

Разработка алгоритмов решения задачи

Принято различать логическое и физическое проектирование. Логическое проектирование не учитывает особенностей среды, в которой будет выполняться программа (технические и программные средства компьютера). При выполнении физического проектирования все эти параметры должны быть учтены.

Логическое проектирование при процедурном подходе предполагает детальную проработку последовательности действий будущей программы. Его начинают с определения структуры будущего программного продукта: отдельная программа или программная система, состоящая из нескольких взаимосвязанных программ. Затем переходят к разработке алгоритмов программ.

Алгоритм - формально описанная последовательность действий, которые необходимо выполнить для получения требуемого результата.

Различают последовательности действий (вычислений) линейной, разветвленной и циклической структуры.

Линейная структура процесса вычислений предполагает, что для получения результата необходимо выполнить некоторые операции в определенной последовательности. Например, для определения площади треугольника по формуле Герона необходимо сначала определить полупериметр треугольника, а затем по формуле его площадь.

Разветвленная структура процесса вычислений предполагает, что конкретная последовательность операций зависит от значений одного или нескольких параметров. Например, если дискриминант квадратного уравнения не отрицателен, то уравнение имеет два корня, а если отрицателен, то действительных корней нет.

Циклическая структура процесса вычислений предполагает, что для получения результата некоторые действия необходимо выполнить несколько раз. Например, для того, чтобы получить таблицу значений функции на заданном интервале изменения аргумента с заданным шагом, необходимо соответствующее количество раз определить следующее значение аргумента и посчитать для него значение функции.

Процессы вычислений циклической структуры в свою очередь можно разделить на три группы:



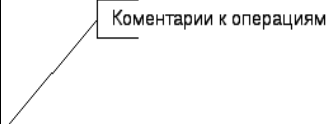
1. циклические процессы, для которых количество повторений известно – **счетные циклы или циклы с заданным количеством повторений**;
2. циклические процессы, завершающиеся по достижении или нарушении некоторых условий - **итерационные циклы**;
3. циклические процессы, из которых возможны два варианта выхода: выход по завершении процесса и досрочный выход по какому-либо дополнительному условию - **поисковые циклы**.

Формальное описание алгоритмов осуществляют с использованием схем алгоритмов и псевдокодов. На изображение схем алгоритмов существует ГОСТ 19.701-90, согласно которому каждой группе действий ставится в соответствие блок особой формы.

Некоторые часто используемые обозначения приведены в табл. 1.

Таблица 1. Основные элементы схем алгоритма.

Наименование	Обозначение	Функция
Блок начало-конец (пуск-остановка)		Элемент отображает вход из внешней среды или выход из нее (наиболее частое применение – начало и конец программы). Внутри фигуры записывается соответствующее действие.
Блок вычислений (вычислительный блок)		Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы представления, расположения). Внутри фигуры записывают непосредственно сами операции, например, операцию присваивания $a = 10*b + c$.
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. Вход в элемент обозначается линией, входящей обычно в верхнюю вершину элемента. Если выходов два или три то обычно каждый выход обозначается линией, выходящей из оставшихся вершин (боковых и нижней). Если выходов больше трех, то их следует показывать одной линией, выходящей из вершины (чаще нижней) элемента, которая затем разветвляется. Соответствующие результаты вычислений могут записываться рядом с линиями, отображающими эти пути. Примеры решения: в общем случае – сравнение (три выхода: $>$, $<$, $=$); в программировании – условные операторы <i>if</i> (два выхода: <i>true</i> , <i>false</i>) и <i>case</i> (множество выходов).
Предопределенный процесс		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 – вызов процедуры или функции.
Данные (ввод-вывод)		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 символы).
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 цикла – в зависимости от типа организации цикла. Часто для изображения на блок-схеме цикла вместо данного символа используют символ решения, указывая в нем условие, а одну из линий выхода замыкают выше в блок-схеме (перед операциями цикла).
Соединитель		Символ отображает вход в часть схемы и выход из другой части этой схемы. Используется для обрыва линии и продолжения ее в другом месте (для избегания излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц). Соответствующие соединительные символы должны иметь одинаковое (при том уникальное) обозначение.
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 (при этом группа выделяется замкнутой пунктирной линией). Также символ комментария следует использовать в тех случаях, когда объём текста, помещаемого внутри некоего символа (например, символ процесса, символ данных и др.), превышает размер самого этого символа.

*Для создания блок схем алгоритмов удобно использовать программу **Microsoft Office Visio** или её аналоги.

При разработке алгоритма каждое действие обозначают соответствующим блоком, показывая их последовательность линиями со стрелками на конце. Для простоты чтения схемы желательно, чтобы линия входила в блок сверху, а выходила снизу. Если линии идут не слева направо и не сверху вниз, то стрелка в конце линии обязательна, в противном случае ее можно не ставить.

