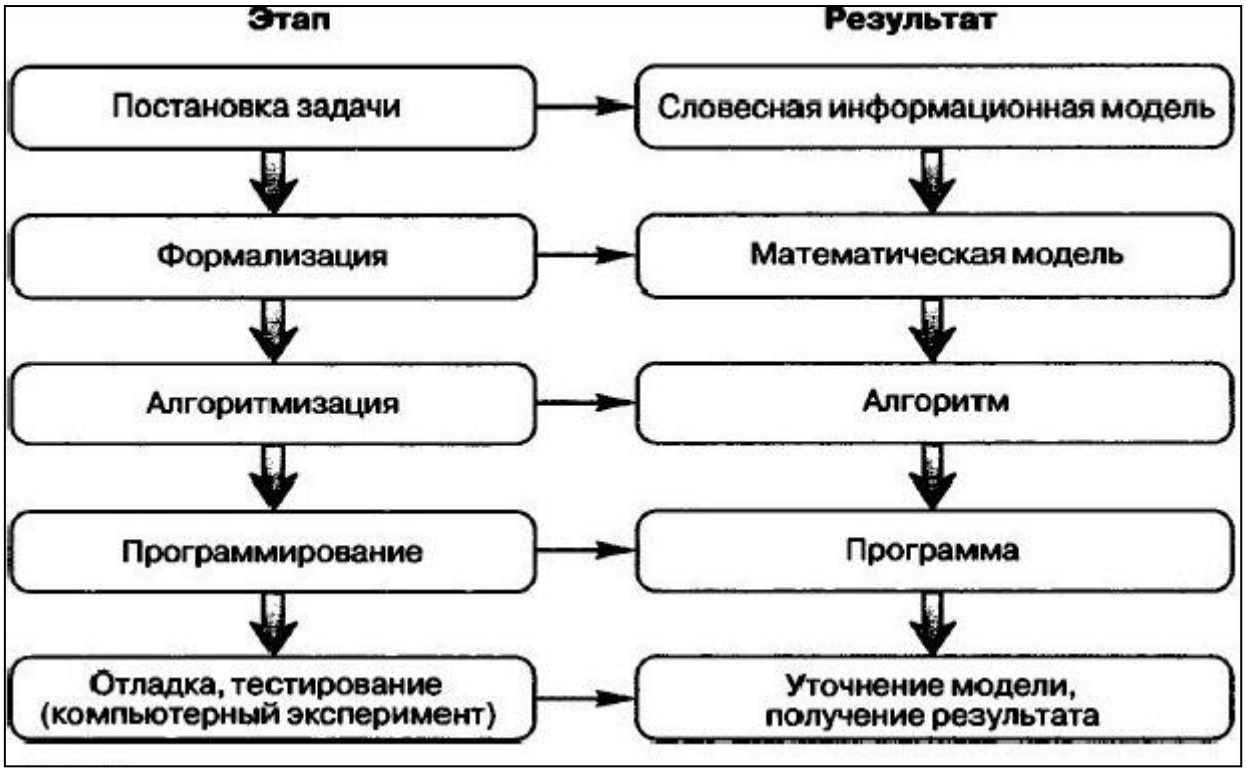
Вопрос 1. Формализация и алгоритмизация поставленных задач

1. Этапы решения задач

Если задуматься, вся жизнь человека – это решения задач. С развитием научно-технического прогресса эти задачи постоянно усложнялись, пока их решение с помощью простых вычислений не стало вызывать явные трудности. Возникновение программирования вызвано потребностью человечества в организации и структурировании больших объемов данных для решения сложных задач.

Программа – это описание множества операций, которые необходимо применить к некоторым данным в определенной последовательности для достижения конкретного результата. Основные отличия языков программирования друг от друга заключаются в разнице допускаемых типов данных, наборе операций, применимых к ним и механизмах управления последовательностью применения операций к данным.

Решение любой задачи в программировании включает в себя 5 основных этапов:



На первом этапе происходит постановка задачи, осознание её условий. При этом должно быть чётко определено, что дано (какие исходные данные известны, какие данные допустимы) и что требуется найти в решаемой задаче. Также должны быть чётко выделены существенные свойства рассматриваемого объекта, указаны связи между исходными данными и результатами.

