Структуры и алгоритмы обработки данных

Лекция 1

Введение в теорию алгоритмов
Блок-схема алгоритма

Структуры и алгоритмы обработки данных

Цель курса:
рассмотреть

- ***основные понятия об алгоритме в программах и алгоритмизации решения задач***
- ***основные понятия о данных к алгоритмам, их базовые типы и структуры, вопросы их использования в алгоритмизации задач***

Структуры и алгоритмы обработки данных

Литература

1. Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989. – 360 с.
2. Вирт Н. Алгоритмы + структуры данных = программы / Пер. с англ.- М.: Мир, 1985.
3. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы. Пер. с англ.-М. : Издательский дом «Вильямс», 2001
4. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.. Алгоритмы. Построение и анализ. 2-е издание. М: Изд. Дом «Вильямс», 2007
5. Д. Кнут. Искусство программирования для ЭВМ. Т. 1-3, М.: Мир, 1978, 1995 и др..
6. Сибуя М., Ямамото Т. Алгоритмы обработки данных. - М: Мир, 1986 -218с.
7. Лэгсам Й, Огенстайн М. Структуры данных для персональных ЭВМ - М: Мир, 1989 -586с.
8. Ключарев А. А., Матьяш В. А., Щекин С. В. Структуры и алгоритмы обработки данных: Учеб. пособие/ СПбГУАП. СПб., 2003. 172 с.: ил

Понятие алгоритма

Историческая справка

Содержательные явления, приведшие к возникновению понятия «алгоритм», прослеживаются в математике в течении всего его времени существования

- **алгоритм Евклида нахождения наибольшего общего кратного натуральных чисел, найденный еще в III веке до нашей эры и доживший до наших дней**
- **в XV веке был известен алгоритм, разработанный самаркандским астрономом Аль-Каши, вычисления числа π, которое он вычислил с 17 верными значащими цифрами после запятой**
- **……..**

Понятие алгоритма

Историческая справка

Первоначально понятие алгоритма -

- *словесные правила, схемы, формулы, использовались для описания вычислительного процесса*
- *не точное математическое определение, а лишь объяснение смысл слова «алгоритм»*

С алгоритмами, т.е. эффективными процедурами, однозначно приводящими к результату, математики имели дело всегда

- *умножения «столбиком»*
- *деления «углом»*
- *метод исключений неизвестных при решении систем линейных уравнений*

Понятие алгоритма

Историческая справка

20-х - 30-х гг XX века:

- **вопросы обоснования математики**
- **развитие вычислительной математики**
- **развитие вычислительной техники**

Необходимо уточнить понятия алгоритм как объекта математической теории

Появился раздел дискретной математики называемый теорией алгоритмов

Основоположники теории алгоритмов – великие математики XX века А.И. Колмогоров, А.А. Марков, А.П. Ершов, А.И. Мальцев, В.А. Успенский, А.М. Тьюринг, К. Гёдель, А. Чёрчь, А. Туэ, Э.Л. Пост и др.

Этапы решения задач на ЭВМ

Этапы

- *постановка задачи*
- *формализация*
- *создание технического задания на исходную задачу*
- *разработка алгоритма решения задачи*
- *кодирование*
- *тестирование*
- *отладка и документирование программы*
- *получение решения исходной задачи путем выполнения законченной программы*

Схема процесса создания программ для решения прикладных задач

Математическая модель

- Неформальный алгоритм

Абстрактное описание данных

- Формальный алгоритм

Описание данных на языке программирования

- Программа на языке программирования

- *Создание программы можно рассматривать как процесс последовательного преобразования информации от первоначальной неформальной постановки задачи, до получения завершенной программы на языке программирования*
- *Это преобразование затрагивает как описания информационных объектов задачи (данные) так и описания действий над этими объектами (алгоритмы)*

Определение алгоритма

Алгоритм является базовым основополагающим понятием информатики, а **алгоритмизация** (программирование) – основным разделом курса информатики (ядром курса)

Понятие алгоритма, как и понятие информации, точно определить невозможно. Поэтому встречаются самые разнообразные определения – от "наивно-интуитивных" ("алгоритм – это план решения задачи") до "строго формализованных" (нормальные алгоритмы Маркова).

