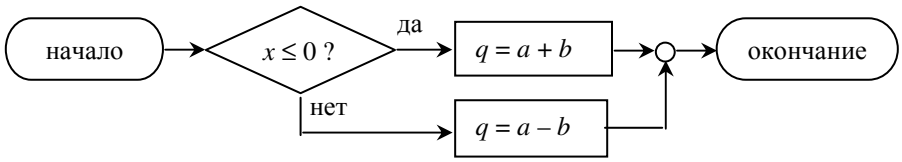
РАЗДЕЛ 2. ПРОГРАММНЫЕ СРЕДСТВА ИНФОРМАТИКИ

1.3. Структурное описание алгоритма

При структурном (в виде *блок-схемы*) описании алгоритм изображается в виде графической схемы, состоящей из так называемых вершин (блоков) и дуг (ребер). Вершины изображаются овалами, прямоугольниками и т.д. В них записываются шаги алгоритма. Дуги изображаются линиями со стрелками. Они соединяют вершины, показывая последовательность выполнения шагов, т.е. соответствуют отношениям условного или безусловного следования. Дуги, соответствующие отношению безусловного следования, не отмечаются.

Например,



В блок-схеме присутствуют вершины разного типа, основными из которых являются:

1). Вершины начала и окончания (изображаются овалами). У вершины-начала нет входящих дуг. У нее имеется лишь *одна* исходящая, направленная к вершине, с которой начинается алгоритм. У вершины-окончания нет исходящих дуг и может быть несколько входящих.

2). Вершины-действия (изображаются прямоугольниками) соответствуют шагам, в которых выполняются действия. Каждая такая вершина может иметь несколько входящих дуг и только одну исходящую.

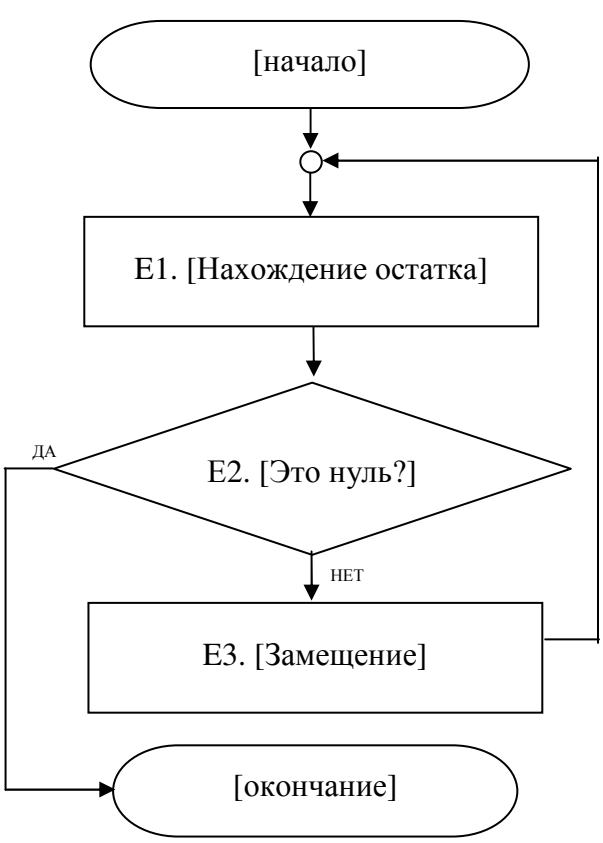
3). Вершины ввода/вывода (изображаются параллелограммами) соответствуют шагам, в которых выполняется ввод или вывод данных. Каждая такая вершина может иметь несколько входящих дуг и только одну исходящую.

4). Вершины-условия (изображаются ромбами) соответствуют шагам, в которых проверяются условия. Каждая такая вершина может иметь несколько входящих дуг и обязательно не менее двух исходящих. Если условием является логическое выражение, то исходящих дуг две. Одна из них соответствует отношению следования при условии, что проверяемое логическое выражение истинно. Такая дуга отмечается меткой «Да». Другая исходящая дуга соответствует отношению следования при условии, что проверяемое логическое выражение ложно. Такая дуга отмечается меткой «Нет».

