**Алгоритмизация и программирование**

**Алгоритмы, виды алгоритмов, описание алгоритмов. Формальное исполнение алгоритмов**

Термин «алгоритм», впервые употребленный в современном значении Лейбницем (1646–1716), является латинизированной формой имени великого персидского математика Мухаммеда бен Муссы аль-Хорезми (ок. 783 – ок. 850). Его книга «Об индийском счете» в XII в. была переведена на латинский язык и пользовалась широкой популярностью не одно столетие. Имя автора европейцы произносили как *Алгоритми* (*Algorithmi*), и со временем так стали называть в Европе всю систему десятичной арифметики.

Научное определение алгоритма дал А. Чёрч в 1930 году. В наше время понятие алгоритма является одним из основополагающих понятий вычислительной математики и информатики.

**Алгоритм** — это точное и полное описание последовательности действий над заданными объектами, позволяющее получить конечный результат.

Можно сказать, что алгоритм решения какой-либо задачи — это последовательность шагов реализации (или нахождения) этого решения, а процесс построения алгоритма (алгоритмизация) — разложение задачи на элементарные действия или операции.

Область математики, известная как *теория алгоритмов*, посвящена исследованию свойств, способов записи, области применения различных алгоритмов, а также созданию новых алгоритмов. Теория алгоритмов находит широкое применение в различных областях деятельности человека — в технике, производстве, медицине, образовании и т. д. Появление компьютера позволило решать чрезвычайно сложные, трудоемкие задачи.

Определение алгоритма для применения в области информатики нуждается в некотором уточнении. Во-первых, решение задач в информатике всегда связано с преобразованием информации, а значит, исходными данными и результатом работы алгоритма должна быть информация. Это может быть представлено в виде схемы.



Во-вторых, алгоритмы в информатике предназначены для реализации в виде компьютерных программ или для создания некоторой компьютерной технологии. Для выполнения алгоритма требуется конечный объем оперативной памяти и конечное время.

Основные требования, предъявляемые к алгоритмам:

***Дискретность (прерывность)***: алгоритм должен представлять решение задачи в виде последовательности простых (или ранее определенных) этапов (шагов). Каждый шаг алгоритма формулируется в виде инструкций (команд).

***Определенность (детерминированност***ь; лат. *determinate* — определенность, точность): шаги (операции) алгоритма должны допускать однозначную трактовку и быть понятными для исполнителя алгоритма. Это свойство указывает на то, что любое действие в алгоритме должно быть строго определено и описано для каждого случая.

***Массовость:*** алгоритм должен давать решение не только для конкретного набора значений, а для целого класса задач, который определяется диапазоном возможных исходных данных (область применимости алгоритма). Свойство массовости подразумевает использование переменных в качестве исходных данных алгоритма.

***Результативность:***алгоритм должен давать конкретный результат, т. е. должны быть рассмотрены все возможные ситуации и для каждой из них получен результат. Под результатом может пониматься и сообщение о том, что задача решения не имеет.

***Конечность:*** количество шагов алгоритма должно быть конечным.

***Эффективность:***количество шагов и сами шаги алгоритма должны быть такими, чтобы решение могло быть найдено за конечное и, более того, приемлемое время.

Для оценки и сравнения алгоритмов существует много критериев. Чаще всего *анализ алгоритма* (или, как говорят, *анализ сложности алгоритма*) состоит в оценке временных затрат на решение задачи в зависимости от объема исходных данных. Используются также термины «временная сложность», «трудоемкость» алгоритма. Фактически эта оценка сводится к подсчету количества основных операций в алгоритме, поскольку каждая из них выполняется за заранее известное конечное время. Кроме временной сложности, должна оцениваться также емкостная сложность, т. е. увеличение затрат памяти в зависимости от размера исходных данных. Оценка сложности дает количественный критерий для сравнения алгоритмов, предназначенных для решения одной и той же задачи. Оптимальным (наилучшим) считается алгоритм, который невозможно значительно улучшить в плане временных и емкостных затрат.

Анализом сложности алгоритмов, исследованием классов задач, решаемых с помощью алгоритмов той или иной сложности, и многими другими теоретическими вопросами занимается специальная область информатики.

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных базовых элементов.