На втором этапе описательная информационная модель формализуется, т. е. записывается с помощью некоторого формального языка.

Для этого требуется:

- понять, к какому классу принадлежит рассматриваемая задача;

- записать известные связи между исходными данными и результатами с помощью математических соотношений;
- выбрать наиболее подходящий способ для решения задачи.

На **третьем этапе** осуществляется построение алгоритма — чёткой инструкции, задающей необходимую последовательность действий для решения задачи. Алгоритм чаще всего представляется в форме блок-схемы ввиду её наглядности и универсальности.

На **четвёртом этапе** алгоритм записывается на одном из языков программирования.

На **пятом этапе** осуществляется отладка и тестирование программы. Этап отладки и тестирования также называют компьютерным экспериментом.

С прикладной точки зрения данные этапы сводятся к четкой последовательности действий:

1) Разработка технического задания (постановка задачи на содержательном уровне).
2) Формализация задачи (построение математической модели).
3) Выбор (разработка) метода решения задачи.
4) Разработка алгоритма (алгоритмизация).
5) Выбор языка/системы программирования.
6) Написание программы.
7) Отладка и тестирование программы.
8) Оптимизация программы.
9) Документирование программы.
10) Вычисление и обработка результатов.

2. Формализация задач в программировании

Постановка задачи:

- Содержательный анализ существа задачи: определение явления или объекта.
- Изучение общих свойств рассматриваемого физического явления или объекта, выделение основных частей и связей между ними.
- Определение конечной цели и результатов работы.
- Анализ известной информации и определение исходных данных.
- Выработка общего подхода к исследуемой проблеме (выяснение, существует ли решение поставленной задачи и единственно ли оно).
- Анализ возможностей используемой вычислительной среды.

Для изучения сложных явлений и объектов широко используется **метод моделирования**, состоящий в создании и исследовании упрощённых заменителей реальных объектов. Объект-заменитель принято называть моделью, а исходный объект — прототипом или оригиналом.

Модель — это новый объект, который отражает существенные с точки зрения цели моделирования признаки изучаемого предмета, процесса или явления.

Модель может быть:

- натурной (материальной), то есть воспроизводить внешний вид, структуру или поведение объекта (например, глобус);

- и**нформационной,** то есть описывать объект на одном из языков представления (кодирования) информации: словесное описание, таблица, рисунок, схема, чертёж, формула, алгоритм, компьютерная программа и т. д.

Формализация — это замена реального объекта его формальным описанием, т. е. его информационной моделью.

Информационные модели существуют отдельно от объектов моделирования и могут подвергаться обработке независимо от них. Построив информационную модель, человек использует её вместо объекта-оригинала для исследования этого объекта, решения поставленной задачи.

Классификация информационных моделей:

1) **С точки зрения предметной области** выделяются физические, экологические, экономические, социологические и другие модели.

2) **В зависимости от учёта фактора** времени выделяют динамические (изменяющиеся с течением времени) и статические (не изменяющиеся с течением времени) модели.

3) **В зависимости от формы представления информации об объекте моделирования** различают знаковые, образные и смешанные (образно-знаковые) виды информационных моделей.

Знаковые информационные модели строятся с использованием различных естественных и формальных языков (знаковых систем).

Образные информационные модели (рисунки, фотографии и др.) представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации.

В смешанных информационных моделях сочетаются образные и знаковые элементы. Примерами смешанных информационных моделей могут служить географические карты, графики, диаграммы и пр. Во всех этих моделях используются одновременно и графические элементы, и знаки.

Знаковые модели могут быть в форме:

1) Словесные модели — это описания предметов, явлений, событий, процессов на естественных языках (например, учебная и художественная литература).

2) Математические модели строятся с использованием математических понятий и формул и используют совокупность множества формальных языков (алгебраический, геометрический и т.д.). Основным языком информационного моделирования в науке является язык математики.

3) Компьютерные математические модели реализуются с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств. Такие модели используются для процессов, происходящих в окружающем нас мире, описываются очень сложными математическими соотношениями (уравнениями, неравенствами, системами уравнений и неравенств). До появления компьютеров, обладающих высокой скоростью вычислений, у человека не было возможности проводить соответствующие вычисления, на счёт «вручную» уходило очень много времени.