Определение алгоритма

Определения современной математики:

- последовательность действий со строго определенными правилами выполнения;
- предписание, определяющее содержание и последовательность операций, переводящих исходные данные в искомый результат;
- точное описание некоторого вычислительного процесса или любой иной последовательности действий;
- точное и полное предписание о последовательности выполнения конечного числа действий, необходимых для решения любой задачи данного типа

Определение алгоритма

Алгоритм — это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи

А. Колмогоров

Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату

А. Марков

Определение алгоритма

Алгоритм – формально описанная вычислительная процедура, получающая исходные данные (вход алгоритма или аргумент) и выдающая результат вычислений на выход

Томас Х. Кормен и др., Алгоритмы: построение и анализ, 2-е изд.: Пер. с англ.- М. : Издательский дом "Вильямс", 2005

Определение алгоритма

Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством - формальным исполнителем

Задача исполнителя - точная реализация уже имеющегося алгоритма. Формальный исполнитель не обязан вникать в сущность алгоритма, а возможно, и неспособен его понять

Пример формального исполнителя - стиральная машина-автомат

В информатике универсальным исполнителем алгоритмов является компьютер

Виды алгоритмов

В ЗАВИСИМОСТИ ОТ ЦЕЛИ, НАЧАЛЬНЫХ УСЛОВИЙ ЗАДАЧИ, ПУТЕЙ ЕЕ РЕШЕНИЯ, ОПРЕДЕЛЕНИЯ ДЕЙСТВИЙ ИСПОЛНИТЕЛЯ:

- **Вероятностный (стохастический) алгоритм -** дает программу решения задачи несколькими путями или способами, приводящими к вероятному достижению результата

Виды алгоритмов

В ЗАВИСИМОСТИ ОТ ЦЕЛИ, НАЧАЛЬНЫХ УСЛОВИЙ ЗАДАЧИ, ПУТЕЙ ЕЕ РЕШЕНИЯ, ОПРЕДЕЛЕНИЯ ДЕЙСТВИЙ ИСПОЛНИТЕЛЯ:

- **Эвристический алгоритм -**
  достижение конечного результата программы действий однозначно не предопределено, так же как не обозначена вся последовательность действий, не выявлены все действия исполнителя

К эвристическим алгоритмам относят, например, инструкции и предписания. В этих алгоритмах используются универсальные логические процедуры и способы принятия решений, основанные на аналогиях, ассоциациях и прошлом опыте решения схожих задач

Виды алгоритмов

В ЗАВИСИМОСТИ ОТ ЦЕЛИ, НАЧАЛЬНЫХ УСЛОВИЙ ЗАДАЧИ, ПУТЕЙ ЕЕ РЕШЕНИЯ, ОПРЕДЕЛЕНИЯ ДЕЙСТВИЙ ИСПОЛНИТЕЛЯ:

- **Линейный алгоритм -**
  набор команд (указаний), выполняемых последовательно друг за другом
- **Разветвляющийся алгоритм -**
  алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов

Виды алгоритмов

В ЗАВИСИМОСТИ ОТ ЦЕЛИ, НАЧАЛЬНЫХ УСЛОВИЙ ЗАДАЧИ, ПУТЕЙ ЕЕ РЕШЕНИЯ, ОПРЕДЕЛЕНИЯ ДЕЙСТВИЙ ИСПОЛНИТЕЛЯ:

- **Циклический алгоритм -**
  алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными
  К циклическим алгоритмам сводится большинство методов вычислений перебора вариантов
- **Цикл программы — последовательность команд , которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторому условию**

Виды алгоритмов

В ЗАВИСИМОСТИ ОТ ЦЕЛИ, НАЧАЛЬНЫХ УСЛОВИЙ ЗАДАЧИ, ПУТЕЙ ЕЕ РЕШЕНИЯ, ОПРЕДЕЛЕНИЯ ДЕЙСТВИЙ ИСПОЛНИТЕЛЯ:

- **Вспомогательный (подчиненный) алгоритм (процедура) -** алгоритм, ранее разработанный и целиком используемый при алгоритмизации конкретной задачи

Способы задания алгоритма

- **словесный** - запись последовательности действий на естественном языке
- **графический** - с помощью специальных графических символов
- **формульный** - с помощью математических формул, которые определяют порядок вычислений
- **табличный** - в виде таблицы, в которой фиксируются этапы исполнения алгоритма и результаты исполнения

Способы задания алгоритма

- **словесный - запись последовательности действий на естественном языке**

1. Скопировать целое положительное значение *X* во вспомогательную переменную *x*.
2. Скопировать целое положительное значение *Y* во вспомогательную переменную *y*.
3. Если *x*≠*y*, перейти к п. 4, иначе – к п. 7.
4. Если *x*>*y*, перейти к п. 5, иначе – к п. 6.
5. Записать в *x* результат вычисления выражения *x*-*y* и перейти к п. 3.
6. Записать в *y* результат вычисления выражения *y*-*x* и перейти к п. 3.
7. Конец. *x* – результат работы.

