Основы программирования

В результате изучения дисциплины студент должен:

- **Знать** понятие алгоритма, основные свойства алгоритма, способы представления алгоритмов, основные алгоритмические структуры (линейную, ветвление, циклическую), виды циклов (с параметром, с предусловием, с постусловием), историю, классификацию, типы языков программирования, современные методы и средства разработки алгоритмов и программ, этапы решения задач, этапы производства программного продукта, способы отладки, базовые понятия и сведения о языке VBA, методы и особенности программирования в языке VBA, типы данных языка VBA, операторы языков, методы реализации основных алгоритмических структур, описание и основные методы работы (поиск, сортировка) с массивами в VBA, основные процедуры и функции работы со строковыми данными.

- **Уметь** составлять алгоритм решения задачи, представлять его в виде блок-схем и на VBA, создавать, отлаживать и тестировать программы на VBA, использовать процедуры и функции в процессе программирования, рекурсивные алгоритмы решения задач, реализовывать основные алгоритмические структуры с помощью языка VBA, осуществлять поиск в массивах разной размерности наиболее эффективным методом, осуществлять сортировку массива, работать со строковыми данными.

- **Владеть** навыками алгоритмизации задач, навыками выбора наиболее эффективных способов решения задачи, навыками применения полученных знаний в профессиональной деятельности, основными навыками работы с современными программами.

Лабораторные работы: Критерии оценки студентов по результатам контрольных

1 неделя (20%), 2 неделя (40%), 3 неделя (40%)

Весовые коэффициенты по видам работ:
лекции 20%, лабораторные 50%, другие (тест, сертификат) 25%

Лекции

В процентах по количеству посещенных лекций (0 – 50%).
По результатам тестирования:
40 – 50 % (отлично) тест написан на «отлично»;
30 – 40 % (хорошо) тест написан на «хорошо» и «отлично»;
20 – 30 % (удовлетворительно) тест написан не менее чем на «удовлетворительно»;
менее 20% (неудовлетворительно) тест написан на «неудовлетворительно»

Лабораторные работы

Оценивается в процентах от выполненных и защищенных лабораторных работ по темам.
90 – 100 % (отлично) - выполнены и защищены все лаб. работы;
75 – 89 % (хорошо) - выполнены все лабораторные работы, одна защищена;
50 – 74 % (удовлетворительно) - выполнены все лаб. работы;
0 – 49 % (неудовлетворительно) – выполнены не все лаб. работы

Отчет по лабораторной работе:

- Титульная страница
- Тип
- Задача: *Текст задачи*
- Постановка задачи
- Блок-схема
- Решение на языке (листинг)

Основы программирования

Лекция 1.

Структура программного обеспечения

Под программным обеспечением понимается совокупность программ, выполняемых вычислительной системой.

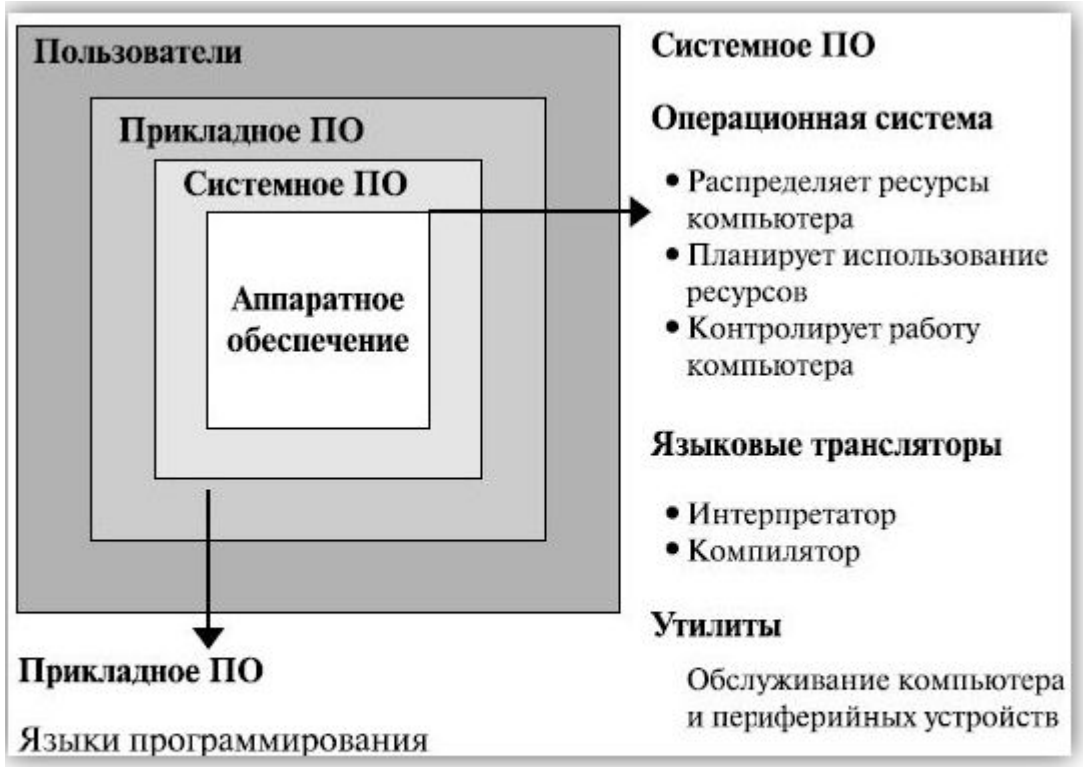
К программному обеспечению относится также вся область деятельности по проектированию и разработке программного обеспечения:

- ✔технология проектирования программ;
- ✔методы тестирования программ;
- ✔методы доказательства правильности программ;
- ✔анализ качества работы программ;
- ✔документирование программ;
- ✔разработка и использование программных средств, облегчающих процесс проектирования программного обеспечения.