Особый интерес для компьютерного математического моделирования представляют сложные системы, элементы которых могут вести себя случайным образом. Примерами таких систем являются многочисленные **системы массового обслуживания**: билетные кассы, торговые предприятия, ремонтные мастерские, служба «Скорой помощи», транспортные потоки на городских дорогах и многие другие модели. Многим знакома ситуация, когда, придя в кассу, магазин, парикмахерскую, мы застаём там очередь. Приходится либо вставать в очередь и какое-то время ждать, либо уходить, т. е. покидать систему необслу- женным. Возможны случаи, когда заявок на обслуживание в системе мало

или совсем нет; в этом случае она работает с недогрузкой или простаивает. В системах массового обслуживания количество заявок на обслуживание, время ожидания и точное время выполнения заявки заранее предсказать нельзя — это случайные величины.

4) Имитационные модели воспроизводят поведение сложных систем, элементы которых могут вести себя случайным образом.

Имитационное моделирование — это искусственный эксперимент, при котором вместо проведения натурных испытаний с реальным оборудованием проводят опыты с помощью компьютерных моделей. Для получения необходимой информации осуществляется многократный «прогон» моделей со случайными исходными данными, генерируемыми компьютером. В результате образуется такой же набор данных, который можно было бы получить при проведении опытов на реальном оборудовании или в реальной системе. Однако имитационное моделирование на компьютере осуществляется гораздо быстрее и обходится значительно дешевле, чем натурные эксперименты.

Формализация задачи в программировании заключается в построении *математической модели* рассматриваемого объекта, явления или процесса, когда в результате предыдущего анализа существа решаемой задачи устанавливается её принадлежность к одному из известных классов задач и выбирается соответствующий математический аппарат, определяется формат исходных данных и результатов работы, вводится определенная система условных обозначений.

Математическая модель — это система математических соотношений, учитывающих наиболее существенные взаимосвязи в изучаемом классе объектов/явлений и их свойства, в совокупности с определенной областью допустимых значений исходных данных и областью допустимых значений искомых результатов.

После математической постановки задачи отвлекаются от её предметной сущности и оперируют с абстрактными математическими понятиями, величинами, формулами для **выбора метода решения задачи**. При этом нужно учитывать:

- характеристики самого метода, сложность формул и соотношений, связанных с ним;
- необходимую точность вычислений.

3. Разработка алгоритма

На этом этапе осуществляется процесс разработки структуры алгоритма, реализующего выбранный метод решения, и осуществляется запись «придуманной» структуры на языке, «понятном» самому разработчику.

С точки зрения программирования, **алгоритм** – последовательность однозначно определенных команд (шагов, инструкций) исполнителю (человеку или машине), выполнение которых последовательно приводит к достижению поставленной цели.

Простейшими алгоритмами являются изучаемые в школе правила сложения, вычитания, умножения и деления чисел, многие грамматические правила, правила геометрических построений и т. д.

Каждый алгоритм предназначен для определённого исполнителя.

Исполнитель — это некоторый объект (человек, животное, техническое устройство), способный выполнять определённый набор команд.

Различают два типа исполнителей:

- **Формальный исполнитель** одну и ту же команду всегда выполняет одинаково.
- **Неформальный исполнитель** может выполнять команду по-разному.

Формальные исполнители необычайно разнообразны, но для каждого из них можно указать следующие характеристики: круг решаемых задач (назначение), среду, систему команд и режим работы.

Круг решаемых задач — построение цепочек символов, выполнение вычислений, построение рисунков на плоскости и т. д.

Среда исполнителя — **о**бласть, обстановка, условия, в которых действует исполнитель. Исходные данные и результаты любого алгоритма всегда принадлежат среде того исполнителя, для которого предназначен алгоритм.

Система команд исполнителя — совокупность всех команд, которые могут быть выполнены исполнителем, где команда это предписание исполнителю о выполнении отдельного законченного действия. Алгоритм всегда составляется в системе команд того исполнителя, который будет его выполнять.

Режимы работы исполнителя. Для большинства исполнителей предусмотрены режимы непосредственного управления и программного управления. В первом случае исполнитель ожидает команд от человека и каждую поступившую команду немедленно выполняет. Во втором случае исполнителю сначала задаётся полная последовательность команд (программа), а затем он выполняет все эти команды в автоматическом режиме. Ряд исполнителей работает только в одном из названных режимов.

