ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

Кафедра «Организация технологических процессов**»**

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.ДВ.09.01 Информатика

Направление подготовки 38.03.02 Менеджмент

Профиль образовательной программы Производственный менеджмент

Форма обучения заочная

СОДЕРЖАНИЕ

1. КОНСПЕКТ ЛЕКЦИЙ

1. 1 Лекция №1 (2 часа).

Тема: «Введение, основные понятия информатики»

1.1.1 Вопросы лекции:

1. Этапы информатизации общества
2. Цель и задачи информатики. Основные понятия
3. Структура информатики
4. Социальные, правовые и этические аспекты информатики

1.1.2

1. Этапы информатизации общества

Бурное развитие компьютерной техники и ИТ послужило толчком к развитию общества, построенного на использовании различной информации и получившего название информационного общества. В информационном обществе изменяется не только производство, но и весь уклад жизни, система ценностей, возрастает значимость культурного досуга по отношению к материальным ценностям. По сравнение с индустриальным обществом, где все направлено на производство и потребление товаров, в информационном обществе производятся интеллект и знания. Материальной и технологической базой информационного общества становятся компьютерная техника и компьютерные сети, ИТ, телекоммуникационные связи.

Информационное общество – общество, в котором большинство работающих занято производством, хранением, переработкой и реализацией информации, особенно высшей ее формы – знаний.

Ближе всех на пути к информационному обществу стоят страны с развитой информационной индустрией (США, Япония, Англия, страны Западной Европы).

Для перехода от индустриального общества к информационному должна была возникнуть ситуация информационного кризиса. И она возникла с тем, что в 20 веке лавинообразный поток информации, хлынувший на человека, сделал практически невозможной его ориентацию в этом объеме. Возникло большое количество избыточной, лишней информации. Началом перехода к информационному обществу стало внедрение современных средств обработки и передачи информации в различных сферах деятельности человека. Этот процесс называется информатизация.

Информатизация общества – это процесс, при котором создаются условия, удовлетворяющие потребностям любого человека в получении необходимой информации (по закону РФ «Об информации, информатизации и защите информации» от 25 января 1995 года).

Цель информатизации – улучшение качества жизни людей за счет увеличения производительности и облегчения условий их труда.

2 Цель и задачи информатики. Основные понятия

Термин «информатика» прочно и надежно вошел в нашу жизнь. Он был заимствован из французского языка и обозначал название области, связанной с автоматизированной обработкой информации с помощью ЭВМ. Именно развитие компьютерной техники способствовало выделению информатики в самостоятельную область человеческой деятельности.

Следует отметить, что определений информатики в современной литературе множество. Это происходит оттого, что данная область знаний относительно новая и соответствующий понятийный аппарат не совсем устоялся. Анализ определений позволил выделить их существенную часть и сформулировать то определение, которое приведено ниже.

Информатика - область человеческой деятельности, связанная с процессами преобразования информации с помощью компьютеров и других средств вычислительной техники.

3 Структура информатики

С информатикой часто связывают одно из следующих понятий: это либо **отрасль производства**, либо **фундаментальная наука**, либо **прикладная дисциплина**, либо совокупность определенных средств, используемых для преобразования информации.

В состав **технических средств** входят компьютеры и связанные с ними периферийные устройства (мониторы, клавиатуры, принтеры и плоттеры, модемы и т.д.), линии связи, средства оргтехники и т.п., т.е. те материальные ресурсы, которые обеспечивают преобразование информации, причем главенствующую роль в этом списке играет компьютер.

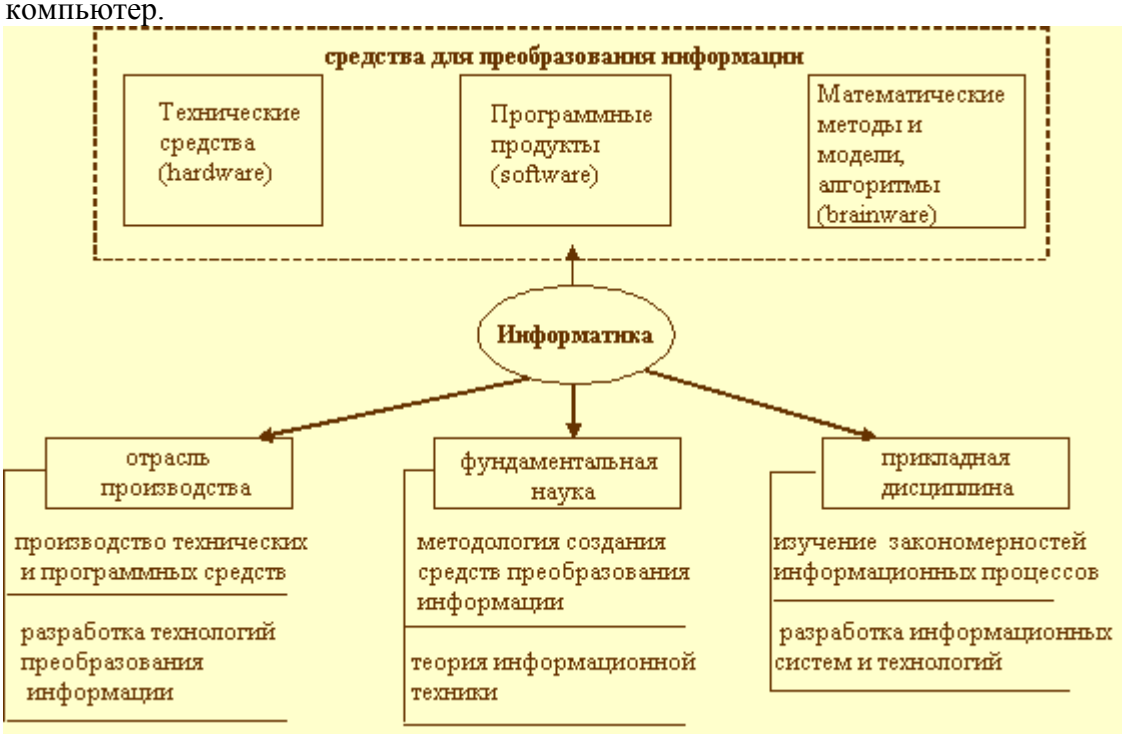


Рис 1.1 Структура информатики

К **программным средствам** (продуктам) относятся операционные системы, интегрированные оболочки, системы программирования и проектирования программных продуктов, различные прикладные пакеты, такие, как текстовые и графические редакторы, бухгалтерские и издательские системы и т.д. **Математические методы, модели и алгоритмы** являются тем базисом, который положен в основу проектирования и изготовления любого программного или технического средства в силу их исключительной сложности и, как следствие, невозможности умозрительного подхода к созданию.

Перечисленные выше три ресурсных компонента информатики играют разную роль в процессе информатизации общества. Так, совокупность программных и технических средств, имеющихся в том или ином обществе, и позволяет сделать его информационным, когда каждый член общества имеет возможность получить практически любую (исключая, естественно, секретную) интересующую его информацию (такие потребители информации называются конечными пользователями). В то же время, сложность технических и программных систем заставляет использовать имеющиеся технические и программные продукты, а также нужные методы, модели и алгоритмы для проектирования и производства новых и совершенствования старых технических и программных систем. В этом случае можно сказать, что средства преобразования информации используются для производства себе подобных. Тогда их пользователем является специалист в области информатики, а не конечный пользователь.

Разработкой абстрактных методов, моделей и алгоритмов, а также связанных с ними математических теорий занимается **фундаментальная наука**. Ее прерогативой является исследование процессов преобразования информации и на основе этих исследований разработка соответствующих теорий, моделей, методов и алгоритмов, которые затем применяются на практике.

Практическое использование результатов исследований информатики как фундаментальной науки воплощается в информатике - **отрасли производства**. В самом деле, широко известны западные фирмы по производству программных продуктов, такие как Microsoft, Lotus, Borland, и технических средств - IBM, Apple, Intel, Hewlett Packard и другие. Помимо производства самих технических и программных средств разрабатываются также и технологии преобразования информации.

Подготовкой специалистов в области преобразования информации занимается информатика как **прикладная дисциплина**. Она изучает закономерности протекания информационных процессов в конкретных областях и методологии разработки конкретных информационных систем и технологий.