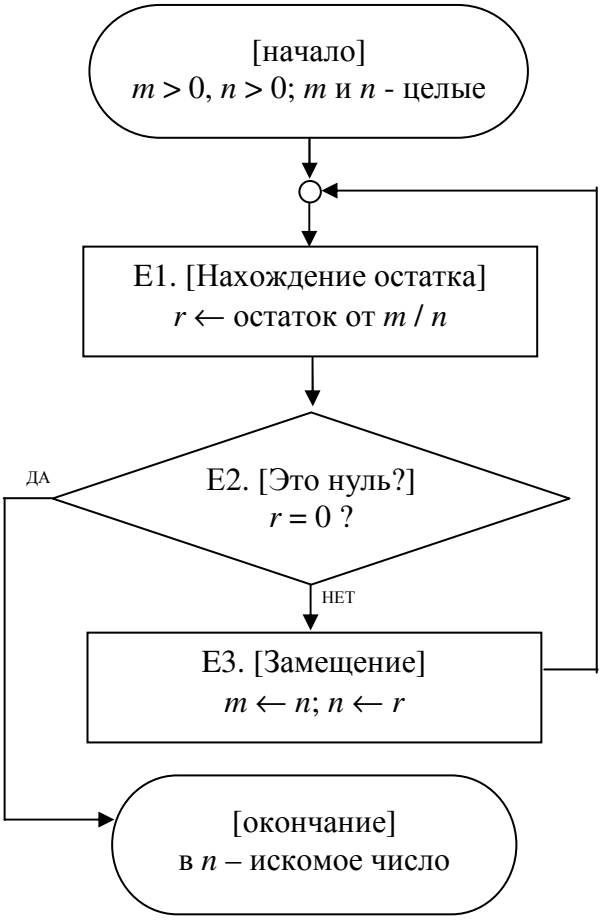
5). Если у вершины несколько входящих дуг, то для их объединения можно использовать специальную вершину, изображаемую кругом небольшого размера. Исходящая из нее дуга направлена к вершине, имеющей эти входящие дуги.

Блок-схема может быть краткой и подробной (укрупненной). В краткой блок-схеме в вершинах указывается лишь наименование и основная суть шагов. В подробной блок-схеме в вершинах указывается полное описание шагов. Блок-схема первого типа сопровождает пошаговое описание алгоритма, а блок-схема второго типа может его заменить.

Например, алгоритм Евклида может быть описан следующими блок-схемами:



а) краткая схема



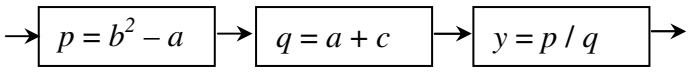
б) подробная блок-схема

Блок-схема – часто наиболее рациональный способ, обеспечивающий понимание любого алгоритма, но перегружать ее деталями не рекомендуется. Поэтому сложные алгоритмы обычно описывают в пошаговой форме, дополняя это описание краткой блок-схемой, как это сделано в первой блок-схеме в рассматриваемом примере.

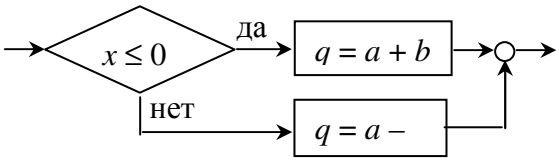
1.4. Элементарные алгоритмические структуры

Любой алгоритм представляет собой комбинацию трех элементарных алгоритмических структур: *линейная*, *ветвящаяся*, *циклическая*.

Линейная структура описывает процесс, в котором операции (действия) выполняются последовательно, в порядке их описания. Вершины, отображающие эти действия, располагаются в линейной последовательности. Такие процессы имеют место, например, при вычислении арифметических выражений, когда имеются конкретные числовые данные и над ними выполняются соответствующие условию задачи действия. Например, для вычисления y = (b^2 – a*c) / (a + c) имеем следующую алгоритмическую структуру.