Обязательные свойства алгоритма:

1) Дискретность: путь решения задачи разделён на отдельные шаги (действия). Каждому действию соответствует предписание (команда). Только выполнив одну команду, исполнитель может приступить к выполнению следующей команды.

2) Понятность: алгоритм состоит только из команд, входящих в систему команд исполнителя, т. е. из таких команд, которые исполнитель может воспринять и по которым может выполнить требуемые действия.

3) Определённость: в алгоритме нет команд, смысл которых может быть истолкован исполнителем неоднозначно; недопустимы ситуации, когда после выполнения очередной команды исполнителю неясно, какую команду выполнять следующей. Благодаря этому результат алгоритма однозначно определяется набором исходных данных: если алгоритм несколько раз применяется к одному и тому же набору исходных данных, то на выходе всегда получается один и тот же результат.

4) Результативность: алгоритм всегда должен обеспечивать получение результата после конечного, возможно, очень большого, числа шагов. При этом результатом считается не только обусловленный постановкой задачи ответ, но и вывод о невозможности продолжения по какой-либо причине решения данной задачи.

5) Массовость: алгоритм должен обеспечивать возможность его применения для решения любой задачи из некоторого класса задач. Например, алгоритм нахождения корней квадратного уравнения должен быть применим к любому квадратному уравнению, алгоритм перехода улицы должен быть применим в любом месте улицы, алгоритм приготовления лекарства должен быть применим для приготовления любого его количества и т. д.

Построение алгоритма включает в себя:

- разложение вычислительного процесса решения задачи на возможные составные части;
- установление порядка их следования;
- описание содержания каждой такой части в той или иной форме;
- последующую проверку, которая должна показать, обеспечивается ли реализация выбранного метода.

В процессе разработки алгоритм проходит несколько этапов детализации:

- Первоначально составляется **укрупненная схема алгоритма**, в которой отражаются наиболее важные и существенные связи между исследуемыми процессами (или частями процесса). Ориентируясь на крупноблочную структуру алгоритма, можно быстрее и проще разработать несколько различных его вариантов, провести их анализ, оценку и выбрать наилучший (оптимальный).

Каждый элемент крупноблочной схемы алгоритма должен быть максимально самостоятельным и логически завершённым в такой степени, чтобы дальнейшую его детализацию можно было выполнять независимо от детализации остальных элементов.

- На последующих этапах **детализируются** выделенные на предыдущих этапах части вычислительного процесса, имеющие некоторое самостоятельное значение.

На каждом этапе детализации выполняется многократная проверка и исправление схемы алгоритма. Подобный подход позволяет во многом избежать возможных ошибочных решений.

Алгоритмы широко используются для автоматизации процессов.

Разработка алгоритма — как правило, трудоёмкая задача, требующая от человека глубоких знаний, изобретательности и больших временных затрат. Зато решение задачи по готовому алгоритму требует от исполнителя только строгого следования заданным предписаниям. Он может не вникать в смысл того, что делает, и не рассуждать, почему он поступает так, а не иначе, т. е. он может действовать формально.

Способность исполнителя действовать формально обеспечивает возможность автоматизации деятельности человека. Для этого:

1) процесс решения задачи представляется в виде последовательности простейших операций;

2) создаётся машина (автоматическое устройство), способная выполнять эти операции в последовательности, заданной в алгоритме;

3) человек освобождается от рутинной деятельности, выполнение алгоритма поручается автоматическому устройству.

Способы описания алгоритмов отличаются по простоте и наглядности. В практике программирования наибольшее распространение получили:

- словесная запись алгоритмов;
- схемы алгоритмов (блок-схемы);
- структурограммы (диаграммы Насси — Шнейдермана).

Словесное описание — запись алгоритма в виде набора высказываний на обычном разговорном языке. Имеет минимум ограничении и является наименее формализованным. Однако все разговорные языки обладают неоднозначностью, поэтому могут возникнуть различные толкования текста алгоритма, заданного таким образом. Алгоритм в словесной форме может оказаться очень объёмным и трудным для восприятия.