Таким образом, главная функция информатики состоит в разработке методов и средств преобразования информации с использованием компьютера, а также в применении их при организации технологического процесса преобразования информации.

В рамках **прикладной дисциплины** информатики изучаются следующие вопросы:

- понятие информации, ее свойства, измерение информации, использование в управлении;
- способы кодирования информации;
- понятие и составные части информационных процессов;
- организация технических устройств преобразования информации, в частности компьютера;
- структура и методология проектирования программного обеспечения.

Задачи информатики:

- Разработка и производство современных средств вычислительной техники;
- Проектирование и внедрение прогрессивных технологий обработки информации, и как результат этого – возможность дальнейшей информатизации общества и повышения уровня его информационной культуры.

4 Социальные, правовые и этические аспекты информатики

Социальные аспекты информатики

Термин “социальные аспекты” применительно к большей части наук. Однако, информатика – не только наука.

Мало какие факторы так влияют на социальную сферу обществ как информатизация.

Информатизация общества – процесс проникновения информационных технологий во все сферы жизни и деятельности общества. Она сильнейшим образом влияет на структуру экономики ведущих в экономическом отношении стран. В числе их лидирующих отраслей промышленности традиционные добывающие и обрабатывающие отрасли оттеснены максимально наукоемкими производствами электроники, средств связи и вычислительной техники .В этих странах постоянно растут капиталовложения в научные исследования, включая фундаментальные науки. Темпы развития сферы высоких технологий и уровень прибылей в ней превышают в 5-10 раз темпы развития традиционных отраслей производства. Такая политика имеет и социальные последствия – увеличение потребности в высокообразованных специалистах и связанный с этим прогресс системы высшего образования. Информатизация меняет и облик традиционных отраслей промышленности и сельского хозяйства. Промышленные роботы, управляемые ЭВМ, станки с ЧПУ стали обычным оборудованием. Новейшие технологии в сельскохозяйственном производстве не только увеличивают производительность труда, но

и облегчают его, вовлекают более образованных людей.

Казалось бы, компьютеризация и информационные технологии несут в мир одну лишь благодать, но социальная сфера столь сложна, что последствия любого, даже гораздо менее глобального процесса, редко бывают однозначными. Рассмотрим, например, такие социальные последствия информатизации как рост производительности труда, интенсификацию труда, изменение условий труда. Все это, с одной стороны, улучшает условия жизни многих людей, повышает степень материального и интеллектуального комфорта, стимулирует рост числа высокообразованных людей, а с другой – является источником повышенной социальной напряженности. Например, появление на производстве промышленных роботов ведет к полному изменению технологии, которая перестает быть ориентированной на человека. Тем самым меняется номенклатура профессий. Значительная часть людей вынуждена менять либо специальность, либо место работы – рост миграции населения характерен для большинства развитых стран. Государство и частные фирмы поддерживают систему повышения квалификации и переподготовки, но не все люди справляются с сопутствующим стрессом. Одним словом, жизнь в "информационном обществе" легче, по-видимому, не становится, а вот то, что она значительно меняется – несомненно

Правовые аспекты информатики

Деятельность программистов и других специалистов, работающих в сфере информатики, все чаще выступает в качестве объекта правового регулирования. Некоторые действия при этом могут быть квалифицированы как правонарушения (преступления).

Правовое сознание в целом, а в области информатики особенно, в нашем обществе находится на низком уровне. Все ли знают ответы на следующие вопросы:

• можно ли, не копируя купленную программу, предоставить возможность пользоваться ею другому лицу;

• кому принадлежит авторское право на программу, созданную студентом в ходе выполнения дипломной работы;

• можно ли скопировать купленную программу для себя самого, чтобы иметь резервную копию;

• можно ли декомпилировать программу, чтобы разобраться в ее деталях или исправить ошибки;

• в чем состоит разница между авторским и имущественным правом.

Вопросов, подобных этим, возникает множество, но остановимся на правовом регулировании в области информатики в России. К 1992 году был принят Закон Российской Федерации "О ПРАВОВОЙ ОХРАНЕ ПРОГРАММ ДЛЯ ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И БАЗ ДАННЫХ", содержащий обширный план приведения российского законодательства в сфере информатики в соответствие с мировой практикой. Действие этого Закона распространяется на отношения, связанные с созданием и использованием программ для ЭВМ и баз данных. Также предусматривалось внести изменения и дополнения в Гражданский кодекс РФ, в Уголовный кодекс РФ, другие законодательные акты, связанные с вопросами правовой охраны программ для электронных вычислительных машин и баз данных, привести решения Правительства РФ в соответствие с Законом, обеспечить пересмотр и отмену государственными ведомствами и другими организациями РФ их нормативных актов, противоречащих указанному Закону, обеспечить принятие нормативных актов в соответствии с указанным Законом и т.д. Главное содержание данного Закона – юридическое определение понятий, связанных с авторством и распространением компьютерных программ и баз данных, таких как Авторство, Адаптация, База данных, Воспроизведение, Декомпилирование. Использование, Модификация и т.д., а также установление прав, возникающих при создании программ и баз данных – авторских, имущественных, на передачу, защиту, регистрацию, неприкосновенность и т.д.

Этические аспекты информатики

Этика – система норм нравственного поведения человека.

Далеко не все правила, регламентирующие деятельность в сфере информатики, можно свести в правовым нормам. Очень многое определяется соблюдением неписаных правил поведения для тех, кто причастен к миру компьютеров. Впрочем, в этом отношении информатика ничуть не отличается от любой другой сферы деятельности человека в обществе.

Морально-этические нормы в среде информатиков отличаются от этики повседневной жизни несколько большей открытостью, альтруизмом. Большинство нынешних специалистов-информатиков сформировались и приобрели свои знания и квалификацию благодаря бескорыстным консультациям и содействию других специалистов. Очевидно, поэтому они готовы оказать бескорыстную помощь, дать совет или консультацию, предоставить компьютер для выполнения каких-либо манипуляций с дискетами и т.д. Ярким примером особой психологической атмосферы в среде информатиков является расширяющееся международное движение программистов, предоставляющих созданные ими программные средства для свободного распространения.

2.1. Лекция № 2 (2 часf)

Тема: Алгоритмизация вычислительных процессов

6.1.1. Вопросы лекции:

1. Понятие алгоритмизации вычислительных процессов
2. Понятие алгоритма и его свойства, способы описания алгоритмов
3. Основные типы алгоритмов
3.1. Алгоритм линейной структуры
3.2. Алгоритм ветвящейся структуры
3.3. Алгоритм циклической структуры

6.1.2. Краткое содержание вопросов

1. Понятие алгоритмизации вычислительных процессов

Разработке алгоритма предшествуют такие этапы, как формализация и моделирование задачи. Формализация предполагает замену словесной формулировки решаемой задачи краткими символьными обозначениями, близкими к обозначениям в языках программирования или к математическим. Моделирование задачи является важнейшим этапом, целью которого является поиск общей концепции решения. Обычно моделирование выполняется путем выдвижения гипотез решения задачи и их проверке любым рациональным способом (прикидочные расчеты, физическое моделирование и т.д.). Результатом каждой проверки является либо принятие гипотезы, либо отказ от нее и разработка новой.

При разработке алгоритма используют следующие основные принципы.

Принцип поэтапной детализации алгоритма (другое название - "проектирование сверху-вниз"). Этот принцип предполагает первоначальную разработку алгоритма в виде укрупненных блоков (разбиение задачи на подзадачи) и их постепенную детализацию.

Принцип "от главного к второстепенному", предполагающий составление алгоритма, начиная с главной конструкции. При этом, часто, приходится "достраивать" алгоритм в обратную сторону, например, от середины к началу.

Принцип структурирования, т.е. использования только типовых алгоритмических структур при построении алгоритма. Нетиповой структурой считается, например, циклическая конструкция, содержащая в теле цикла дополнительные выходы из цикла. В программировании нетиповые структуры появляются в результате злоупотребления

командой безусловного перехода (GoTo). При этом программа хуже читается и труднее отлаживается.

Говоря о блок-схемах, как о средстве записи алгоритма, можно дать еще один совет по их разработке. Рекомендуется после внесения исправлений в блок-схему аккуратно перерисовывать ее с учетом этих исправлений. Аккуратность записи есть аккуратность мысли программиста. Аккуратно записанный и детализованный алгоритм упрощает его программирование и отладку.

Этапы решения задачи на компьютере.