Нахождение наибольшего общего делителя

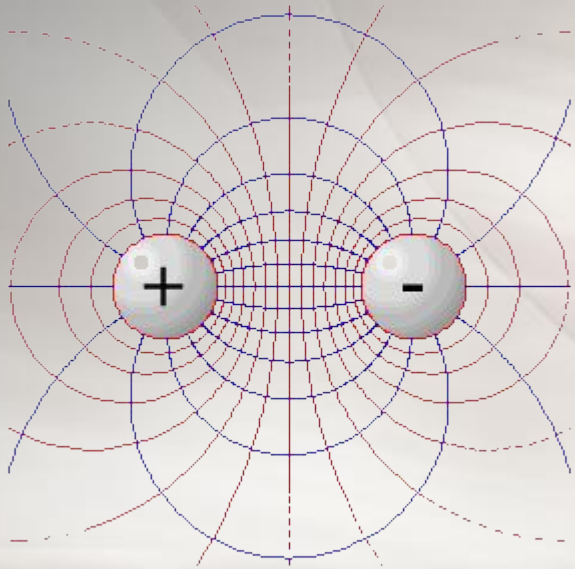
Способы задания алгоритма

- **графический** - с помощью специальных графических символов

Блок-схема

Способы задания алгоритма

- **формульный** - с помощью математических формул, которые определяют порядок вычислений



$$\varphi_0(M) = \frac{q_1}{4\pi\varepsilon_0}\frac{1}{r_{M_1M}} + \frac{q_2}{4\pi\varepsilon_0}\frac{1}{r_{M_2M}}$$

Формула для расчета поля электростатического потенциала, созданного двумя точечными зарядами

Способы задания алгоритма

- **табличный** - в виде таблицы, в которой фиксируются этапы исполнения алгоритма и результаты исполнения

Склады	Магазины МГ1	Магазины МГ2	Магазины МГ3	A_i (U_i)
СК1	4	3	2 5	40 (0)
	30+	10	3	
СК2	3 6	2	1	50 (-1)
	3	10+	40-	
СК3	5 10	4	3 7	60 (1)
	5	60	4	
СК4	8	7 6	6 3	30 (4)
	30-	-1	+ -3	
B_j (V_j)	60 (4)	80 (3)	40 (2)	

Поиск оптимального решения транспортной задачи методом потенциалов

Способы задания алгоритма

- **псевдокод** – система обозначений и правил, предназначенная для единообразной записи алгоритмов; занимает промежуточное место между естественным и формальным языками

```
НОД( X, Y )
     x:=X;
     y:=Y;
     пока( x ≠ y ) повторять
          если( x > y ) то x:=x-y;
          иначе           y:=y-x;
     конец цикла
     вывести x
конец
```

Нахождение наибольшего общего делителя

Способы задания алгоритма

- **Запись на языке программирования**

```
int gcd( int x, int y )
{
    assert( x > 0 && y > 0 );
    while( x != y )
    {
        if( x > y ) x = x - y;
        else        y = y - x;
    }
    return x;

}
```

Нахождение наибольшего общего делителя

Правила построения алгоритма

- **Первое правило** – при построении алгоритма необходимо задать множество объектов, с которыми будет работать алгоритм
  - *Формализованное* (закодированное) *представление этих объектов* носит название данных
  - Алгоритм приступает к работе с некоторым набором данных, которые называются *входными*, и в результате своей работы выдает данные, которые называются *выходными*

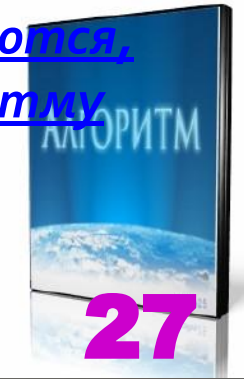
Алгоритм преобразует входные данные в выходные.