Структура программного обеспечения

Существует два основных типа программного обеспечения: системное (общее) и прикладное (специальное)

Каждый тип программного обеспечения выполняет различные **функции:**



✔Системное программное обеспечение – это набор программ, которые управляют компонентами компьютера, такими как *процессор*, коммуникационные и периферийные устройства.

✔ К прикладному программному обеспечению относятся программы, написанные для пользователей или самими пользователями, для задания компьютеру конкретной работы.

Этапы подготовки и решения задач на компьютере

Решение задачи на компьютере - это процесс автоматического преобразования информации в соответствии с поставленной целью.

Под процессом решения задачи на ЭВМ надо понимать **совместную деятельность человека и компьютера**. Этот процесс остается пока достаточно сложным и трудоемким, поэтому представляется в виде нескольких последовательных этапов. При этом на долю человека приходится творческая деятельность, а на долю машины - автоматическая обработка информации в соответствии с заданным ей алгоритмом.

Этапы подготовки и решения задач на компьютере

Компьютер предназначен для решения разнообразных задач: научно-технических, инженерных, разработки системного программного обеспечения, обучения, управления производственными процессами и т.д.

В процессе подготовки и решения на компьютере научно-технических задач можно выделить следующие этапы:

1. Постановка задачи
2. Формализация
3. Построение алгоритма
4. Запись алгоритма на языке программирования
5. Отладка и тестирование
6. Анализ полученных результатов

Непосредственно к программированию относятся этапы 3, 4, 5.

Этапы подготовки и решения задач на компьютере

На этапе постановки задачи должно быть четко сформулировано, что дано, и что требуется найти.

Здесь очень важно определить полный набор исходных данных, необходимых для получения решения:

✔определить цель решения задачи;
✔ определить необходимый объем информации;
✔дать точную формулировку задачи;
✔предложить идею решения задачи;
✔описать исходные данные и способы их хранения;
✔определить форму выдачи результатов.

Этапы подготовки и решения задач на компьютере

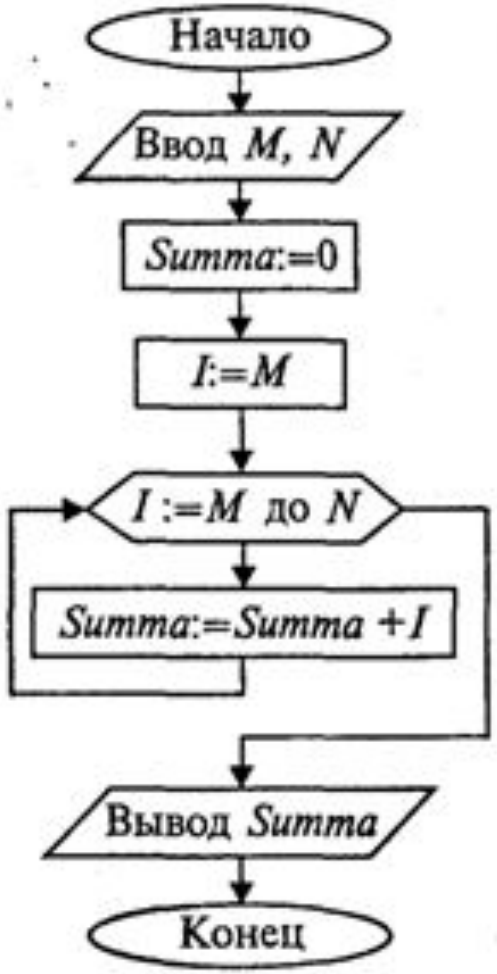
Второй этап – **формализация задачи.**
Чаще всего задача переводится на язык математических формул, уравнений, отношений. Если решение требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей математической модели.

Составляющая этапа формализации - **выбор и обоснование метода решения:** модель решения задачи реализуется на основе конкретных приемов и методов решения. В большинстве случаев математическое описание задачи трудно перевести на машинный язык. Выбор и использование метода решения позволяет свести решение задачи к конкретному набору машинных команд. При обосновании метода решения рассматриваются вопросы влияния различных факторов и условий на конечный результат, в том числе на точность вычислений, время решения задачи на компьютере, требуемый объем памяти и др.

Этапы подготовки и решения задач на компьютере

Третий этап – **построение алгоритма.**

На данном этапе составляется алгоритм решения задачи, в соответствии с выбранным методом решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные блоки, определяется последовательность выполнения этих блоков.



Этапы подготовки и решения задач на компьютере

На этапе **записи алгоритма на языке программирования** алгоритм решения переводится на конкретный язык программирования.

Отладка и тестирование: процесс устранения синтаксических и логических ошибок в программе. В процессе трансляции программы с помощью синтаксического и семантического контроля выявляются недопустимые конструкции и символы (или сочетания символов) для данного языка программирования. Компьютер выдает сообщение об ошибках в форме, соответствующей этому языку. Затем проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными.