Решение задачи разбивается на этапы:

1. Постановка задачи
2. Формализация (математическая постановка)
3. Выбор (или разработка) метода решения
4. Разработка алгоритма
5. Составление программы
6. Отладка программы
7. Вычисление и обработка результатов

1. При постановке задачи выясняется конечная цель и вырабатывается общий подход к решению задачи. Выясняется сколько решений имеет задача и имеет ли их вообще. Изучаются общие свойства рассматриваемого физического явления или объекта, анализируются возможности данной системы программирования.

2. На этом этапе все объекты задачи описываются на языке математики, выбирается форма хранения данных, составляются все необходимые формулы.

3. Выбор существующего или разработка нового метода решения (очень важен и, в то же время личностный этап).

4. На этом этапе метод решения записывается применительно к данной задаче на одном из алгоритмических языков (чаще на графическом).

5. Переводим решение задачи на язык, понятный машине.

2. Понятие алгоритма и его свойства, способы описания алгоритмов

"Алгоритм" является фундаментальным понятием информатики. Представление о нем необходимо для эффективного применения вычислительной техники к решению практических задач. Алгоритм - это предписание исполнителю (человеку или автомату) выполнить точно определенную последовательность действий, направленных на достижение заданной цели. Алгоритм - это сформулированное на некотором языке правило, указывающее на действия, последовательное выполнение которых приводит от исходных данных к искомому результату. Значение слова алгоритм очень схоже со значением слов рецепт, процесс, метод, способ. Однако любой алгоритм, в отличие от рецепта или способа, обязательно обладает следующими свойствами.

Свойства алгоритма (отличающие его от любых других предписаний): понятность (для конкретного исполнителя); дискретность (команды последовательны, с точной фиксацией моментов начала и конца выполнения команды); точность (после выполнения каждой команды точно известно, завершено ли исполнение алгоритма или же какая команда должна выполниться следующей); результативность (после конечного числа шагов задача решается или же становится ясно, что процесс решения не может быть продолжен): массовость (алгоритм единым образом применяется к любой конкретной формулировке задачи, для которой он разработан).

1. Дискретность - разбиение алгоритма на ряд отдельных законченных действий - шагов. Выполнение алгоритма разбивается на последовательность законченных действий

- шагов. Каждое действие должно быть закончено исполнителем алгоритма прежде, чем он приступит к исполнению следующего действия.

2. Точность - однозначные указания. На каждом шаге однозначно определено преобразование объектов среды исполнителя, полученной на предыдущих шагах алгоритма. Если алгоритм многократно применяется к одному и тому же набору исходных данных, то на выходе он получает каждый раз один и тот же результат. Запись алгоритма должна быть такой, чтобы на каждом шаге его выполнения было известно, какую команду надо выполнять следующей.

3. Понятность - однозначное понимание и исполнение каждого шага алгоритма его исполнителем. Алгоритм должен быть записан на понятном для исполнителя языке.

4. Результативность - обязательное получение результата за конечное число шагов. Каждый шаг (и алгоритм в целом) после своего завершения дает среду, в которой все объекты однозначно определены. Если это по каким-либо причинам невозможно, то алгоритм должен сообщать, что решение задачи не существует. Работа алгоритма должна быть завершена за конечное число шагов. Информатика оперирует только с конечными объектами и конечными процессами, поэтому вопрос о рассмотрении бесконечных алгоритмов остается за рамками теории алгоритмов.

5. Массовость - применение алгоритма к решению целого класса однотипных задач.

Порядок выполнения алгоритма:

- Действия в алгоритме выполняются в порядке их записи
- Нельзя менять местами никакие два действия алгоритма
- Нельзя не закончив одного действия переходить к следующему

Для записи алгоритмов используются специальные языки:

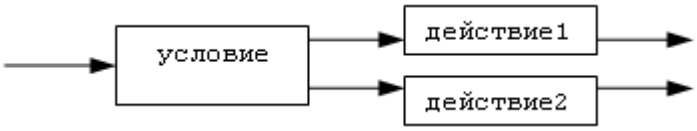
1. Естественный язык (словесная запись)
2. Формулы
3. Псевдокод
4. Структурограммы
5. Синтаксические диаграммы
6. Графический (язык блок-схем)

1. Естественный язык:

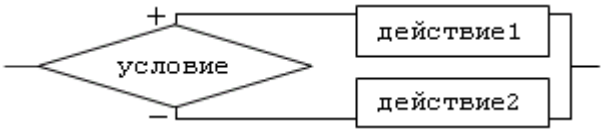
если условие то действие1 иначе действие2

2. Структурограмма:

3. Синтаксическая диаграмма:



4. Графический язык:



Составление алгоритмов графическим способом подчиняется двум ГОСТам:

1. ГОСТ 19.002-80, соответствует международному стандарту ИСО 2636-73. Регламентирует правила составления блок-схем.

2. ГОСТ 19.003-80, соответствует международному стандарту ИСО 1028-73. Регламентирует использование графических примитивов.

Название	Символ (рисунок)	Выполняемая функция (пояснение)
1. Блок вычислений		Выполняет вычислительное действие или группу действий
2. Логический блок		Выбор направления выполнения алгоритма в зависимости от условия
3. Блоки ввода/вывода		Ввод или вывод данных вне зависимости от физического носителя
		Вывод данных на печатающее устройство
4. Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
5. Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
6. Блок модификации		Выполнение действий, изменяющих пункты алгоритма
7. Соединитель		Указание связи между прерванными линиями в пределах одной страницы
8. Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Правила построения блок-схем:

Блок-схема выстраивается в одном направлении либо сверху вниз, либо слева направо

Все повороты соединительных линий выполняются под углом 90 градусов

3. Основные типы алгоритмов

3.1. Алгоритм линейной структуры

Линейным называется вычислительный процесс, в котором предусматривается получение результата путем однократного выполнения последовательности действий при любых значениях исходных данных. Характерной особенностью линейного вычислительного процесса является то, что направление вычислений не зависит от исходных данных и промежуточных результатов.

Алгоритм линейного вычислительного процесса графически может быть представлен **блоком следования** – **композицией** (объединением) нескольких следующих друг за другом блоков ПРОЦЕСС (рис. 1).

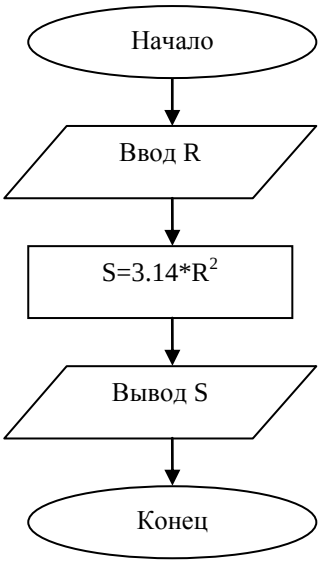
Рисунок 1- Блок-схема линейного алгоритма

Линии, соединяющие отдельные блоки на блок-схеме алгоритма и указывающие на последовательность действий, называются **линиями потока**. Направление линии потока сверху вниз принимается за основное и стрелкой не обозначается.

Рассмотрим пример задачи, решение которой представляет собой линейный вычислительный процесс, и составим блок-схему алгоритма.

Пример: Найдите площадь круга S при заданном значении радиуса R. Для вычисления площади использовать формулу $S=3,14*R^2$.

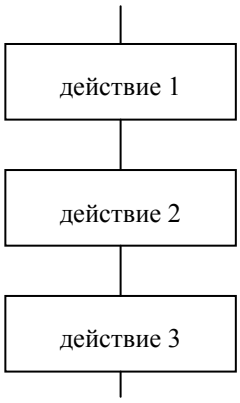
Блок-схема



Как правило, составление алгоритма линейного вычислительного процесса не вызывает затруднений, однако в "чистом виде" такие вычислительные процессы встречаются весьма редко. Чаще всего они являются составной частью более сложных вычислительных процессов. Таким образом:

- линейный вычислительный процесс является наиболее простым при реализации его на ЭВМ.

