмов

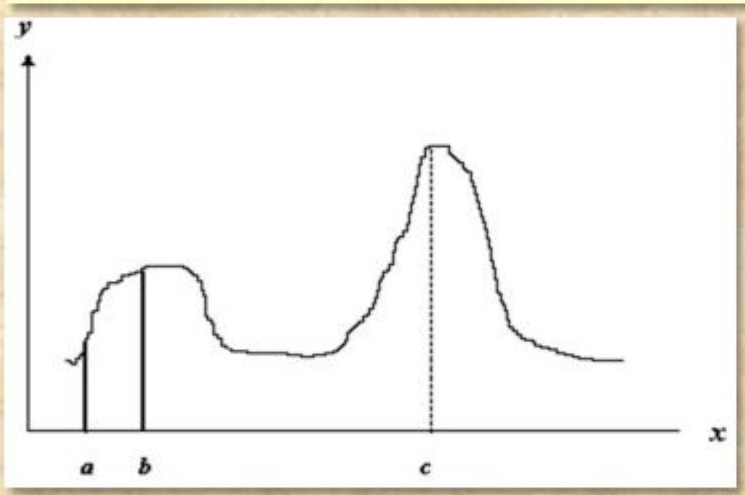
1 Метод частных целей

Сущность метода – сведение трудной задачи к последовательности более простых задач

1. Можем ли мы решить часть задачи? Можно ли, игнорируя некоторые условия, решить оставшуюся часть задачи?
2. Можем ли мы решить задачу для частных случаев? Можно ли разработать алгоритм, который дает решение, удовлетворяющее всем условиям задачи, но входные данные которого ограничены некоторым подмножеством всех входных данных?
3. Есть ли что-то, относящееся к задаче, что мы не достаточно хорошо поняли? Если попытаться глубже вникнуть в некоторые особенности задачи, сможем ли мы что-то узнать, что поможет нам подойти к решению?
4. Встречались ли мы с похожей задачей, решение которой известно? Можно ли видоизменить ее решение для решения нашей задачи? Возможно ли, что эта задача эквивалентна известной нерешенной задаче?

2 Метод подъема

Сущность метода – начинается с принятия начального предположения или вычисления начального решения задачи. Затем начинается (насколько возможно) быстрое движение «вверх» от начального решения по направлению к лучшим решениям. Когда алгоритм достигнет такой точки, из которой больше невозможно двигаться наверх, алгоритм останавливается



Невозможно гарантировать, что окончательное решение всегда будет оптимальным

Эта ситуация часто ограничивает применение метода подъема

3 Отрабатывание назад

Сущность метода – работа начинается с цели или решения задачи и далее осуществляется движение к начальной постановке задачи. Затем, если эти действия обратимы, производится движение

Выводы по теме

1. **Алгоритмическое обеспечение** играет **важную роль** в процессе разработки и создания информационных систем

2. **Алгоритмы** обладают следующими **свойствами**:
   - конечность
   - детерминированность
   - результативность
   - массовость
   - эффективность

3. **Алгоритмы** бывают следующих **типов**:
   - линейный
   - разветвляющиеся
   - циклические

4. В зависимости от степени детализации, поставленных целей, методов и технических средств решения задачи используются **различные формы представления** алгоритмов

5. **Структурные алгоритмы** обладают **преимуществами**:
   - понятность
   - проверяемость
   - модифицируемость

Вопросы

1. Какими свойствами должен обладать алгоритм?
2. Какие типы алгоритмов используются в программировании?
3. Какие формы представления алгоритмов применяются на практике?
4. Какие базовые структуры применяются для алгоритмизации?
5. Какой поток управления называется линейным?
6. Какие этапы предусматривает полная разработка алгоритма?
7. Какие принципы лежат в основе создания эффективных алгоритмов?
8. Какой алгоритм называется структурным?