Анализ полученных результатов. Первоначально выполняется многократное решение задачи на компьютере для различных наборов исходных данных. Получаемые результаты анализируются специалистом, поставившим задачу. Разработанная программа поставляется заказчику в виде готовой к исполнению машинной программы.

Что такое алгоритм?

Название **"алгоритм"** произошло от латинской формы имени величайшего среднеазиатского математика **Мухаммеда ибн Муса ал-Хорезми** (Alhorithmi), жившего в 783—850 гг.

В своей книге "Об индийском счете" он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними "столбиком", знакомые теперь каждому школьнику.

В XII веке эта книга была переведена на латынь и получила широкое распространение в Европе.

Интуитивное определение понятия

Алгоритм – это четко определенный план действий для исполнителя.

Алгоритм – совокупность четко определенных правил для решения задачи за конечное число шагов.

Примеры алгоритмов : рецепт приготовления блюда, решение квадратного уравнения, правила по русскому языку и др.

Алгоритм —понятное и точное предписание исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов, приводящее к определенному результату.

АЛГОРИТМИЗАЦИЯ – процесс разработки алгоритма (плана действий) для решения задачи.

Понятие алгоритма и его свойства

На этапе **записи алгоритма на языке программирования** алгоритм решения переводится на конкретный язык программирования.

Свойство 1º. Дискретность. Вычислительный процесс идет в дискретном времени. На каждом такте выполняется элементарный шаг алгоритма. Сам алгоритм состоит из конечного числа таких шагов. Исходные данные определяют начальное состояние памяти, а выполнение каждого шага приводит к изменению состояния памяти. Каждый шаг меняет прежнее состояние на новое.

Свойство 2º. Детерминированность. Процедура преобразований данных на каждом шаге работы алгоритма однозначно определена. Такой шаг содержит информацию о том, какое элементарное действие и над какими данными осуществляется в данном шаге, куда записать результат и на какой шаг следует перейти после выполнения этого действия.

Свойство 3º. Результативность (конечность). Алгоритм направлен на получение определенного результата. За конечное число тактов вычислительный процесс должен быть завершен – должен произойти *результативный останов*.

Свойство 4º. Массовость. Исходные величины могут варьироваться в известных пределах, а сам алгоритм – это процедура многократного применения.

Представление понятия алгоритма на интуитивно-содержательном уровне с учетом перечисленных выше свойств является достаточным для практического программирования, в рамках которого создаются теории для совершенствования технологий программирования.

Исполнитель алгоритма — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

- среда;
- элементарные действия;
- система команд;
- отказы.

Среда или обстановка

Среда (или обстановка) — это "место обитания" исполнителя.

Система команд

Каждый исполнитель может выполнять команды только из некоторого строго заданного списка — **системы команд исполнителя.**

Для каждой команды должны быть заданы **условия применимости** (в каких состояниях среды может быть выполнена команда) и описаны **результаты выполнения команды**.

После вызова команды исполнитель совершает соответствующее **элементарное действие**.

Отказы

Отказы исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды.

Программа

Программа – это

- алгоритм, записанный на каком-либо языке программирования
- набор команд для компьютера

Команда – это описание действий, которые должен выполнить компьютер.

- откуда взять исходные данные?
- что нужно с ними сделать?

Оператор – это команда языка программирования высокого уровня.

Формы представления алгоритма

- **словесная** (запись на естественном языке);
- **графическая** (блок-схема, структурограмма);
- **псевдокоды** (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
- **программная** (тексты на языках программирования).

Словесная
(запись на естественном языке)

- представляет собой описание последовательных этапов обработки данных.

Алгоритм задается в произвольном изложении на естественном языке.

Словесный способ не имеет широкого распространения, так как такие описания:

- строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Алгоритм Эвклида

Например. Записать алгоритм нахождения **наибольшего общего делителя (НОД)** двух натуральных чисел (алгоритм Эвклида).

Алгоритм может быть следующим:

- задать два числа;
- если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
- определить большее из чисел;
- заменить большее из чисел разностью большего и меньшего из чисел;
- повторить алгоритм с шага 2.

Графический способ

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Блок-схема

Графическое представление называется схемой алгоритма или **блок-схемой.**

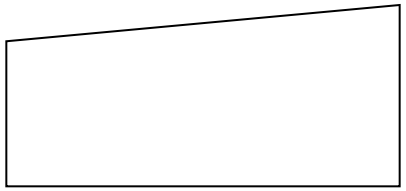
В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде **блочного символа**.

Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий.

Основные блоки блок-схемы

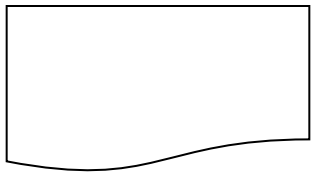
Блок начала и конца блок-схемы

Ввод/вывод данных с/на любой носитель

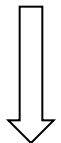


Ввод данных с клавиатуры

Математический блок

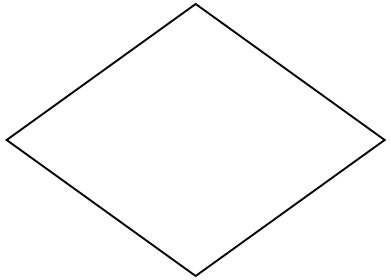


Блок вывода на печать



Направление вычисления

Основные блоки блок-схемы



Логический блок

Вызов процедуры или функции

Для циклов с параметром

Для описательных комментариев или пояснительных записей в целях объяснения или примечаний

Псевдокод

представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов.