Пока мы не имеем формализованных входных данных, мы не можем построить алгоритм

Правила построения алгоритма

- **Второе правило** – для работы алгоритма требуется память
  - *В памяти размещаются* входные данные, с которыми алгоритм начинает работать, промежуточные данные и выходные данные, которые являются результатом работы алгоритма
  - Память является дискретной, т. е. состоящей из отдельных ячеек
  - Поименованная ячейка памяти носит название переменной

В теории алгоритмов размеры памяти не ограничиваются, т. е. считается, что мы можем предоставить алгоритму любой необходимый для работы объем памяти



Правила построения алгоритма

- **Третье правило** – **дискретность**
  - Алгоритм строится из отдельных шагов (действий, операций, команд). Точнее, из множества шагов

- **Четвертое правило** – **детерминированность**
  - После каждого шага необходимо указывать, какой шаг выполняется следующим, либо давать команду остановки

Правила построения алгоритма

- **Пятое правило** – **сходимость (результативность)**
  - Алгоритм должен *завершать работу после конечного числа шагов*
  - При этом необходимо указать, что считать *результатом работы алгоритма*

Свойства алгоритма

- **Дискретность** (прерывность, раздельность) – алгоритм должен *представлять процесс решения задачи как последовательное выполнение простых* (или ранее определенных) *шагов*
  - *Каждое действие*, предусмотренное алгоритмом, *исполняется* только *после того, как закончилось исполнение предыдущего*

Свойства алгоритма

- **Определенность** – каждое правило алгоритма должно быть четким, однозначным
  - Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче

- **Результативность** (конечность) – алгоритм должен приводить к решению задачи за конечное число шагов

Свойства алгоритма

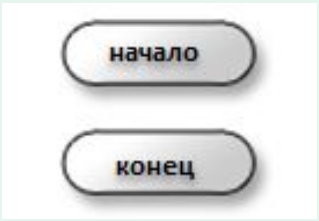
- **Массовость** – алгоритм решения задачи разрабатывается в общем виде, то есть он должен быть применим для некоторого класса задач, различающихся только исходными данными
  - исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма

Блок-схема алгоритма

Блок-схема алгоритма - *графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и* **блоков** - *графических символов, каждый из которых соответствует одному шагу алгоритма*

Внутри блока дается описание соответствующего действия

Основные типы блоков

	***Начало и конец алгоритма*** (для функций «Вход», «Выход»)
	***Блок действия*** Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы предста расположения). Внутри фигуры записывают непос венно сами операции, например, операцию присваивания: **a = 10*b + c**

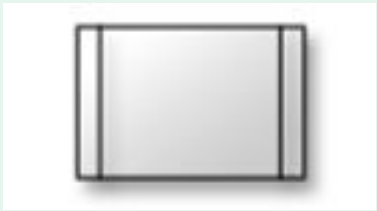
Основные типы блоков

Логический блок (блок условия)

Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента

Примеры решения: в общем случ
сравнение (три выхода: >, <, =)

Основные типы блоков



Блок предопределенного процесса

Символ отображает выполнение процесса, состоящего из одной или нескольких операций, который определен в другом месте программы (в подпрограмме, модуле). Внутри символа записывается название процесса и передаваемые в него данные.

Например, в программировании – процедуры или функции

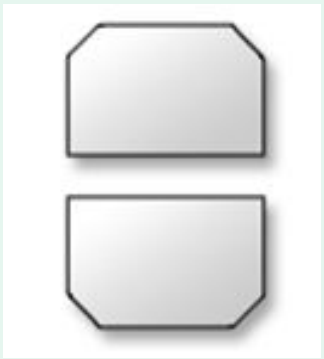
Основные типы блоков

Блок ввода информации

Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)

Данный символ не определяет носителя данных (для указания типа носителя данных используются специфические си

Основные типы блоков



Граница цикла

Символ состоит из двух частей − соответственно, начало и конец цикла − операции, выполняемые внутри цикла, размещаются между ними

Условия цикла и приращения записываются внутри символа начала или конца в зависимости от типа организаци