***Логическая структура*** любого алгоритма может быть представлена комбинацией трех базовых структур:

1. **следование** — образуется из последовательности действий, следующих одно за другим;
2. **ветвление (развилка)**— обеспечивает в зависимости от результатов проверки условия (ДА или НЕТ) выбор одного из альтернативных путей алгоритма;
3. **цикл** — обеспечивает многократное выполнение некоторой совокупности действий, которая называется *телом цикла.*

Для описания алгоритмов наиболее распространены следующие методы (языки):

***Обычный язык.*** Изложение алгоритма ведется на обычном языке с разделением на последовательные шаги.

***Блок-схемы***. Графическое изображение алгоритма с помощью специальных значков-блоков.

***Формальные алгоритмические языки (языки программирования).***При записи алгоритмов используют строго определенный набор символов и составленных из них специальных зарезервированных слов. Имеют строгие правила построения языковых конструкций.

***Псевдокод.*** Синтез алгоритмического и обычного языков. Элементы некоторого базового алгоритмического языка используются для строгой записи базовых структур алгоритма.

***Словесный способ*** (запись на обычном языке) не имеет широкого распространения, т. к. таких описаний есть ряд недостатков:

* строго не формализуемы;
* достаточно многословны;
* могут допускать неоднозначность толкования отдельных предписаний;
* сложные задачи с анализом условий, с повторяющимися действиями трудно представляются в словесной или словесно-формульной форме.

***Графический способ***представления информации является более наглядным и компактным по сравнению со словесным способом. При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое графическое представление алгоритма называется *блок-схемой.* Определенному типу действия (ввод/вывод данных, проверка условия, вычисление выражения, начало и конец алгоритма и т. п.) соответствует определенная геометрическая фигура — *блочный символ.* Блоки соединяются между собой линиями переходов, которые определяют очередность выполнения действий.

|  |  |  |
| --- | --- | --- |
| **Название символа** | **Графическое изображение** | **Комментарии** |
| Пуск/Останов (блоки начала и конца алгоритма) | https://examer.ru/i/theory/inf/11.png | Указание на начало или конец алгоритма |
| Ввод/Вывод данных (блоки ввода, вывода | https://examer.ru/i/theory/inf/12.png | Организация ввода/вывода в общем виде |
| Процесс (операторные блоки) | https://examer.ru/i/theory/inf/13.png | Выполнение вычислительного действия или последовательности действий (можно объединять в один блок), которые изменяют значение, форму представления или размещение данных |
| Условие (условный блок) | https://examer.ru/i/theory/inf/14.png | Выбор направления выполнения алгоритма. Если условие, записанное внутри ромба, выполняется, то управление передается по стрелке «да», в противном случае — по стрелке «нет». Таким образом, реализуется процесс изменения последовательности вычислений в зависимости от выполнения условия |
| Начало цикла с параметром | https://examer.ru/i/theory/inf/15.png | Используется для организации циклических конструкций с известным количеством итераций (повторений) и известным шагом изменения параметра цикла. Внутри блока для параметра цикла указываются через запятую его начальное значение, конечное значение и шаг изменения. Цикл, для которого неизвестно количество повторений, записывается с помощью условного и операторных блоков |
| Предопределенный процесс | https://examer.ru/i/theory/inf/16.png | Используется для указания обращений к вспомогательным алгоритмам, существующим автономно в виде некоторых самостоятельных модулей, и для обращения к библиотечным подпрограммам |
| Печать сообщений (документ) | https://examer.ru/i/theory/inf/17.png | Вывод результатов на печать |

При составлении блок-схемы необходимо проверять выполнение следующих условий:

1. из каждого прямоугольника и параллелограмма (кроме конца алгоритма) должна выходить только одна стрелка;
2. в каждый прямоугольник и параллелограмм (кроме начала алгоритма) должна входить хотя бы одна стрелка;
3. в каждый ромб должна входить хотя бы одна стрелка, а выходить из него — две стрелки, помеченные словами «ДА» и «НЕТ».

**Псевдокод** занимает промежуточное положение между естественным языком и языками программирования. В псевдокоде не приняты строгие синтаксические правила для записи команд, что отличает формальные языки программирования. Однако в псевдокоде есть некоторые конструкции, которые присущи формальным языкам, что облегчает переход от записи алгоритма на псевдокоде к записи алгоритма на языке программирования. Псевдокоды бывают разные. Рассмотрим учебный (школьный) алгоритмический язык АЯ.