В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя.

В псевдокоде обычно **имеются некоторые конструкции, присущие формальным языкам (служебные слова)**, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке.

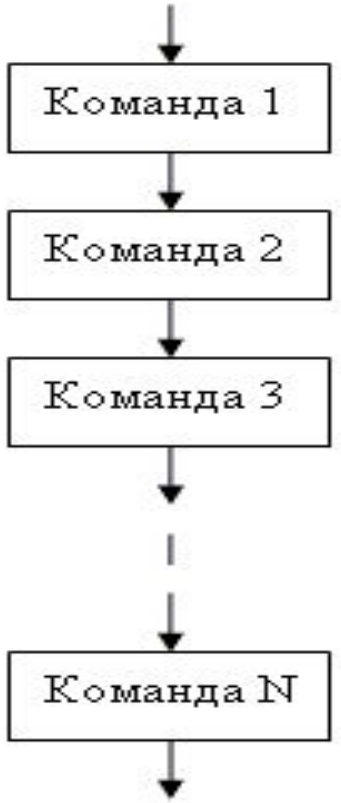
Основные алгоритмические структуры и их суперпозиции

- Линейный (последовательный)
- Разветвляющийся (ветвление)
- Циклический (цикл).

Любой сколь угодно сложный алгоритм может быть разработан на основе трёх типовых структур: следования, ветвления и повторения (циклов). При этом структуры могут располагаться последовательно друг за другом или вкладываться друг в друга.

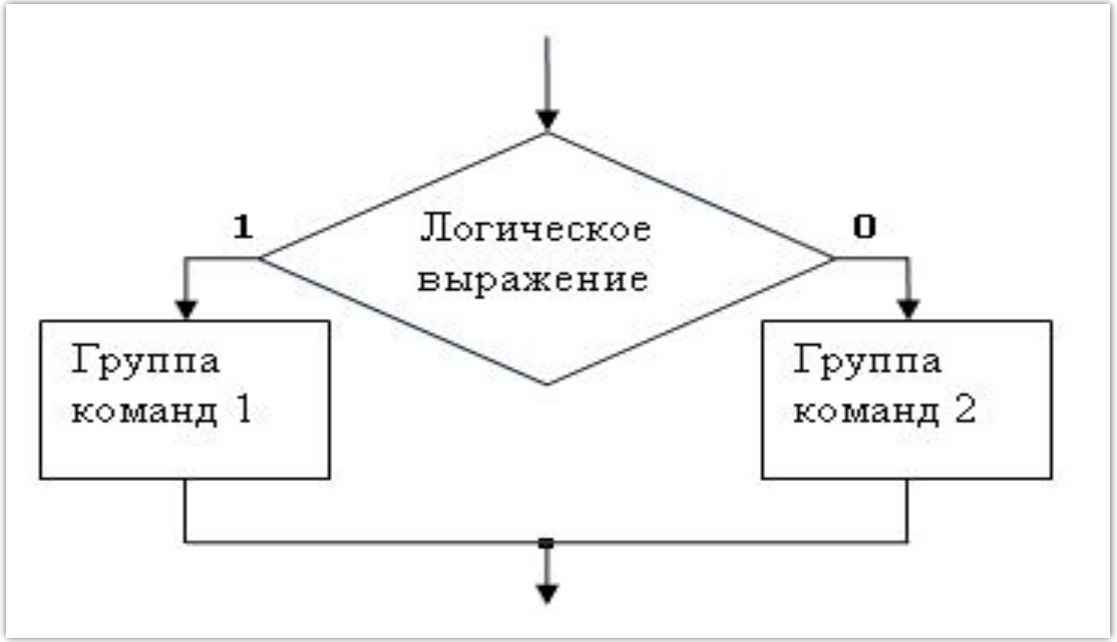
Линейная структура (следование).

- Наиболее простой алгоритмической структурой является линейная. В ней все операции выполняются один раз в том порядке, в котором они записаны.