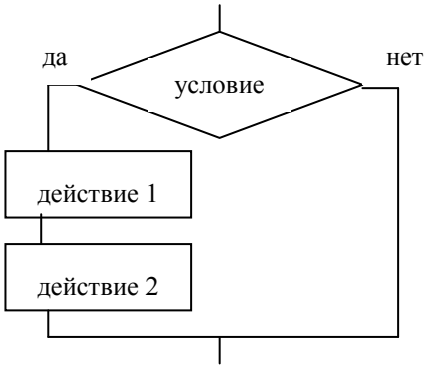
- блок-схема алгоритма линейного вычислительного процесса представляет собой **композицию** (объединение) нескольких следующих друг за другом блоков ПРОЦЕСС.



3.2. Алгоритм ветвящейся структуры

Разветвляющимся вычислительным процессом называется процесс, направление вычислений в котором зависит от результата проверки некоторого условия (условий).

Алгоритм разветвляющегося вычислительного процесса предусматривает **выбор** одной из нескольких возможных **альтернатив** (последовательностей действий) в зависимости от значения исходных данных или промежуточных результатов. Каждую из этих последовательностей называют **ветвью алгоритма**. Блок-схема алгоритма разветвляющегося вычислительного процесса содержит, по крайней мере, один блок РЕШЕНИЕ. Направления линий потока сверху вниз и слева направо на блок-схеме принимаются за основные и стрелками не обозначаются.



Графической интерпретацией алгоритма разветвляющегося вычислительного процесса является **блок разветвления алгоритма**, в котором может быть предусмотрен **полный** (рис. 2) или **неполный** (рис. 3) **выбор.**

В блоке разветвления алгоритма с полным выбором в зависимости от результата проверки условия выполняются только действия ветви "да" (т. е., действия 1 и 2) или только действия ветви "нет" (действия 3 и 4). В блоке с неполным выбором в зависимости от результата проверки условия либо выполняются действия какой-либо ветви, либо они игнорируются.

Некоторой разновидностью блока разветвления алгоритма является **блок множественного выбора** (рис. 3). В нем, в зависимости от результатов выбора, выполняется одно из предусмотренных действий.

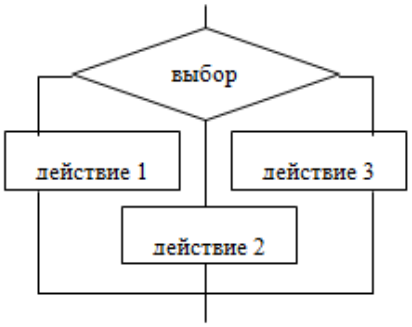
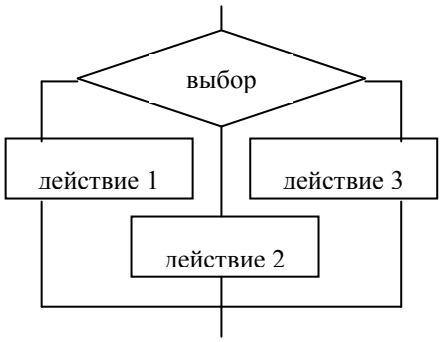


Рисунок 3– Блок множественного выбора

Рассмотрим пример задачи, алгоритм решения которой носит разветвляющийся характер.



Пример 2. Составить схему алгоритма для вычисления значений функции

$$y = \begin{cases} 2x + a, & \text{если } x < 0, \\ x + 3b^2, & \text{если } x \geq 0. \end{cases}$$

Поскольку значение функции вычисляется по разным формулам в зависимости от знака аргумента **x**, в алгоритме вычислительного процесса должна быть предусмотрена проверка знака названного аргумента. Блок-схема алгоритма решения задачи на ЭВМ представлена на рис. 4.

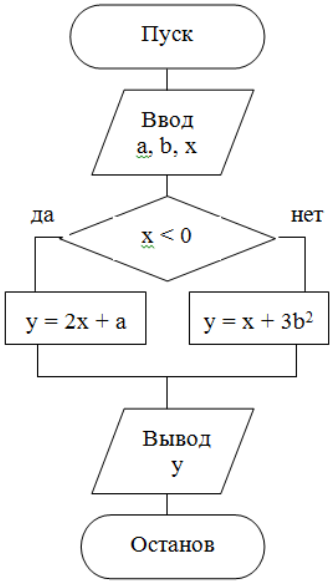


Рисунок 4 – Блок-схема алгоритма

В разветвляющихся вычислительных процессах могут иметь место так называемые "**пустые**" **ветви**. В направлении пустых ветвей задача, как правило, не имеет решения (вычислительный процесс не приводит к получению какого-либо промежуточного или конечного результата). Рассмотрим следующий пример.

Пример 3. Составить блок-схему алгоритма для вычисления значений функции

$$y = \begin{cases} x - a, \ если \ \ x = 0, \\ x + b^2, \ если \ \ 0 < x < 3, \\ x^2 + c, \ если \ \ 3 < x < 5. \end{cases}$$

Как видно из условия задачи, переменная **x** может принимать только значения, находящиеся в диапазоне от 0 до 5. Следовательно, для остальных значений **x** ветви в алгоритме оказываются пустыми. В блок-схеме алгоритма должны быть предусмотрены действия ЭВМ на случай появления исходных данных (значений **x**), не описанных логическими условиями. На рис. 6 показан один из возможных способов решения проблемы.

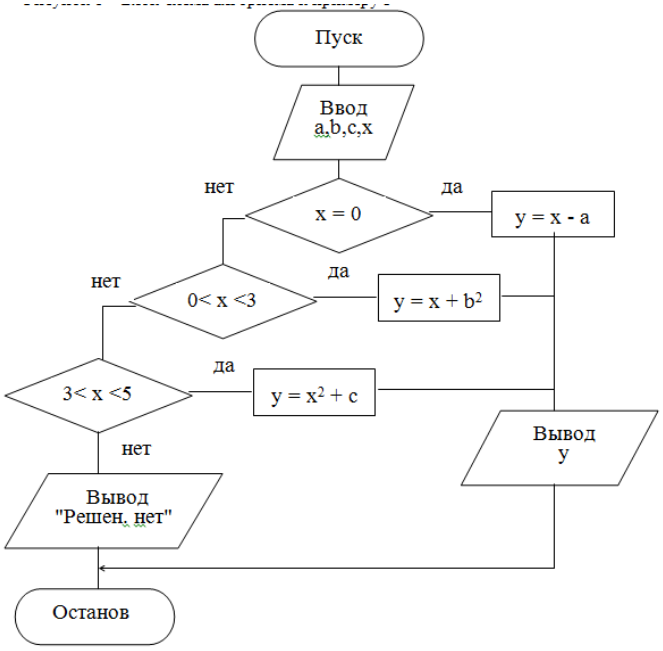


Рисунок 5 – Блок-схема алгоритма к примеру 3

В заключение кратко подведем итог:

1. Алгоритм разветвляющегося вычислительного процесса предусматривает выбор одной из нескольких возможных альтернатив в зависимости от значения исходных данных или промежуточных результатов.

2. В блоке разветвления алгоритма может быть предусмотрен полный или неполный выбор.

3. В разветвляющихся вычислительных процессах могут иметь место "пустые" ветви.

3.3. Алгоритм циклической структуры

Циклическим называется вычислительный процесс, в котором получение результата обеспечивается путем **многократного повторения некоторой последовательности действий.**

Графической интерпретацией алгоритма циклического вычислительного процесса является **блок цикла.** Различают несколько разновидностей блока цикла: **блок цикла с параметром, блок цикла с предварительным условием** и **блок цикла с последующим условием.**

Блок-схема **блока цикла с параметром** представлена на рис. 6.

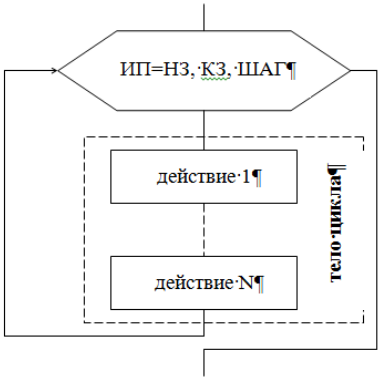


Рисунок 6 – Блок цикла с параметром

На рис. 6 приняты следующие сокращения:

ИП – имя ячейки памяти, в которую заносится значение параметра;

НЗ – начальное значение параметра;

КЗ – конечное значение параметра;

ШАГ – величина приращения параметра после каждого выполнения тела цикла.