Пример 1. Словесное описание алгоритма нахождения наибольшего общего делителя (НОД) пары натуральных чисел (алгоритм Евклида).

Чтобы найти НОД двух чисел, составьте таблицу из двух столбцов и назовите столбцы X и Y. Запишите первое из заданных чисел в столбец X, а второе — в столбец Y. Если данные числа не равны, замените большее из них на результат вычитания из большего числа меньшего. Повторяйте такие замены до тех пор, пока числа не окажутся равными, после чего число из столбца X считайте искомым результатом.

Построчная запись — это запись на естественном языке, но с соблюдением некоторых дополнительных правил:

- каждое предписание записывается с новой строки;
- предписания (шаги) алгоритма нумеруются;
- исполнение алгоритма происходит в порядке возрастания номеров шагов, начиная с первого (если не встречается никаких специальных указаний).

Кроме слов естественного языка предписания могут содержать математические выражения и формулы.

Пример 2. Построчная запись алгоритма Евклида.

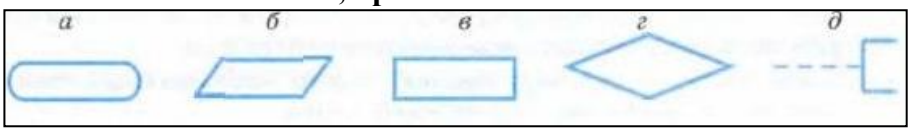
1. Обозначить первое из заданных чисел X, второе обозначить Y.
2. Если X = Y, то перейти к п. 8.
3. Если X > Y, то перейти к п. 4, иначе перейти к п. 6.
4. Заменить X на X — Y.
5. Перейти к п. 2.
6. Заменить Y на Y — X.
7. Перейти к п. 2.
8. Считать X искомым результатом.

Построчная запись алгоритма позволяет избежать ряда неопределённостей, её восприятие не требует дополнительных знаний. Однако ее запись требует от человека большого внимания.

Наилучшей наглядностью обладают графические способы записи алгоритмов; самый распространённый среди них — блок-схема.

Блок-схема представляет собой графический документ, дающий представление о порядке работы алгоритма. Здесь предписания изображаются с помощью различных геометрических фигур, а последовательность выполнения шагов указывается с помощью линий, соединяющих эти фигуры. Направления линий связи *слева направо* и *сверху вниз* считаются стандартными, соответствующие им линии связи можно изображать *без стрелок*. Линии связи *справа налево* и *снизу вверх* изображаются *со стрелками*.

Условные обозначения, применяемые в блок-схемах:.



Выполнение алгоритма всегда начинается с блока начала и оканчивается при переходе на блок конца (рис. 2, а). Из начального блока выходит одна линия связи; в конечный блок входит одна линия связи.

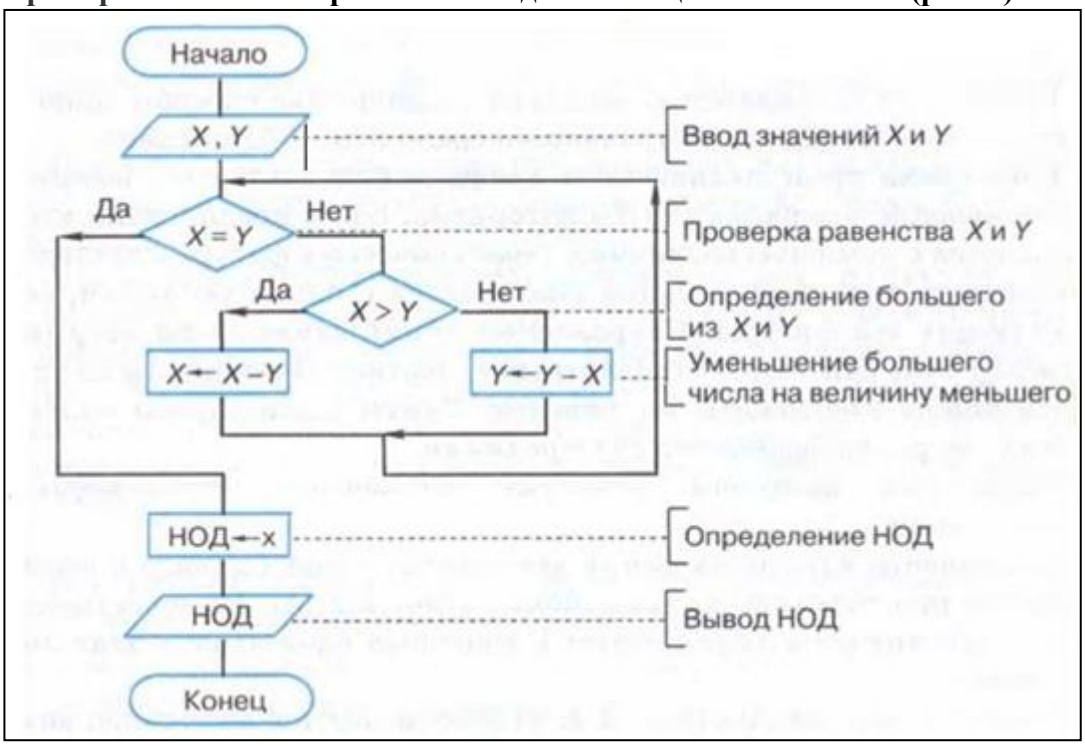
Внутри блока данных (рис. 2, б) перечисляются величины, значения которых должны быть введены (исходные данные) или выведены (результаты) в данном месте схемы. В блок данных входит одна линия связи, и из блока исходит одна линия связи.