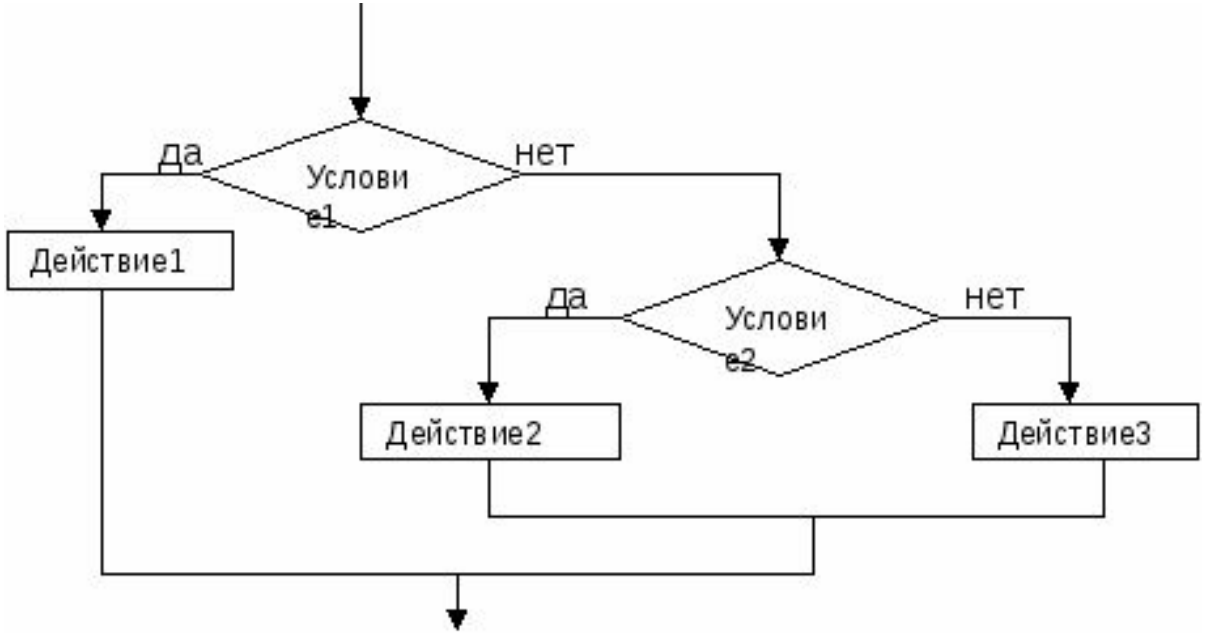
Ветвление

В полном ветвлении предусмотрено два варианта действий исполнителя в зависимости от значения логического выражения (условия). Если условие истинно, то выполняться будет только первая ветвь, иначе только вторая ветвь.



Вторая ветвь может быть пустой. Такая структура называется неполным ветвлением или обходом.

Из нескольких ветвлений можно сконструировать структуру «выбор» (**множественное ветвление**), которая будет выбирать не из двух, а из большего количества вариантов действий исполнителя, зависящих от нескольких условий.



Существенно, что выполняется только одна ветвь - в такой структуре значение приобретает порядок следования условий: если выполняются несколько условий, то сработает только одно из них - первое сверху.

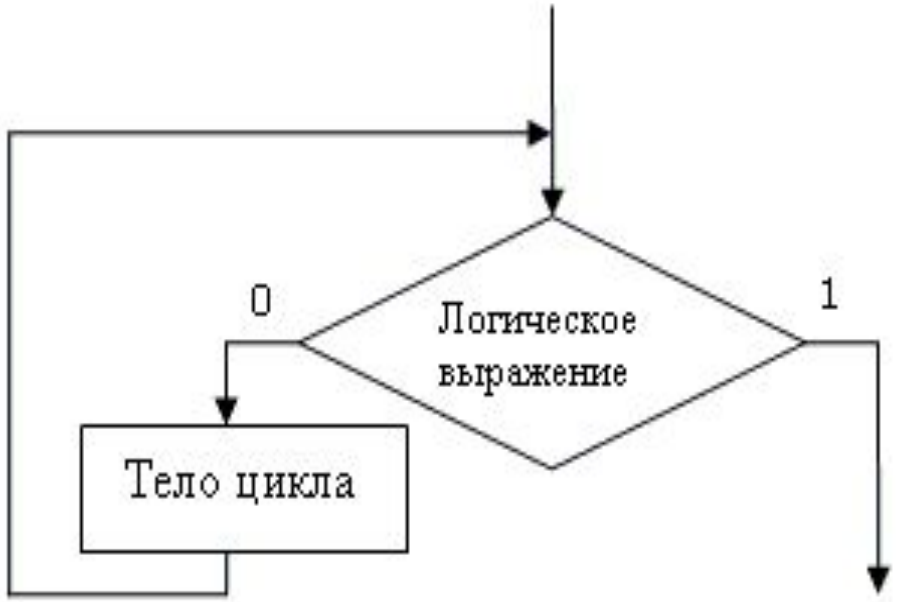
Цикл (повторение)

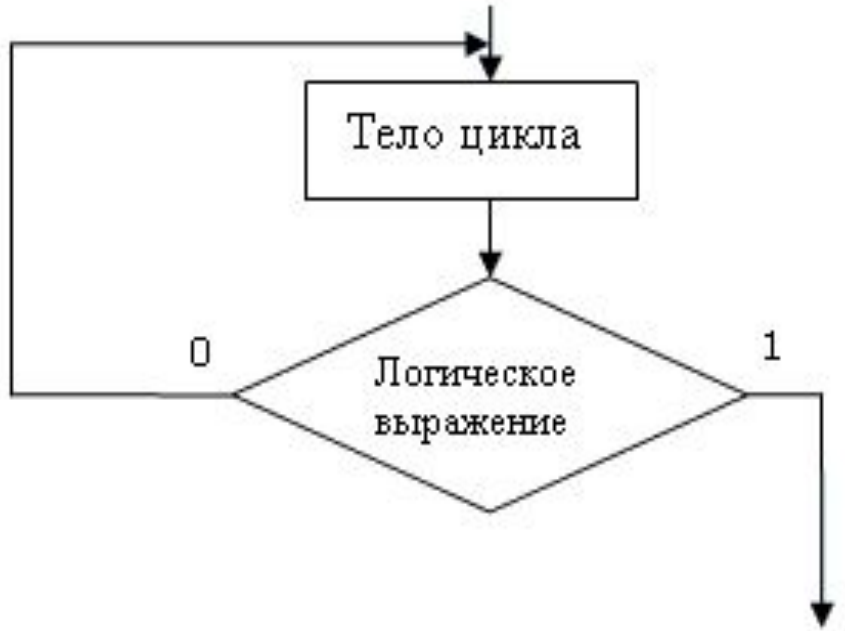
Цикл позволяет организовать многократное повторение одной и той же последовательности команд - она называется телом цикла. В различных видах циклических алгоритмов количество повторений может зависеть от значения логического выражения (условия) или может быть жестко задано в самой структуре.