Тело цикла представляет собой линейный вычислительный процесс и выполняется столько раз, сколько разных значений примет параметр в заданных пределах от НЗ до КЗ. Блок цикла с параметром относится к циклу **с явно выраженным числом повторений** (число повторений известно заранее). Для таких циклов характерным является то, что задаются:

- начальное и конечное значения **параметра** цикла;

- закон изменения параметра цикла при каждом повторном выполнении тела цикла;

- количество повторных **выполнений** тела цикла (вытекает из первых двух пунктов).

Блок цикла с предварительным условием и блок цикла с последующим условием относятся к так называемым **итерационным циклам**. В таких циклических вычислительных процессах число повторений тела цикла заранее не известно. Выход из цикла осуществляется не после того, как цикл повторится заданное число раз, а при

выполнении определенного условия, связанного с проверкой значения монотонно изменяющейся в теле цикла величины. Блок-схема блока цикла с предварительным условием представлена на рис. 8, а блока цикла с последующим условием – на рис. 7.

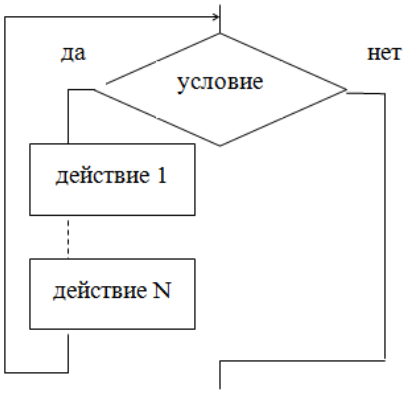


Рисунок 7 – Блок цикла с предварительным условием

Кратко суть алгоритма цикла с предварительным условием можно изложить следующим образом: **пока выполняется условие – повторять действия**. В таких циклах возможны ситуации, когда тело цикла не выполняется ни разу (например, если при первой же проверке не выполняется условие, то сразу происходит выход из цикла).

В цикле с последующим условием (рис. 8) тело цикла выполняется не менее одного раза. При этом **действия**, предусмотренные в теле цикла, **выполняются до тех пор, пока не выполнится заданное условие.**

Рассмотренные блоки циклов позволяют описать **простые** циклические вычислительные процессы. При решении сложных задач может возникнуть необходимость внутри одного цикла организовать дополнительно один или несколько циклов. Такие циклы называются **вложенными**. При этом цикл, внутри которого создается другой цикл, называется **внешним**, а цикл, создаваемый внутри другого – **внутренним**. Правила организации как внешнего, так и внутреннего циклов те же, что и для простых циклов. Параметры внешнего и внутреннего циклов должны быть разными.

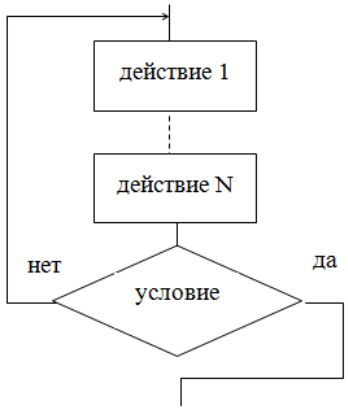


Рисунок 8 - Блок цикла с последующим условием

3.1.Лекция № 3 (2 часа)

Тема: Программирование на алгоритмическом языке QBASIC

3.1.1. Вопросы лекции:

1. Основные понятия алгоритмического языка QBASIC. История развития QBASIC. Операции над числами и данными

2. Встроенные функции

3. Арифметические выражения
4. Основные операторы алгоритмического языка QBASIC
4.1. Оператор присваивания LET
4.2. Операторы ввода данных READ, DATA, RESTORE, INPUT
4.3. Операторы вывода данных PRINT, LPRINT
4.4. Условные операторы IF … THEN, GOTO, SELECT CASE
4.5. Операторы организации цикла FOR … NEXT, WHILE … WEND, DO … LOOP
4.6. Операторы: REM, CLS, STOP, END

8.1.2. Краткое содержание вопросов

1. Основные понятия алгоритмического языка QBASIC. История развития QBASIC. Операции над числами и данными

Бе́йсик (от BASIC, сокращение от англ. Beginner's All-purpose Symbolic Instruction Code — универсальный код символических инструкций для начинающих; англ. basic — основной, базовый) — семейство высокоуровневых языков программирования.

Был разработан в 1963 году профессорами Дартмутского колледжа Томасом Курцем и Джоном Кемени.

Язык предназначался для обучения программированию и получил широкое распространение в виде различных диалектов, прежде всего, как язык для домашних компьютеров.

В 1975 году, Пол Ален, молодой программист из Бостона, в содружестве со студентом Гарвардского университета Биллом Гейтсом написали программу, реализующую для микрокомпьютера Альтаир 8800 язык Бейсик, впервые использовав его для программного обеспечения персональных компьютеров. Впоследствии Гейтс и Аллен основали собственную фирму Microsoft.[1]

Алгоритмический язык Basic используется преимущественно в режиме диалога человека и ЭВМ. Этот язык ориентирован на решение различных задач вычислительного и невычислительного характера с небольшим объемом исходной информации. Название языка BASIC возникло от сокращения английских слов Beginner's All-purpose Symbolic Instruction Code (многоцелевой язык символических инструкций для начинающих). Следует отметить, что стандарта на язык Basic не существует и различные его модификации могут существенно отличаться друг от друга.

В языке Basic существуют как средства для описания действий алгоритма, которые используются при составлений-программ, - операторы Basic, так и средства, которые служат для общения с ЭВМ. Последние имеют форму приказов для немедленного выполнения и называются командами.

Основным режимом в Basic является программный режим, когда заранее составленная программа полностью вводится в ЭВМ и затем выполняется.

Программа на Basic состоит из строк, которые могут иметь номер. В одной строке может содержаться один или несколько операторов, разделенных символом : (двоеточие). Обычно строки нумеруются, начиная с 10, с шагом 10. Номера строк используются в операторах передачи управления. При этом оператор, которому передается управление, должен быть первым в строке.

2. Встроенные функции

Для вычисления наиболее распространенных элементарных математических функций в языке БЕЙСИК применяют встроенные стандартные функции. Аргумент функции заключается в круглые скобки. Аргументом функции может быть произвольное арифметическое выражение. Для обращения к функции нужно набрать ее имя и указать аргумент. Стандартные математические функции, применяемые в языке Бейсик, приведены в ТАБЛИЦЕ 1.

Аргументы тригонометрических функций должны быть заданы в радианах. Если угол выражен в градусах, то его нужно перевести в радианы по формуле

Значение в радианах = значение в градусах * 3.14/180

В некоторых версиях языка Бейсик имеется функция LOG(X) для вычисления натурального логарифма числа X. Однако от логарифма с любым основанием легко перейти к натуральному логарифму, используя формулу

Log_aN = ln N/ln a, где a – основание.

Над числовыми константами, переменными и стандартными функциями можно производить обычные арифметические операции:

1. вычисляются значения функций
2. ^ возведение в степень,
3. * умножение, / деление,
4. + сложение, - вычитание.

ТАБЛИЦА 1

Обозначение функции на языке Бейсик	Название	Пояснение
ABS(X)	Функция абсолютная величина |x| (модуль)	Функция вычисляет абсолютное значение (модуль) аргумента X
SQR(X)	Функция квадратный корень (√x)	Функция вычисляет квадратный корень положительного аргумента X
LOG(X)	Функция натуральный логарифм ln x	Функция вычисляет натуральный логарифм положительного аргумента X
EXP(X)	Экспоненциальная функция e^x	Вычисляет e^x, где e=2,71828, X-любое число
COS(X)	Функция косинуса cos x	Вычисляет косинус аргумента X
SIN(X)	Функция синуса sin x	Вычисляет синус аргумента X
TAN(X)	Функция тангенса tg x	Вычисляет тангенс аргумента X, находящегося в диапазоне от $-\frac{\pi}{2}$ до $\frac{\pi}{2}$
ATN(X)	Функция арктангенса arctg x	Вычисляет арктангенс аргумента X
INT(X)	Целочисленная функция	Определяет наибольшее целое число, не превосходящее X
RND(X)	Функция случайных чисел	Выдает случайное число, лежащее в интервале от 0 до 1. Значение аргумента X игнорируется

3. Арифметические выражения

Арифметическое выражение – это символическая запись, указывающая правила вычисления числового выражения. Оно состоит из чисел (констант), имен переменных, функций, знаков арифметических операций и скобок.