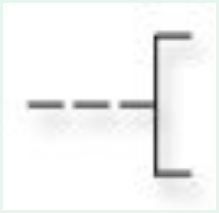
Основные типы блоков

Соединитель

Символ отображает вход в часть схемы и выход из другой части этой схемы

Используется для обрыва линии и продолжения её в другом месте (для избегания излишних пересечений или слишком длинных линий, а также, если схема состои нескольких страниц)

Основные типы блоков



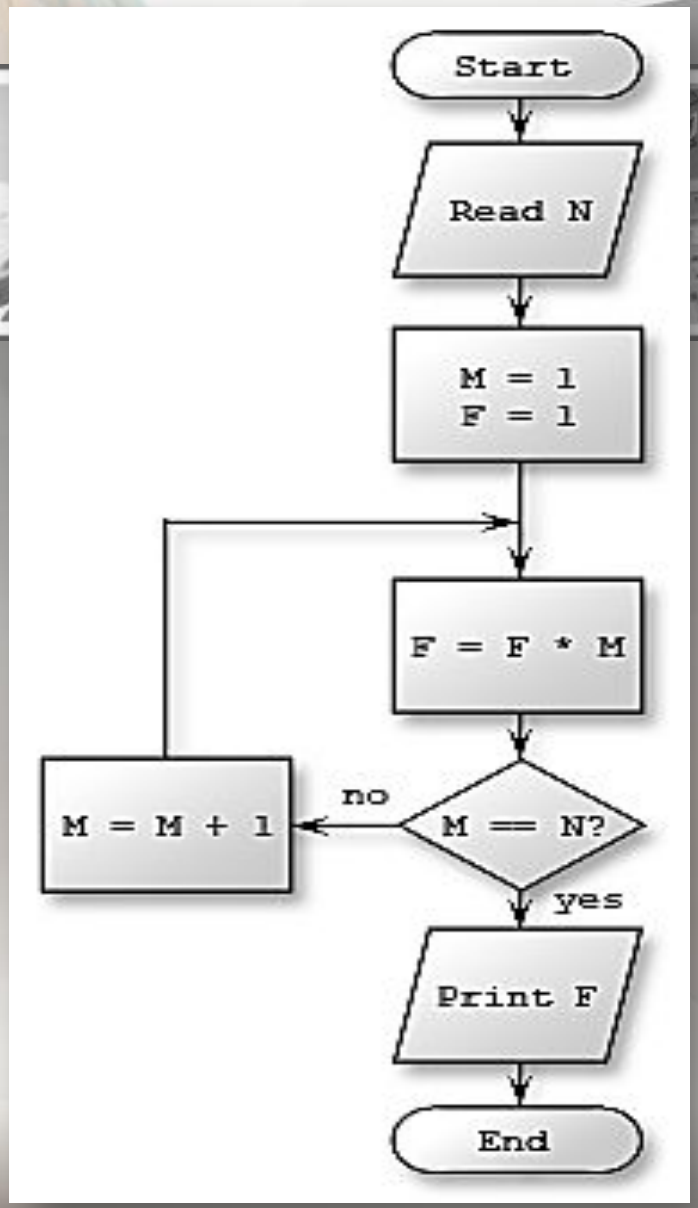
Комментарий

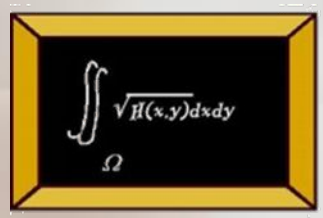
Используется для более подробного описания шага, процесса или группы процессов

Описание помещается со стороны квадратной скобки и охватывается ей по всей высоте

Пунктирная линия идет к описываемому элементу, либо группе элементов

Пример блок-схемы





Блок-схема алгоритма вычисления факториала числа N

Пример блок-схемы

Блок-схема алгоритма нахождения максимального из двух значений

Пример блок-схемы

Составить блок-схему алгоритма определения высот h_a, h_b, h_c треугольника со сторонами a, b, c, если