Различают циклы: «до», «пока», циклы со счётчиком. В циклах «до» и «пока» логическое выражение (условие) может предшествовать телу цикла (цикл с предусловием) или завершать цикл (цикл с постусловием).

Цикл (повторение)

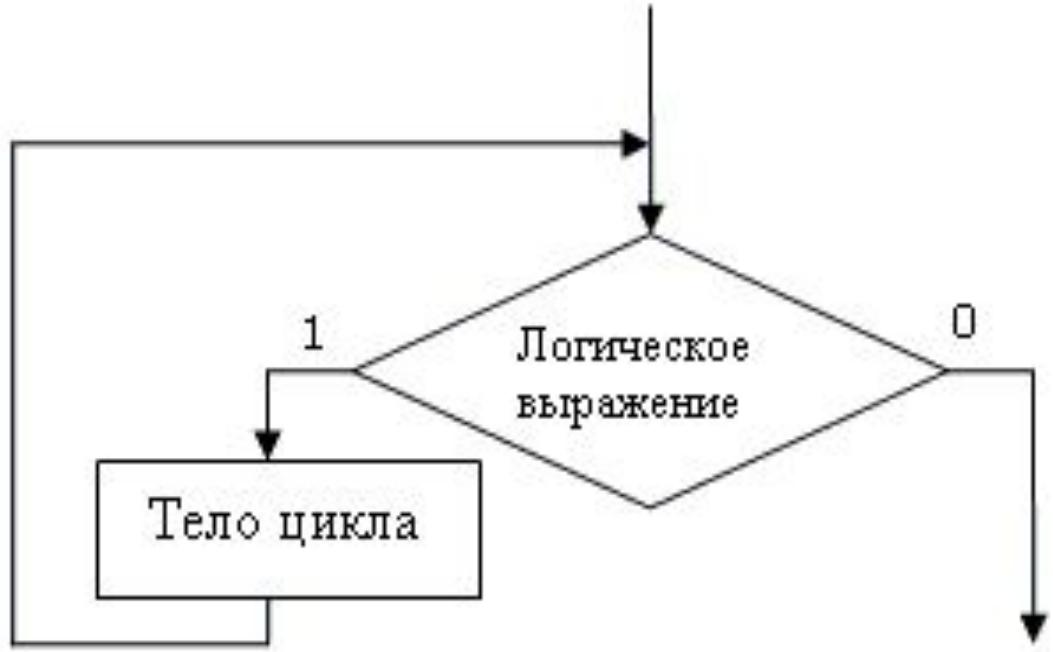
Циклы «до» - повторение тела цикла до выполнения условия:





Цикл (повторение)

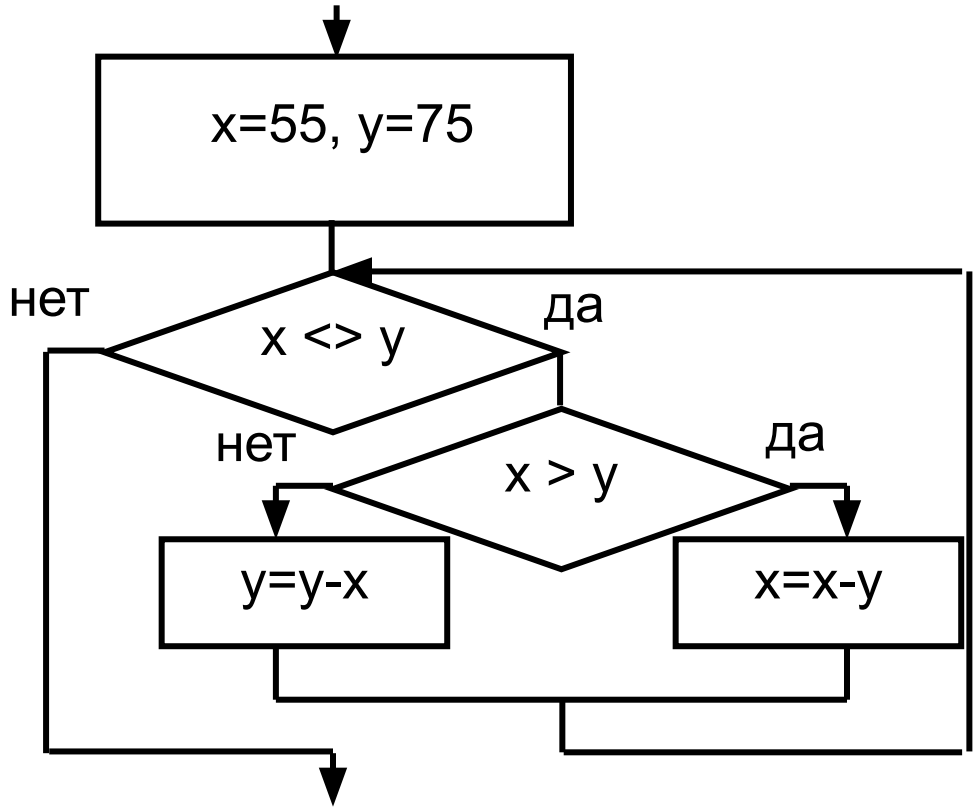
Циклы «пока» - повторение тела цикла пока условие выполняется (истинно):