В блоке обработки данных (рис. 2, в) содержится описание тех действий, которые должны быть выполнены при переходе на этот блок (выполнение определённой операции или группы операций, приводящее к изменению значения, формы или размещения информации). В блок обработки данных входит одна линия связи, и из блока исходит одна линия связи.

Проверка условия изображается с помощью блока принятия решения, внутри которого записывается это условие (рис. 2, г). В блок принятия решения входит одна линия, а выходят две линии, около которых записываются результаты проверки условия.

Комментарии (рис. 2, д) используются для добавления пояснительных записей, делающих блок-схему более понятной.

Пример 3. Запись алгоритма Евклида с помощью блок-схемы (рис. 3).



Алгоритмы могут быть линейными, разветвляющимися, циклическими, состоять из нескольких модулей и т.д.

Создание детальной блок-схемы сложного алгоритма — трудоёмкая задача. Кроме того, блок-схема, не умещающаяся на одном стандартном листе, теряет своё основное преимущество — наглядность. При разработке сложных алгоритмов блок-схемы удобно использовать в качестве средства для наглядного представлениям решения задачи в общем виде.

Диаграмма Насси — Шнейдермана (англ. *Nassi — Shneiderman diagram*) — это графический способ представления структурированных алгоритмов и программ, разработанный в 1972 двумя американскими аспирантами.

Бен Шнейдерман решил, что для записи структурированных алгоритмов не нужны используемые в блок-схемах стрелки и вместе с Айзеком Насси подробно проработал разные способы изображения основных структур управления (последовательностей, ветвлений и циклов).

Диаграммы Насси — Шнейдермана имеют ряд преимуществ перед блок-схемами при разработке структурированных алгоритмов и программ:

- Запись является более компактной (в первую очередь за счёт отсутствия стрелок между элементами).
- Точно соблюдаются принципы структурного программирования (при использовании блок-схем можно случайно получить неструктурированный алгоритм, если быть невнимательным).

- Диаграммы Насси — Шнейдермана удобнее использовать для пошаговой детализации задачи, так как они строятся именно по этому принципу.

Изначально диаграмма представляет собой один прямоугольник (исходная задача), затем в нём рисуется некоторая структура управления, в которой имеется несколько прямоугольников (подзадач исходной задачи), и далее с каждым прямоугольником (подзадачей) может быть проделана та же операция.

Все условные элементы диаграммы Насси — Шнейдермана имеют прямоугольную форму и различаются только внутренним содержимым.

Простое выполняемое действие (в том числе команда языка в программе или подзадача в алгоритме) изображается как прямоугольник, в котором записывается обозначение действия (команда) либо формулировка подзадачи.

Пример записи подзадачи:

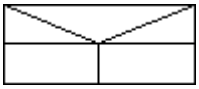
Найти наименьший элемент в массиве

При записи **структуры последовательного выполнения** элементы изображаются вертикально один за другим. При этом все элементы последовательности должны иметь одинаковую ширину — за счёт этого вся последовательность тоже имеет прямоугольную форму.