При записи выражений необходимо учитывать следующие рекомендации и ограничения:

1. Формулу требуется записывать в строку без каких-либо подстрочных и надстрочных знаков.

2. Следует использовать круглые скобки для указания порядка действий, особенно в сомнительных случаях. Вычисления в скобках производятся в первую очередь. Если выражение, содержащее скобки само заключено в скобки, то вычисления производятся, начиная с внутренних скобок. Внутри скобок действия выполняются с лева на право в соответствии с приоритетом операций:

- Сначала вычисляются значения функций,

Затем - выполняются все операции возведения в степень,

- Затем - умножения и деления,
- И наконец, сложения и вычитания.

Если математическое выражение содержит несколько операций одинакового старшинства, то такие операции выполняются последовательно слева на право.

3. Два знак арифметических операций нельзя ставить рядом, а также нельзя опускать знак умножения между сомножителями.

4. Выполнение арифметических операций над арифметическими выражениями одного тепа дает результат того же типа. Операция над целой и вещественной величиной дает вещественный результат.

5. Возведение в целую степень выполняется многократным умножением или с помощью операции ^ возведения в степень.

6. Вычисление результата возведения в степень осуществляется с помощью функций EXP и LOG, если показатель степени - действительное число:

x^y **= EXP(Y*LOG(X))**

Так как логарифмы отрицательных чисел не существуют, то нельзя возводить отрицательные числа в действительную степень.

3.4. Основные операторы алгоритмического языка QBASIC

В результате изучения студент должен знать:

- Понятие оператора;
- Оператор присваивания в Бейсике;
- Операторы задания значений переменным в Бейсике;
- Оператор вывода в Бейсике;
- Оператор ввода в Бейсике;
- Оператор комментариев в Бейсике;
- Оператор очищения экрана;
- Оператор конца текста программы;
- Оператор безусловного перехода;
- Оператор условного перехода в полной форме;
- Оператор условного перехода в неполной форме.

уметь:

- Приводить примеры использования операторов языка программирования Бейсик;
- Использовать операторы языка программирования Бейсик при составлении простейших программ.

Оператор – это предписание ЭВМ, написанное на языке БЕЙСИК. Операторы можно разделить на две группы:

1. Выполняемые операторы – определяют действие программы, указывая Бейсик-системе, какую операцию нужно выполнить (PRINT – печатать, INPUT - ввести);

2. Невыполняемые операторы – описывают характер и упорядочение данных, позволяют вводить в программу примечания и сообщения описательного характера (DATA, REM).

12.1. Лекция № 4 (2часа)

Тема: Компьютерные сети

12.1.1. Вопросы лекции:

1. Основные понятия. Классификация компьютерных сетей
2. Локальные вычислительные сети
3. Глобальная вычислительная сеть
4. Требования, предъявляемые к современным вычислительным сетям

12.1.2. Краткое содержание вопросов

1. Основные понятия. Классификация компьютерных сетей

По мере того, как в информационные процессы вовлекалось все больше пользователей, как расширялся круг задач, решаемых с помощью ЭВМ, возникла необходимость в децентрализации процессов обработки данных. Принцип **централизованной** обработки данных (рисунок 1), когда к одной большой ЭВМ подключалось несколько терминалов (рабочих мест, состоящих из дисплея и клавиатуры), не отвечал высоким требованиям к надежности процесса обработки, затруднял развитие систем и не мог обеспечить необходимые временные параметры при диалоговой обработке данных в многопользовательском режиме. Кратковременный выход из строя центральной ЭВМ приводил к роковым последствиям для системы в целом. Поэтому приходилось дублировать функции центральной ЭВМ, значительно увеличивая затраты на создание и эксплуатацию систем обработки данных.

Распределенная обработка данных – обработка данных, выполняемая на независимых, но связанных между собой ЭВМ, представляющих распределенную систему.

Для реализации распределенной обработки данных были созданы **многомашинные ассоциации**, структура которых разрабатывается по одному из следующих направлений:

– многомашинные вычислительные комплексы (МВК);

– компьютерные (вычислительные) сети.

Многомашинный вычислительный комплекс – это группа установленных рядом вычислительных машин, объединенных с помощью специальных средств сопряжения и выполняющих совместно единый информационно-вычислительный процесс.

Многомашинные вычислительные комплексы могут быть:

– **локальными,** при условии установки ЭВМ в одном помещении;

– **дистанционными**, если некоторые ЭВМ комплекса установлены на значительном расстоянии от центральной ЭВМ и для передачи данных используются телефонные каналы связи.

С появлением персональных ЭВМ их также стали объединять в сети.

Компьютерная (вычислительная) сеть – это совокупность компьютеров, соединенных с помощью каналов связи в единую систему, удовлетворяющую требованиям распределенной обработки данных.

Глобальная вычислительная сеть объединяет абонентов, расположенных в различных странах, на различных континентах. Взаимодействие между абонентами такой сети может осуществляться на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные вычислительные сети позволяют решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам. Примером глобальной вычислительной сети служит **всемирная сеть Internet.**

Региональная вычислительная сеть связывает абонентов, расположенных на значительном расстоянии друг от друга. Она может включать абонентов внутри большого

города, экономического региона, отдельной страны. Обычно в этом случае расстояние между абонентами вычислительной сети составляет десятки – сотни километров.

Локальная вычислительная сеть объединяет абонентов, расположенных в пределах небольшой территории. В настоящее время не существует четких ограничений на территориальный разброс абонентов локальной вычислительной сети. Обычно такая сеть привязана к конкретному месту. К классу локальных вычислительных сетей относятся сети отдельных предприятий, фирм, банков, офисов и т. д. Протяженность такой сети можно ограничить пределами 2 … 2,5 км.

2. Локальные вычислительные сети

Назначение всех видов компьютерных ЛВС определяется, в основном, двумя функциями:

обеспечение совместного использования аппаратных и программных ресурсов сети;

обеспечение совместного доступа к ресурсам данных.

Локальные вычислительные сети объединяют относительно небольшое число ПЭВМ (обычно от 10 до 100) в пределах небольшой территории (одно помещение, здание или учреждение). Традиционное название – локальная вычислительная сеть – скорее дань тем временам, когда сети использовались в основном для решения вычислительных задач. Сегодня же в 99% случаев речь идет исключительно об обмене информацией в виде текстов, графических и видео-образов, числовых массивов.

Основными техническими средствами ЛВС являются **рабочие станции** (обычные ПЭВМ), которые могут быть подключены к более мощному компьютеру – **сетевому серверу**.

Сервер – это мощный компьютер, способный одновременно обрабатывать сотни (тысячи) запросов на обслуживание, поступающих от рабочих станций.

Рабочей станцией (или **сетевым узлом) называется ПЭВМ, используемая в качестве клиента, то есть потребителя сетевых услуг, предоставляемых сервером.**

3. Глобальная вычислительная сеть

Зарождение Интернета принято считать с момента появления первой компьютерной сети, родиной которой в середине 60-х годов двадцатого века стала Америка.

В то время еще не существовало персональных компьютеров, и крупные американские университеты могли себе позволить 1-2 больших компьютера. Компьютерное время было драгоценным ресурсом, и на него заранее записывались. Люди работали ночами, чтобы ни минуты этого времени не пропало даром.

Наконец появилась идея соединить между собой компьютеры разных университетов, чтобы сделать возможным удаленное использование любого свободного в данный момент компьютера. Этот проект получил название ARPANET. К концу 1969 года были соединены компьютеры четырех университетов и появилась первая компьютерная сеть.

Очень скоро обнаружилось, что сеть в основном используется не для вычислений на удаленном компьютере, а для обмена сообщениями между пользователями. В 1972 году, когда ARPANET уже соединял 23 компьютера, была написана первая программа для обмена электронной почтой по сети. Электронную почту оценили по достоинству, что побудило целый ряд государственных организаций и корпораций к созданию собственных компьютерных сетей. Эти сети обладали тем же недостатком, что и ARPANET: они могли соединять только ограниченное число однотипных компьютеров. Кроме того, они были не совместимы друг с другом.