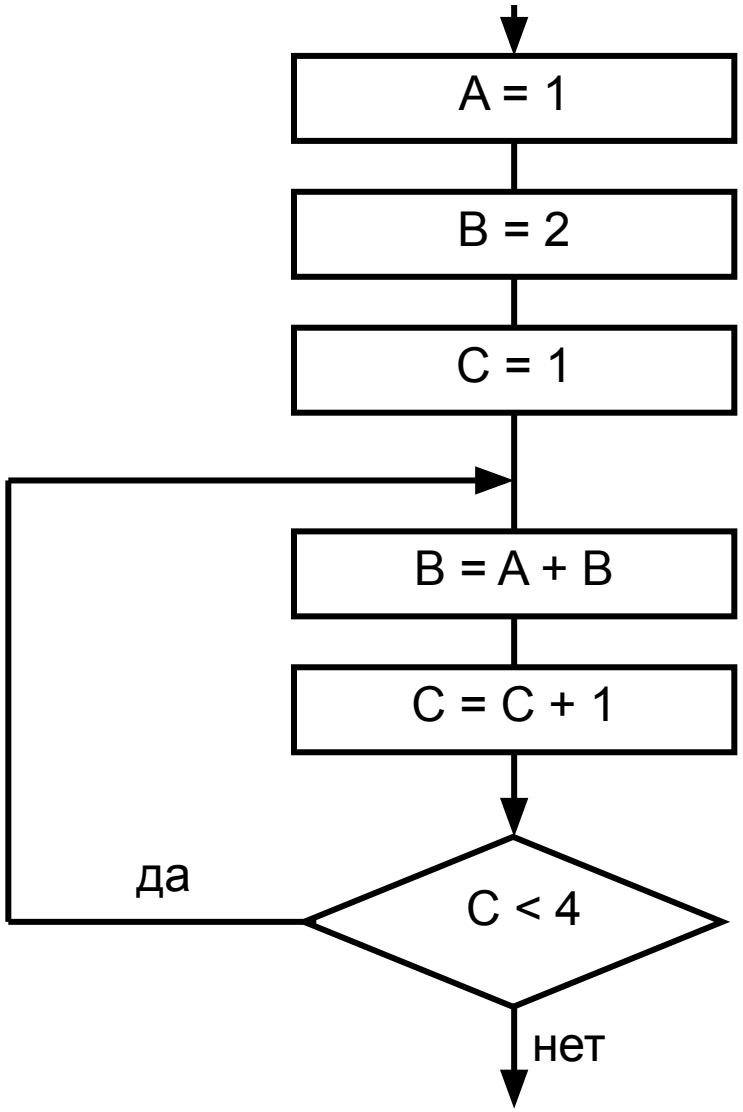
Циклы со счётчиком (с параметром) – повторение тела цикла заданное число раз:

Задания для самостоятельной работы

1) Определите значение целочисленной переменной x после выполнения следующего фрагмента алгоритма:



2) Определите значение переменной B :



Чем отличается программный способ записи алгоритмов от других?

- При записи алгоритма в словесной форме, в виде блок-схемы или на псевдокоде допускается определенный произвол при изображении команд. Вместе с тем такая запись точна настолько, что позволяет человеку понять суть дела и исполнить алгоритм.
- Однако на практике в качестве исполнителей алгоритмов используются компьютеры. Поэтому алгоритм, предназначенный для исполнения на компьютере, должен быть записан на понятном ему языке. И здесь на первый план выдвигается необходимость точной записи команд, не оставляющей места для произвольного толкования их исполнителем.
- Следовательно, язык для записи алгоритмов должен быть формализован. Такой язык принято называть языком программирования, а запись алгоритма на этом языке — программой для компьютера.

Преимущества алгоритмических языков перед машинными

- **алфавит алгоритмического языка значительно шире алфавита машинного языка**, что существенно повышает наглядность текста программы;
- **набор операций,** допустимых для использования, **не зависит от набора машинных операций**, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;
- **формат предложений** достаточно **гибок и удобен** для использования, что позволяет с помощью одного предложения задать достаточно содержательный этап обработки данных;
- требуемые операции задаются с помощью **общепринятых математических обозначений**;
- **данным в алгоритмических языках присваиваются индивидуальные имена**, выбираемые программистом;
- в языке может быть предусмотрен значительно **более широкий набор типов данных** по сравнению с набором машинных типов данных.
- Таким образом, алгоритмические языки в значительной мере являются **машинно-независимыми**. Они облегчают **работу программиста** и **повышают надежность** создаваемых **программ**.