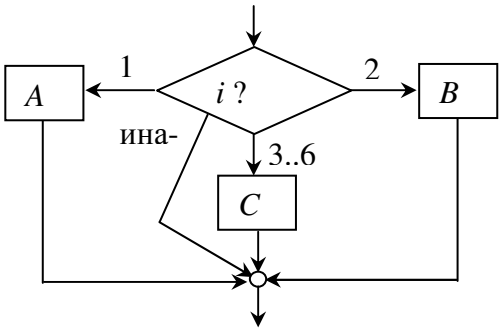
Ветвящейся называется структура, описывающая процесс, для реализации которого предусмотрено несколько направлений (ветвей). Каждое отдельное направление является отдельной ветвью. Направление ветвления выбирается в соответствии с результатом проверяемого условия. Если условием является логическое выражение, то предполагается альтернативный выбор. Такой ветвящийся процесс включает в себя две ветви и называется *простым* (или *альтернативным*). Например,



реализует вычисление

$$y = \begin{cases} a + b, \ если\ x \le 0 \\ a - b, \ если\ x > 0 \end{cases}$$

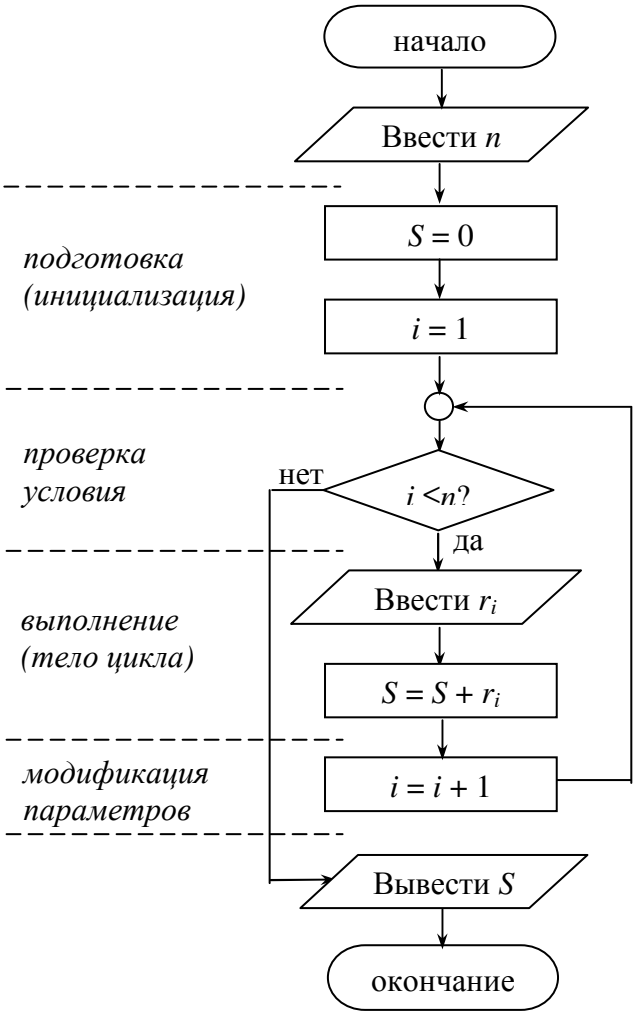
Если процесс предполагает более двух ветвей, то он называется *сложным*. Например,



Здесь 3..6 означает от 3 до 6.

Очевидно, что сложный ветвящийся процесс можно представить с помощью простых.

Циклическая структура описывает процесс, содержащий цикл. *Цикл* – это многократно повторяющаяся последовательность операций. Например, накопление суммы n чисел x_1, x_2, ..., x_n в переменной S можно описать так.



В организации цикла можно выделить следующие этапы:
– подготовка (инициализация); здесь $S \leftarrow 0$, $i \leftarrow 1$;
– выполнение (тело цикла); здесь $S \leftarrow S + x_i$;
– модификация параметров; здесь $i \leftarrow i + 1$;
– проверка условий; здесь $i \le n$.