Пример записи последовательности:

Открыть файл на чтение
Прочитать первую строку файла
Вывести прочитанное значение на экран
Закрыть файл

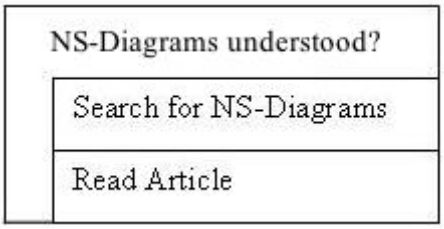
Структура простого ветвления изображается в виде прямоугольника, разделённого горизонтальной чертой на две части. В верхней части располагается заголовок ветвления, а в нижней — две ветки ветвления, разделённые вертикальной чертой. В заголовке рисуются две линии, ведущие от верхних углов к началу линии, разделяющей ветви. В получившемся вверху треугольнике записывается условие ветвления, в двух нижних треугольниках над ветвями подписываются значения условия, соответствующие этим ветвям, например «истина» и «ложь», или «да» и «нет».



Структура многовариантного выбора изображается похоже на структуру простого ветвления, только основная ветвь и треугольник над ней делятся на много частей вертикальными линиями. В верхнем треугольнике записывается выражение-переключатель, над ветвями записываются соответствующие значения переключателя.

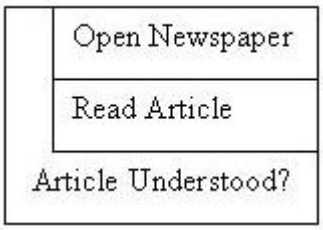
Структура повтора с условием в начале изображается как прямоугольник, внутри которого в правой нижней части нарисован ещё один прямоугольник. Над внутренним прямоугольником записывается заголовок цикла, а внутри него — тело цикла.

Пример записи цикла с предусловием *(в данном примере тело цикла является последовательностью из двух действий):*



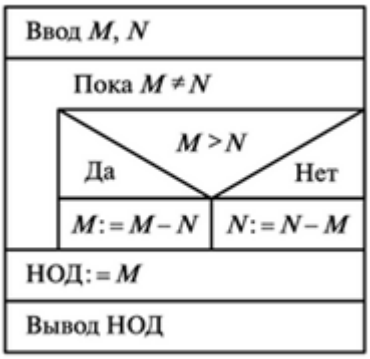
Повтор с постусловием отличается от цикла с предусловием только тем, что внутренний прямоугольник рисуется в правой верхней части внешнего, а заголовок записывается снизу.

Пример записи цикла с постусловием *(в данном примере тело цикла является последовательностью из двух действий):*



При повторе со счётчиком внутренний прямоугольник рисуется в правой части и не касается верха и низа внешнего прямоугольника. Условие цикла записывается сверху.

Пример: запись алгоритма Евклида с помощью структурограммы:



Литература:

1. Босова Л.Л., Босова А.Ю. Информатика. 9 класс: учебник / Л. Л. Босова, А. Ю. Босова. — 6-е изд. — М. : БИНОМ. Лаборатория знаний, 2016. — 208 с. (стр. 5-16, 58-59)
2. Лягинова О.Ю., Можаева М.Г. Этапы решения задач на ЭВМ: онлайн-презентация / Кафедра математики и информатики ЧГУ / https://en.ppt-online.org/593552 (стр. 1-15)
3. Суриков В.Н.,Кудрявцев А.С,Петров Г.А.,Хардиков Е.В Основы алгоритмизации инженерных задач: учебное пособие/ГОУВПО СПБГТУ РП.СПб.,2008. - 158 С. (стр. 7-57)
4. Босова Л.Л., Босова А.Ю. Информатика. 8 класс: учебник / Л. Л. Босова, А. Ю. Босова. — М. : БИНОМ. Лаборатория знаний, 2013. — 155 с. (стр. 46-60)
5. https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%9D%D0%B0%D1%81%D1%81%D0%B8_%E2%80%94_%D0%A8%D0%BD%D0%B5%D0%B9%D0%B4%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%B0 https://ru.wikipedia.org/wiki Диаграмма_Насси_—_Шнейдермана