Алфавит учебного алгоритмического языка является открытым. В него могут быть введены любые понятные всем символы: русские и латинские буквы, знаки математических операций, знаки отношений, специальные знаки и т. д. Кроме алфавита, в алгоритмической нотации определяются *служебные слова*, которые являются неделимыми. Служебные слова обычно выделяются жирным шрифтом или подчеркиванием. К служебным словам относятся:

|  |  |  |
| --- | --- | --- |
| **алг** — заголовок алгоритма | **нц** — начало цикла | **знач** |
| **нач** — начало алгоритма | **кц** — конец цикла | **и** |
| **кон** — конец алгоритма | **дано** | **или** |
| **арг** — аргумент | **надо** | **не** |
| **рез** — результат | **если** | **да** |
| **цел** — целый | **то** | **нет** |
| **сим** — символьный | **иначе** | **при** |
| **лит** — литерный | **всё** | **выбор** |
| **лог** — логический | **пока** | **утв** |
| **вещ** — вещественный | **для** | **ввод** |
| **таб** — таблица | **от** | **вывод** |
| **длин** — длина | **до** |   |

Общий вид записи алгоритма на псевдокоде:

**алг** — название алгоритма (аргументы и результаты)

**дано** — условие применимости алгоритма

**надо** — цель выполнения алгоритма

**нач** — описание промежуточных величин

последовательность команд (тело алгоритма)

**кон**

Часть алгоритма от слова **алг** до слова **нач** называется *заголовком*, а часть, заключенная между словами **нач** и **кон,** — *телом алгоритма* (исполняемой частью алгоритма).

В предложении **алг** после названия алгоритма в круглых скобках указываются *характеристики* (**арг, рез**) и *тип значения* (**цел, вещ, сим, лит** или **лог**) всех входных (аргументы) и выходных (результаты) переменных. При описании массивов (таблиц) используется служебное слово **таб**, дополненное именем массива и граничными парами по каждому индексу элементов массива.

Команды учебного языка:

1. ***Оператор присваивания***, который обозначается «:=» и служит для вычисления выражений, стоящих справа, и присваивания их значений переменным, указанным в левой части. Например, если переменная а имела значение 5, то после выполнения оператора присваивания а:= а + 1, значение переменной а изменится на 6.

2. ***Операторы ввода/вывода:***

**ввод** (список имен переменных)

**вывод** (список вывода)

Список вывода может содержать комментарии, которые заключаются в кавычки.

3. ***Оператор ветвления*** (с использованием команды **если...то… иначе…всё; выбор**);

4. ***Операторы цикла*** (с использованием команд **для, пока, до**).

Запись алгоритма на псевдокоде:



Здесь в предложениях **дано** и **надо** после знака «|» записаны *комментарии*. Комментарии можно помещать в конце любой строки, они существенно облегчают понимание алгоритма.

При записи алгоритма в словесной форме, в виде блок-схемы или на псевдокоде допускается произвольное изображение команд. Вместе с тем такая запись позволяет понять человеку суть дела и исполнить алгоритм. Однако алгоритм, предназначенный для исполнения на компьютере, должен быть записан на строго формализованном языке. Такой язык называется **языком программирования**, а запись алгоритма на этом языке — **компьютерной программой.**

Для решения одной и той же задачи можно предложить несколько алгоритмов. Алгоритмы составляются с ориентацией на определенного *исполнителя алгоритма*. У каждого исполнителя имеется свой конечный набор команд, которые для него понятны и исполняемы. Этот набор называется **системой команд исполнителя.** Пользуясь системой команд, исполнитель может выполнить алгоритм формально, не вникая в содержание поставленной задачи. От исполнителя требуется только строгое выполнение последовательности действий, предусмотренной алгоритмом. Таким образом, в общем случае алгоритм претерпевает изменения по стадиям:

* первая стадия — алгоритм должен быть представлен в форме, понятной человеку, который его разрабатывает;
* вторая стадия — алгоритм должен быть представлен в форме, понятной исполнителю алгоритма (вторая стадия может отсутствовать, если исполнять алгоритм будет сам разработчик).

Вопросы для закрепления

1.Дайте определение понятия Алгоритм;

2.Перечислите основные требования к алгоритмам;

3.Опишите логическую структуру любого алгоритма;

4.Как осуществляется написание алгоритмов на алгоритмическом языке (на языке программирования);

5.Перечислите недостатки словесного способа представления алгоритмов;

6. Выполнение каких условий следует проверять при составлении блок-схемы?

7. Как называется конечный набор команд исполнителя?

8.Перечислите и расшифруйте стадии изменения алгоритма.