В середине 70-х годов для ARPANET были разработаны новые стандарты передачи данных, которые позволяли объединять между собой сети произвольной архитектуры,

тогда же было придумано слово "Интернет". Именно эти стандарты, впоследствии получившие название протокола TCP/IP, заложили основу для роста глобальной компьютерной сети путем объединения уже существующих сетей. Их важным достоинством было то, что сеть считалась в принципе не стопроцентно надежной, и предусматривались средства борьбы с ошибками при передаче данных. В 1983 году сеть ARPANET перешла на новый протокол и разделилась на две независимые сети - военную и образовательную. К этому времени сеть объединяла более тысячи компьютеров, в том числе в Европе и на Гавайских островах. Последние использовали спутниковые каналы связи.

Развитие Интернета получило новый импульс благодаря инициативе Национального научного фонда США (NSF) по созданию глобальной сетевой инфраструктуры для системы высшего образования (1985-88). NSF создал сеть скоростных магистральных каналов связи и выделял средства на подключение к ней американских университетов, при условии, что университет обеспечивал доступ к сети для всех подготовленных пользователей. Интернет оставался преимущественно университетской сетью до начала 90-х годов, однако NSF сразу взял курс на то, чтобы сделать его в дальнейшем независимым от государственного финансирования. В частности, NSF поощрял университеты к поиску коммерческих клиентов. К 1988 году Интернет уже насчитывал около 56 тысяч соединенных компьютеров.

Настоящий расцвет Интернета начался в 1992 году, когда была изобретена новая служба, получившая странное название «Всемирная паутина» (World Wide Web, или WWW, или просто «веб»). WWW позволял любому пользователю Интернета публиковать свои текстовые и графические материалы в привлекательной форме, связывая их с публикациями других авторов и предоставляя удобную систему навигации. Постепенно Интернет начал выходить за рамки академических институтов и стал превращаться из средства переписки и обмена файлами в гигантское хранилище информации. К 1992 году Интернет насчитывал более миллиона соединенных компьютеров.

В настоящее время Интернет продолжает расти с прежней головокружительной скоростью. По оценке специалистов, количество передаваемой информации (трафик) в Интернете увеличивается на 30% ежемесячно. В 1999 году Интернет объединял около 60 миллионов компьютеров и более 275 миллионов пользователей, и каждый день в нем появлялось полтора миллиона новых вебовских документов. Эти оценки довольно приблизительны, потому что в Интернете нет центрального административного органа, который регистрировал бы новых пользователей и новые компьютеры.

В Россию Интернет впервые проник в начале 90-х годов. Ряд университетов и исследовательских институтов приступили в это время к построению своих компьютерных сетей и обзавелись зарубежными каналами связи. Особенно следует отметить Институт Атомной Энергии им. Курчатова. На базе ИАЭ сложились две крупнейшие коммерческие компании, предоставляющие услуги по подключению к Интернету – «Релком» и «Демос», а также Российский Институт Развития Общественных Сетей (РОСНИИРОС). Последний стал в дальнейшем головной организацией, координирующей развитие российской части Интернета.

4. Требования, предъявляемые к современным вычислительным сетям

Производительность

Определяется такими показателями: время реакции системы - время между моментом возникновения запроса и моментом получения ответа. Пропускная способность сети определяется количеством информации, переданной через сеть или ее сегмент в единицу времени. Определяется в битах в секунду.

Надежность

Определяется надежностью работы всех ее компонентов. Для повышения надежности работы аппаратных компонентов обычно используют дублирование, когда при отказе одного из элементов функционирование сети обеспечат другие.

При работе компьютерной сети должна обеспечиваться *сохранность* информации и защита ее от искажений. Как правило, важная информация в сети хранится в нескольких экземплярах. В этом случае необходимо обеспечить согласованность данных (например, идентичность копий при изменении информации).

Одной из функций компьютерной сети является передача информации, во время которой возможны ее потери и искажения. Для оценки надежности исполнения этой функции используются показатели вероятности потери пакета при его передаче, либо вероятности доставки пакета (передача осуществляется порциями, которые называются пакетами).

В современных компьютерных сетях важное значение имеет другая сторона надежности - *безопасность*. Это способность сети обеспечить защиту информации от несанкционированного доступа. Задачи обеспечения безопасности решаются применением как специального программного обеспечения, так и соответствующих аппаратных средств.

Управляемость

При работе компьютерной сети, которая объединяет отдельные компьютеры в единое целое, необходимы средства не только для наблюдения за работой сети, сбора разнообразной информации о функционировании сети, но и средства управления сетью. В общем случае система управления сетью должна предоставлять возможность воздействовать на работу любого элемента сети. Должна быть обеспечена возможность осуществлять мероприятия по управлению с любого элемента сети. Управлением сетью занимается администратор сети или пользователь, которому поручены эти функции. Обычный пользователь, как правило, не имеет административных прав.

Другими характеристиками управляемости являются возможность определения проблем в работе компьютерной сети или отдельных ее сегментов, выработка управленческих действий для решения выявленных проблем и возможность автоматизации этих процессов при решении похожих проблем в будущем.

Расширяемость и масштабируемость

Любая компьютерная сеть является развивающимся объектом, и не только в плане модернизации ее элементов, но и в плане ее физического расширения, добавления новых элементов сети (пользователей, компьютеров, служб). Существование таких возможностей, трудоемкость их осуществления входят в понятие расширяемости. Другой похожей характеристикой является масштабируемость сети, которая определяет возможность расширения сети без существенного снижения ее производительности. Обычно одноранговые сети обладают хорошей расширяемостью, но плохой масштабируемостью. В таких сетях легко добавить новый компьютер, используя дополнительный кабель и сетевой адаптер, но существуют ограничения на количе тво подключаемых компьютеров в связи с существенным падением производительности сети. В многосегментных сетях используются специальные коммуникационные устройства, которые позволяют подключать к сети значительной количество дополнительных компьютеров без снижения общей производительности сети.

Прозрачность

Прозрачность компьютерной сети является ее характеристикой с точки зрения пользователя. Эта важная характеристика должна оцениваться с разных сторон.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

2.1 Лабораторная работа № 1 (2 часа).

Тема: «Понятия информации.»

2.1.1 Цель работы: Сформировать навыки и умения перевода чисел из любой системы счисления в десятичную

2.1.2 Задачи работы:

1. Перевод чисел из одной позиционной системы счисления в другую.
2. Арифметические операции с числами в позиционных системах счисления

2.1.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. ПК, программа калькулятор
2. Калькулятор

2.1.4 Описание (ход) работы:

1. Перевести данное число из десятичной системы счисления в двоичную:
а) 464(10); б) 380,1875(10); в) 115,94(10) (получить пять знаков после запятой в двоичном представлении).

Решение.

```
    464 | 0      380 | 0    |1875          115 | 1      |94
    232 | 0      190 | 0   0|375           57 | 1     1|88
    116 | 0       95 | 1   0|75            28 | 0     1|76
     58 | 0       47 | 1   1|5             14 | 0     1|52
а)   29 | 1   б)  23 | 1   1|0       в)     7 | 1     1|04
     14 | 0       11 | 1                    3 | 1     0|08
      7 | 1        5 | 1                    1 | 1     0|16
      3 | 1        2 | 0
      1 | 1        1 | 1
```

а) 464(10) = 111010000(2); б) 380,1875(10) = 101111100,0011(2); в) 115,94(10) » 1110011,11110(2) (в настоящем случае было получено шесть знаков после запятой, после чего результат был округлен).

Если необходимо перевести число из двоичной системы счисления в систему счисления, основанием которой является степень двойки, достаточно объединить цифры двоичного числа в группы по столько цифр, каков показатель степени, и использовать приведенный ниже алгоритм. Например, если перевод осуществляется в восьмеричную систему, то группы будут содержать три цифры (8 = 23). Итак, в целой части будем производить группировку справа налево, в дробной — слева направо. Если в последней группе недостает цифр, дописываем нули: в целой части — слева, в дробной — справа. Затем каждая группа заменяется соответствующей цифрой новой системы. Соответствия приведены в таблицах.

P	2	00	01	10	11
4	0	1	2	3	

P	2	000	001	010	011	100	101	110	111
8	0	1	2	3	4	5	6	7	

P	2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Переведем из двоичной системы в восьмеричную число 1111010101,11(2).
001 111 010 101,110(2) = 1725,6(8).

Переведем из двоичной системы в шестнадцатеричную число 1111010101,11(2).
0011 1101 0101,1100(2) = 3D5,C(16).