Цикл называется *итерационным*, если число повторений тела цикла заранее не известно, а зависит от переменных, участвующих в вычислениях.

Проверка условия окончания может производиться как в конце цикла, так и в начале. В зависимости от его расположения различают цикл *с нижним окончанием* или *с постусловием* (условие проверяется после тела цикла) и цикл *с верхним окончанием* или *с предусловием* (условие проверяется перед телом цикла). Принципиальное отличие заключается в том, что в первом случае тело цикла обязательно выполняется по крайней мере один раз, а во втором – может не выполниться ни разу.

3.2. Вспомогательные (сервисные) программы

3.2.1. Программы–упаковщики (архиваторы)

Применение упаковщиков полезно для хранения файлов и передачи по коммуникационным каналам. Позволяют за счет применения специальных методов упаковки сжимать файлы во внешней памяти (говорят "*упаковывать*" или "*архивировать*"), т.е. создавать копии файлов меньшего размера. Кроме этого, они обеспечивают объединение множества таких копий в один файл, который называется *архивным*.

Таким образом, архивный файл представляет собой набор из одного или нескольких файлов, помещенных в сжатом виде в единый файл, из которого их можно при необходимости извлечь в первоначальном виде (разархивировать). Архивный файл содержит оглавление, позволяющее узнать, какие файлы в нем содержатся.

Разные архиваторы отличаются форматом создаваемых ими архивных файлов, скоростью работы, степенью сжатия файлов, удобством использования. Наиболее распространенные архиваторы имеют приблизительно одинаковые возможности, и ни один из них не превосходит другие одновременно по всем параметрам.

Наиболее популярны упаковщики Zip, Rar, Ace, Arj. Для них разработаны специальные оболочки, обеспечивающие их выполнение под управлением операционных систем (например, WinZip, WinRar, WinAce для MS Windows).

Названия архивных файлов имеют расширения, указывающие на архиватор. Например, у названий архивных файлов, созданных программой Zip, расширение zip, у архивных файлов Rar – rar, Ace – ace, Arj – arj.

3.2.2. Вирусы и антивирусные программы

Программный вирус – это небольшая программа, специально созданная для того, чтобы затруднить, исказить или исключить обработку информации на ПЭВМ. Впервые программу-вирус создал в 1989 году аспирант университета США Р. Морис. Эта программа была запущена в компьютерную сеть Министерства обороны США и вывела из строя программное обеспечение более 6000 ЭВМ.

Вирус может "приписывать" себя к другим программам (т.е. "*заражать*" их), а также выполнять различные нежелательные действия. Программа, внутри которой находится вирус, называется зараженной. Как только файл с такой программой запускается на выполнение, сначала выполняется вирус (говорят "*перехватывает управление*"). Он находит и "*заражает*" другие программы, а также выполняет какие-либо вредные действия. Для маскировки он может так действовать не всегда, а при выполнении определенных условий (например, дата, время, количество запусков). После того, как вирус выполнит предусмотренные в нем действия, он передает управление программе, в которой находится, и она работает, как обычно. Таким образом, внешне работа зараженной программы выглядит так же, как и незараженной.

Многие вирусы после запуска остаются в ОЗУ резидентно, т.е. до перезагрузки операционной системы, и время от времени выполняют вредные действия. Такие вирусы наиболее опасны. Пока заражено относительно мало программ, наличие вируса может быть практически незаметно. Однако, по происшествии некоторого времени результаты работы вируса становятся явными, например,

– некоторые программы не работают или работают неправильно;
– на экран выводятся посторонние сообщения, символы и т.п.;
– выполнение программ существенно замедляется;
– некоторые файлы испорчены;
– неожиданно форматируется жесткий диск;
и т.д.

Вирус может испортить любой файл, но заразить – только файлы определенного типа. К ним, например, относятся следующие.

1. *Исполняемые файлы*, т.е. файлы с программами (их названия имеют расширения com или exe). Вирусы, заражающие такие файлы, называются ***файловыми***. Они наиболее распространены.