В случае, когда схема алгоритма не уместается на листе, используют соединители. При переходе на другой лист или получении управления с другого листа в комментариях указывается номер листа, например «с листа 3» «на лист 1».

В теории программирования доказано, что для записи любого сколь угодно сложного алгоритма достаточно трех базовых структур:

- **следование** - обозначает последовательное выполнение действий (рис. 2, а);
- **ветвление** - соответствует выбору одного из двух вариантов действий (рис. 2, б);
- **цикл-пока** - определяет повторение действий, пока не будет нарушено условие, выполнение которого проверяется в начале цикла (рис. 2, в).

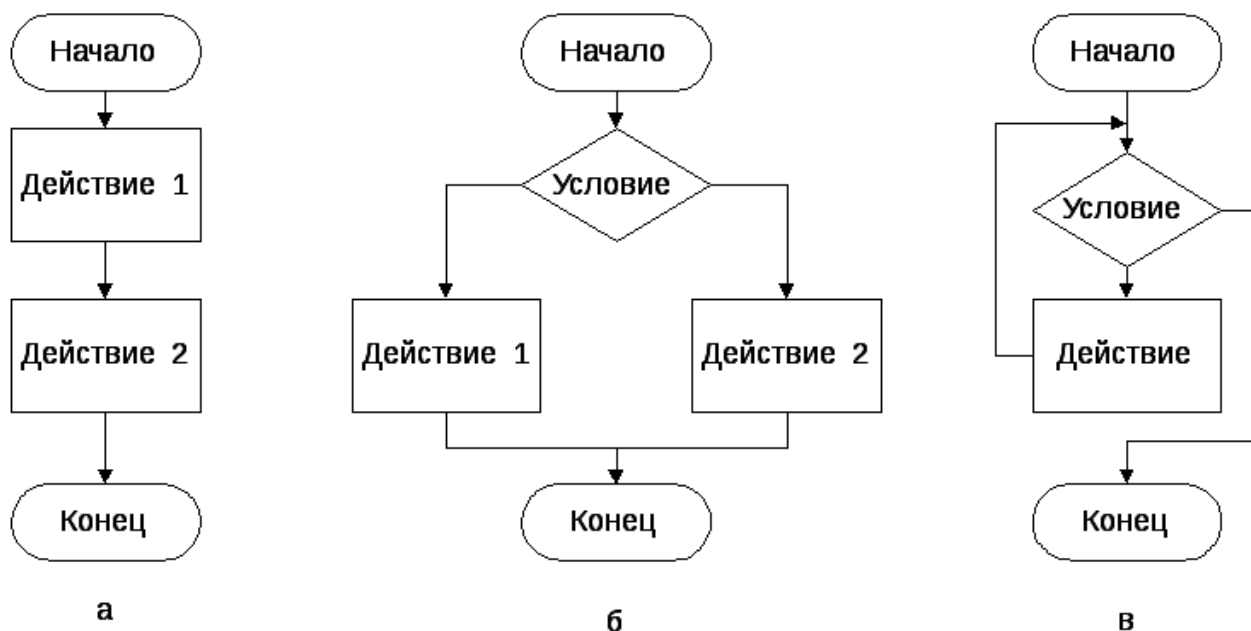


Рис. 2. Базовые алгоритмические структуры

Помимо базовых структур используют три дополнительные структуры,

производные от базовых структур:

- **выбор** - выбор одного варианта из нескольких в зависимости от значения некоторой величины (рис. 3, а);
- **цикл до** - повторение некоторых действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле (рис. 3, б);
- **цикл с заданным числом повторений** (счетный цикл) - повторение некоторых действий указанное число раз (рис. 3, г).

На рис. 1.3б, 1.3г и е показано, как каждая из дополнительных структур может быть реализована через базовые структуры.

Перечисленные структуры были положены в основу **структурного программирования** - технологии, которая представляет собой набор рекомендаций по уменьшению количества ошибок в программах [1]. В том случае, если в схеме алгоритма отсутствуют другие

варианты передачи управления, алгоритм называют структурным, подчеркивая, что он построен с учетом рекомендаций структурного программирования.

Схема алгоритма детально отображает особенности разработанного алгоритма. Иногда такой высокий уровень детализации не позволяет выделить суть алгоритма. В этих случаях для описания алгоритма используют псевдокод.

Псевдокод - описание алгоритма, которое базируется на тех же основных структурах, что и структурные схемы алгоритма. Описать на псевдокоде неструктурный алгоритм нельзя.

Для каждой структуры используют свою форму описания, которая, в общем случае может быть произвольной.