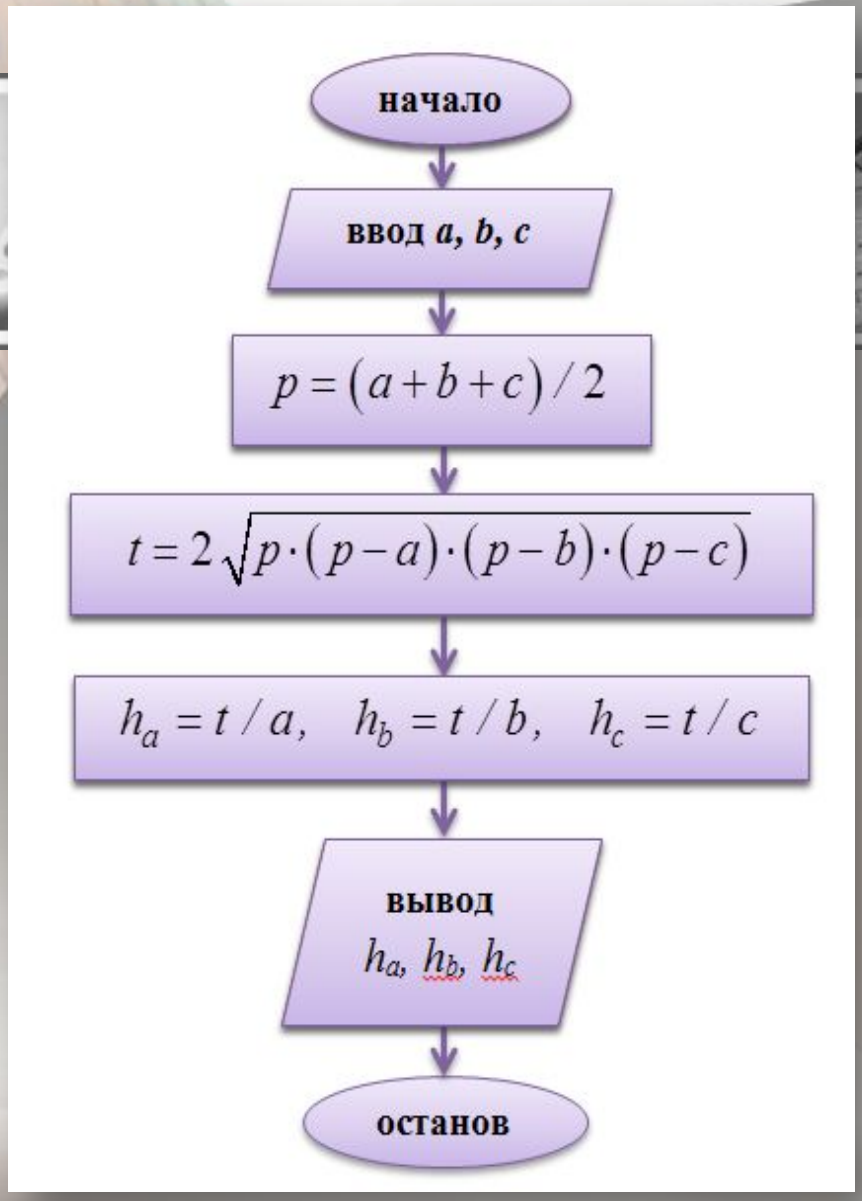
$$h_a = \frac{2}{a}\sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$$

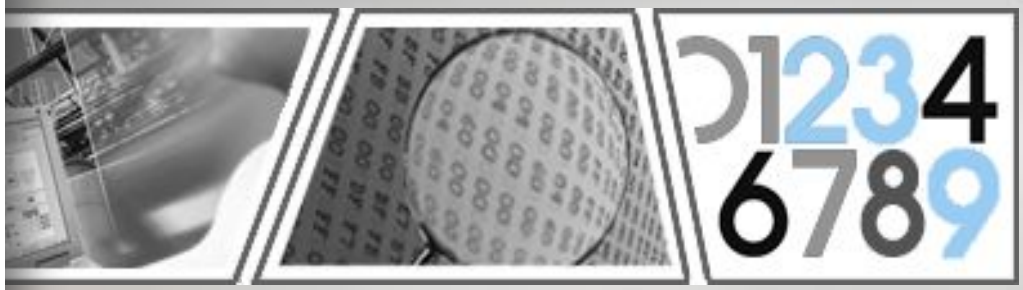
$$h_b = \frac{2}{b}\sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$$

$$h_c = \frac{2}{c}\sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$$

где $p = (a+b+c)/2$

Пример блок-схемы





$$t = 2\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$$

$$h_a = t/a$$
$$h_b = t/b$$
$$h_c = t/c$$