2. *Загрузчик операционной системы.* В самом начале памяти каждого носителя информации, сформатированного в ОС, содержится так называемая главная загрузочная запись MBR (*Master Boot Record*). В этом блоке находится служебная информация (например, таблица разделов) и специальная программа, которая загружает в оперативную память основные компоненты (ядра) ОС. Эта программа называется *загрузчиком ОС*.

Вирусы, заражающие эту область памяти, называются ***загрузочными***, или бутовыми (от названия области Boot Record). Такой вирус начинает свою работу при начальной загрузке ОС и становится резидентным. Распространяется он, заражая загрузочные записи подключенных носителей информации устройств внешней памяти.

3. *Драйверы устройств.* Вирусы, находящиеся в них, начинают свою работу при каждом обращении к соответствующему устройству. Обычно такие вирусы заражают и исполняемые файлы.

4. *Файлы, созданные с помощью текстового процессора или табличного процессора*. Вирусы, заражающие такие файлы, называются ***макро-вирусами***. Они действуют каждый раз, когда происходит работа с этими файлами.

Антивирусные программы предназначены для предотвращения заражения программ компьютерным вирусом и ликвидации последствий заражения. Наиболее распространены следующие *виды антивирусных программ*:

1. ***Сканер*** (*детектор*) обнаруживает зараженные файлы. Сканер производит проверку заданной области файловой структуры. Требует «ручного» запуска и указания области проверки.

2. ***Доктор*** (***фаг***) “лечит” зараженные файлы, т.е. уничтожает вирусы в файлах и помогают восстанавливать зараженные программы.

3. ***Ревизор*** запоминает параметры, характеризующие исходное состояние системных файлов и областей и сравнивает с текущим состоянием. При выявлении несоответствия сообщает о них пользователю и исправляет.

4. ***Фильтр*** (***антивирусный монитор***) помогает предотвратить проникновение вирусных программ. Располагается резидентно в ОЗУ, перехватывает управление у вирусов и сообщает пользователю. Это автоматически работающая программа, не требующая ручного запуска. Она перехватывает обращения к операционной системе, которые вирусы используют для размножения и нанесения вреда.

Для защиты от несанкционированного доступа в компьютерной сети часто используется так называемый ***межсетевой (сетевой) экран***. Его еще называют ***брандмауэр*** *(brandmauer)* или ***файрвол*** *(firewall)*. Основная задача межсетевого экрана – не пропускать сетевые пакеты, которые не соответствуют заранее заданным критериям, определённым при настройке программы. Следует иметь в виду, что межсетевой экран не защищает от вирусов, а предотвращает несанкционированный доступ.

3.2.3. Программы обслуживания дисков

Кроме рассмотренных выше, существует множество других сервисных утилит. Из них наиболее часто используются следующие.

1. *Программы диагностики* – проверяют работоспособность устройств.

2. *Программы оптимизации* (*дефрагментации*) – перемещают все части каждого файла друг к другу и собирают все файлы в начале диска. За счет этого уменьшается число перемещений магнитных головок диска, в результате чего ускоряется доступ к информации и снижается износ дисковода.

3. *Программы уничтожения остатков информации в областях, занимаемых ранее удаленными файлами*. Используются для надежности уничтожения секретной информации. Необходимость выполнения такой функции вызвана тем, что

– при удалении файла уничтожается только его название в каталоге, а не сам файл;

– обычно объем данных в файле меньше, чем отведенное для него пространство памяти, поэтому в кластере, занимаемом последней частью (“*хвостом*”) файла, могут сохраниться остатки информации от файла, который был там ранее.

4. *Программы динамического сжатия* – автоматически (*динамически*) сжимают информацию при записи на диск, а при считывании с диска восстанавливают ее в первоначальном виде. За счет этого увеличивается объем информации, которую можно хранить в памяти. Для программ, например, объем файлов уменьшается в 1,5 раза, а для баз данных – в 4-5 раз.