Программа должна начинаться с ввода исходных данных. Стоит заметить, что любой ввод данных пользователем должен сопровождаться запросом на ввод, чтобы пользователь знал, чего от него ждет программа после запуска. На схеме алгоритма и при записи псевдокодов этот запрос часто не указывают.

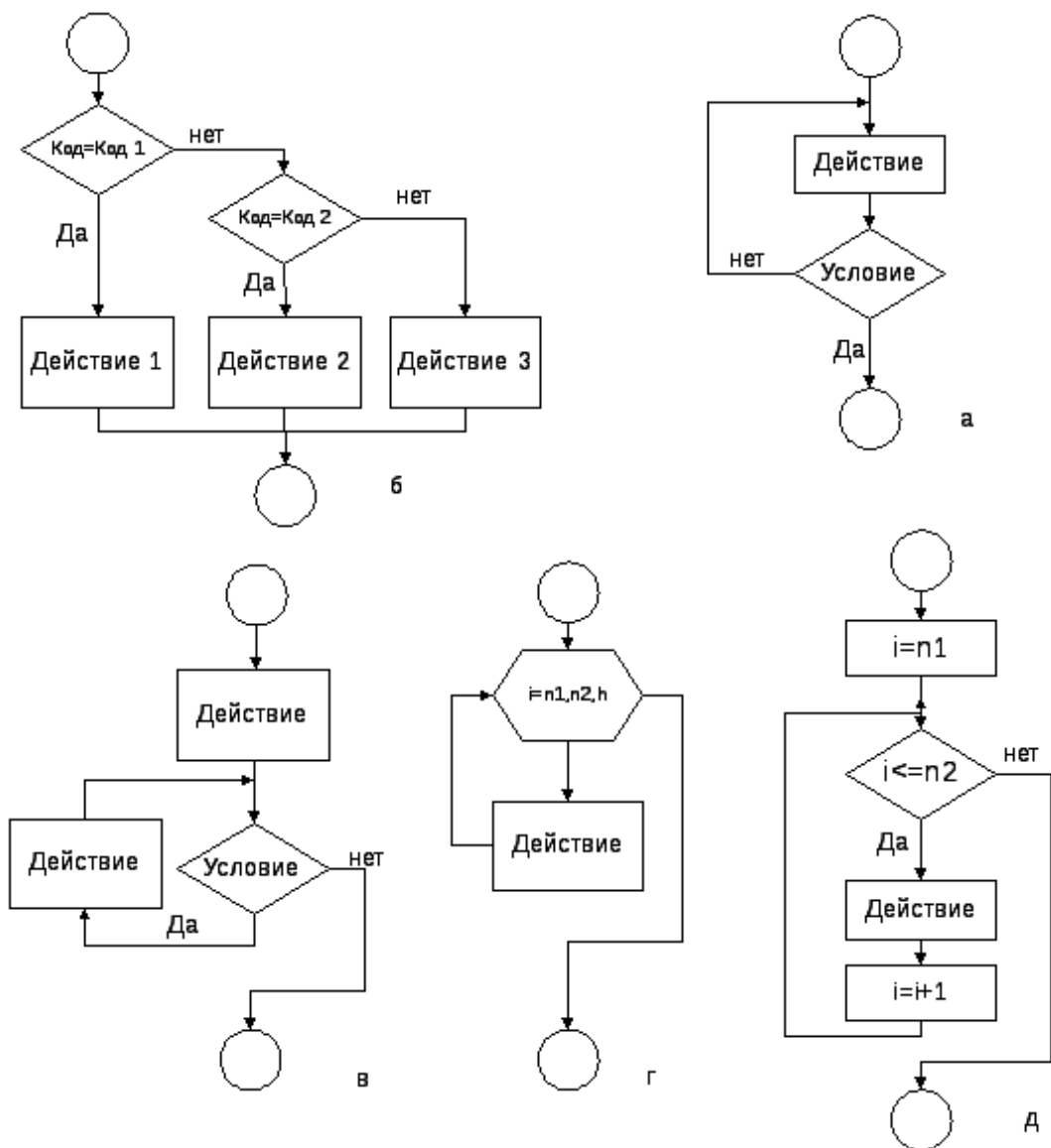


Рис.3. Дополнительные структуры и их реализация через базовые структуры

Задание: Нарисуйте блок схемы алгоритмов решения следующих задач.

1. Алгоритм вычисления значения выражения $K=3b+6a$
2. Алгоритм, определяющий, пройдет ли график функции $y=3x+4$ через точку с координатами x_1, y_1
3. Алгоритм, определяющий факториал натурального числа n
4. Необходимо определить наибольший общий делитель двух натуральных чисел A и B . Для решения поставленной задачи используем алгоритм Евклида, который заключается в последовательной замене большего из чисел на разность большего и меньшего, пока числа не станут равны. Рассмотрим данный алгоритм на двух примерах.