Из истории развития программирования

Эпоха прямого программирования (50-е годы)

- Программирование в кодах : 001 1200 1400 1340

  (1200)+(1400)=>(1340)

Эпоха трансляторов (с середины 50-х годов)

- Программирование низкого уровня
  - Мнемокод, Автокод: c a, b, c (c := a + b)
  - Ассемблер (Assembler): ADD A, B (A := A + B)
- Программирование на языках неструктурного типа:
  - Фортран ***(Fortran – Formula Tranclation) –формульный транслятор***
  - Алгол *(Algol – Algorithmic language) – алгоритмический язык*
  - Бейсик *(Basic – основной, базисный, учебный)*
  - Кобол (Cobol) и другие

Из истории развития программирования

- Структурное программирование *(С 1970 года)*

Паскаль *(Pascal – его создатель Никлаус Вирт)*

Программирование с использованием библиотечных модулей

Объектно - ориентированное программирование

Системы программирования:

оболочка + язык программирования

Турбо оболочки (TURBO)

Визуальные оболочки (VISUAL)

Компоненты языка

- **Алфавит — это фиксированный для данного языка набор основных символов,** т.е. "букв алфавита", из которых должен состоять любой текст на этом языке — никакие другие символы в тексте не допускаются.
- **Синтаксис — это правила построения фраз**, позволяющие определить, правильно или неправильно написана та или иная фраза. Точнее говоря, **синтаксис языка представляет собой набор правил, устанавливающих, какие комбинации символов являются осмысленными предложениями на этом языке.**
- **Семантика** определяет смысловое значение предложений языка. Являясь системой правил истолкования отдельных языковых конструкций, **семантика устанавливает, какие последовательности действий описываются теми или иными фразами языка** и, в конечном итоге, **какой алгоритм определен данным текстом на алгоритмическом языке.**

Основные понятия в алгоритмических языках

1. **Имена** (идентификаторы) — **употребляются для обозначения объектов программы** (переменных, массивов, функций и др.).
2. **Операции**. Типы операций:

- **арифметические** операции + , — , * , / и др. ;
- **логические** операции **и** , **или** , **не** ;
- операции **отношения** < , > , <= , >= , = , <> ;
- операция **сцепки** (иначе, "присоединения", "конкатенации") символьных значений друг с другом с образованием одной длинной строки; изображается знаком "+".

Основные понятия в алгоритмических языках

3. **Данные — величины, обрабатываемые программой**. Имеется три основных вида данных: **константы, переменные и массивы**.

- **Константы** — это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения.

Примеры констант:

  - **числовые** 7.5 , 12 ;
  - **логические** **да** (истина), **нет** (ложь);
  - **символьные** (содержат ровно один символ) "A" , "+" ;
  - **литерные** (содержат произвольное количество символов) "a0", "Мир", "" (пустая строка).

- **Переменные** обозначаются именами и могут изменять свои значения в ходе выполнения программы. Переменные бывают **целые, вещественные, логические, символьные и литерные**.
- **Массивы — последовательности однотипных элементов, число которых фиксировано и которым присвоено одно имя.** Положение элемента в массиве однозначно определяется его индексами (одним, в случае одномерного массива, или несколькими, если массив многомерный). Иногда массивы называют **таблицами.**

Основные понятия в алгоритмических языках

4. Выражения — предназначаются для выполнения необходимых вычислений, состоят из констант, переменных, указателей функций (например, exp(x)), объединенных знаками операций. Выражения записываются в виде **линейных последовательностей символов** (без подстрочных и надстрочных символов, "многоэтажных" дробей и т.д.), что позволяет вводить их в компьютер, последовательно нажимая на соответствующие клавиши клавиатуры.

Различают выражения **арифметические, логические и строковые.**

- **Арифметические выражения** служат для определения одного числового значения. Например, (1+sin(x))/2. Значение этого выражения при x=0 равно 0.5, а при x=p/2 — единице.
- **Логические выражения** описывают некоторые условия, которые могут удовлетворяться или не удовлетворяться. Таким образом, логическое выражение может принимать только два значения — **"истина"** или **"ложь"** (**да** или **нет**). Рассмотрим в качестве примера логическое выражение x*x + y*y < r*r , определяющее принадлежность точки с координатами (x, y) внутренней области круга радиусом r с центром в начале координат. При x=1, y=1, r=2 значение этого выражения — **"истина"**, а при x=2, y=2, r=1 — **"ложь".**
- **Строковые (литерные) выражения, значениями которых являются тексты**. В строковые выражения могут входить литерные и строковые константы, литерные и строковые переменные, литерные функции, разделенные знаками операции сцепки. Например, A + B означает присоединение строки B к концу строки A . Если A = *"куст "*, а B = *"зеленый"*, то значение выражения A + B есть *"куст зеленый".*

Основные понятия в алгоритмических языках

5. Операторы (команды). Оператор — это наиболее крупное и содержательное понятие языка: **каждый оператор представляет собой законченную фразу языка и определяет некоторый вполне законченный этап обработки данных.** В состав операторов входят:

- ключевые слова;
- данные;
- выражения и т.д.

Операторы подразделяются на исполняемые и неисполняемые. **Неисполняемые** операторы предназначены для описания данных и структуры программы, а **исполняемые** — для выполнения различных действий (например, оператор присваивания, операторы ввода и вывода, условный оператор, операторы цикла, оператор процедуры и др.).

Стандартные функции

При решении различных задач с помощью компьютера бывает необходимо вычислить логарифм или модуль числа, синус угла и т.д.

Вычисления часто употребляемых функций осуществляются посредством подпрограмм, называемых стандартными функциями, которые заранее запрограммированы и встроены в транслятор языка.