Соответствие между шестнадцатеричными цифрами и десятичными числами	16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

10 0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15

При переводе чисел из системы счисления с основанием P в десятичную систему счисления необходимо пронумеровать разряды целой части справа налево, начиная с нулевого, и в дробной части, начиная с разряда сразу после запятой слева направо (начальный номер -1). Затем вычислить сумму произведений соответствующих значений разрядов на основание системы счисления в степени, равной номеру разряда. Это и есть представление исходного числа в десятичной системе счисления.

2. Перевести данное число в десятичную систему счисления.

Некоторые неотрицательные степени числа 2 (в десятичной системе счисления)

Показатель	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Степень	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536

Некоторые отрицательные степени числа 2 (в десятичной системе счисления)

Показатель	-1	-2	-3	-4	-5	-6	-7
Степень	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125

а) 1000001(2).

1000001(2)=1× 26+0× 25+0× 24+0× 23+0× 22+ 0× 21+1× 20 = 64+1=65(10).

Замечание. Очевидно, что если в каком-либо разряде стоит нуль, то соответствующее слагаемое можно опускать.

б) 1000011111,0101(2).

1000011111,0101(2)=1×29 + 1×24 + 1×23 + 1×22 + 1×21 + 1×20 + 1×2-2 + 1×2-4 = 512 + 16 + 8 + 4 + 2 + 1 + 0,25 + 0,0625 = 543,3125(10).

Некоторые неотрицательные степени числа 8 (в десятичной системе счисления)

Показатель	0	1	2	3	4
Степень	1	8	64	512	4096

Некоторые отрицательные степени числа 8 (в десятичной системе счисления)

Показатель	-1	-2
Степень	0,125	0,015625

в) 1216,04(8).

1216,04(8)=1×83+2×82+1×81+6×80+4× 8-2 = 512+128+8+6+0,0625 = 654,0625(10).

Некоторые неотрицательные степени числа 16 (в десятичной системе счисления)

Показатель	0	1	2	3	4
Степень	1	16	256	4096	65536

Лабораторная работа 2

Тема: Алгоритмизация вычислительных процессов

2.2.1.Цель работы: Научится составлять простейшие алгоритмы

2.2.2 Задачи работы:

1. Изучить методы составления алгоритмов и их свойства.

2.2.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер.
2. Доска

2.2.4 Описание (ход) работы:

Задание 1. Построить алгоритмы задач указанных преподавателем.

Вычислить значение переменной Y, если для любых значений исходных переменных A и X.

Изучив поставленную задачу, приходим к выводу, что исходной информацией являются переменные A и B, а результат работы программы – значение переменной Y.

Проанализировав формулу расчета Y приходим к выводу, что здесь линейный вычислительный процесс, так как при любых значениях переменных A и X формула имеет смысл и порядок решения задачи единственный:

1) Ввод в память ЭВМ значений переменных A и X.

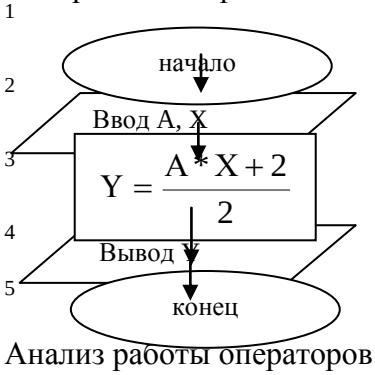
2) Расчет переменной Y.

3) Вывод результата на экран.

Перечень этих пунктов и есть текстовый алгоритм решения задачи.

В контрольной задаче можно рассмотреть только один вариант, например: при A=2 и X=3, Y=4.

Изобразим алгоритм в виде блок–схемы:



Описание алгоритма на языке Бейсик

Блок 2 10 INPUT «введите значения переменных A и X», A, X

Блок 3 20 Y = (A*X + 2) /2

Блок 4 30 PRINT «Y = «Y

40 END

Анализ работы операторов программы.

Строка 10: по оператору ввода INPUT.

1) На экране дисплея высвечивается приглашение к вводу соответствующее символьной константе оператора:

Введите значения переменных A и X:

2) В оперативной памяти компьютера резервируются две ячейки для хранения значений переменных A и X.

3) Приостанавливается работа программы.

Пользователь должен на клавиатуре набрать две константы, являющиеся значениями переменных через запятую и нажать клавишу ENTER.

4) Переменным A и X присваивается значение, то есть в ячейки памяти, зарезервированные для хранения значений этих переменных, запишутся соответствующие

константы. По контрольной задаче, в результате работы оператора Input, получим A=2 и X=3 и программа работает дальше.

Строка 20: по оператору присваивания

1) Резервируется ячейка памяти для хранения значения переменной Y**.**

2) В арифметическое выражение, вместо имен переменных A и X подставляются их значения.

3) Выполняются арифметические операции с учетом их старшинства и получаем значение арифметического выражения.

4) Значение арифметического выражения присваивается переменной Y, то есть в отведенную для переменной Y ячейку памяти, записывается константа, являющаяся значением арифметического выражения.

Строка 30: По оператору вывода PRINT на экране выводится символьная константа Y= и затем значение переменной Y. Таким образом, по контрольной задаче на экране должно появиться изображение Y=4.

Для закрепления материала, сидя за компьютером, отредактируйте этот текст программы:

1) Описать ввод значений A и X двумя операторами INPUT.

2) Записать в программе оператор PRINT так, чтобы на экране в контрольной задаче появился ответ в следующем виде: Y=4 при A=2 и X=3.

Рассмотренная нами первая задача является одноразовой, т.е. по ней можно ввести только по одному значению переменных A и X и получить только одно значение У. Если есть необходимость рассчитать несколько значений У с различными значениями A и X, надо будет каждый ряд запускать программу заново. (Запускать клавишей F2 , а не вводить текст заново с клавиатуры).

Рассчитать Y=(AX+2)/2 для известного количества любых значений A и X, то есть программа должна быть запущена один раз, а получить значений Y столько, сколько нужно. Например, необходимо рассчитать четыре значения переменной Y, со следующими значениями A и X: A=2, -1, 4, 0

X=3, 6, 1, 9

тогда мы должны получить значения Y=4, -2, 3, 1

№ п/п	A	X	Y
1	2	3	4
2	-1	6	-2
3	4	1	3
4	0	9	1

поскольку мы не изменили формулу расчета Y, то порядок решения задачи не изменится: 1) ввод значений A, X

2) расчет Y

3) вывод Y

Но эти три операции необходимо повторить четыре раза. Мы знаем, что если какая то группа операций повторяется, то мы имеем циклический процесс. На первых порах не пытайтесь разработать алгоритм циклического процесса наскоком. Действуйте постепенно, опираясь на теорию, и начните с того, что вспомните классификацию циклических процессов с тем, чтобы определиться, какой циклический процесс организовать в данной задаче:

Так как в задаче № 2 четко сказано, что количество значений переменных A и X известно, то логично сделать вывод: необходимо организовать явный циклический процесс по счетчику.

Итак, исходные данные для решения этой задачи: количество значений A, X и сами их значения.

Введем условные обозначения:

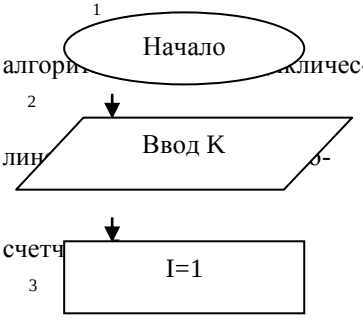
K - количество значений переменных A, X и Y

I – порядковый номер значения переменных

В контрольной задаче K=4 и I=1, 2, 3, 4

В общем случае I=1, 2, …. K

Изобразим алгоритм решения задачи в виде блок – схемы:

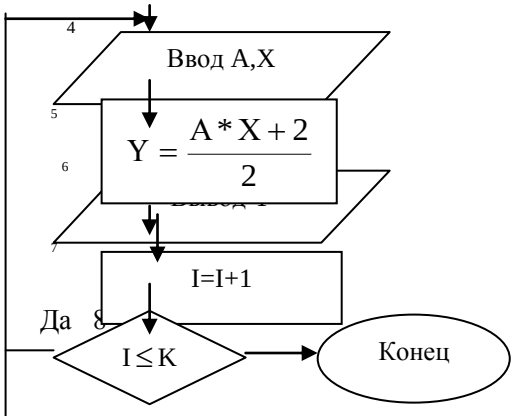


Структура алгоритма – циклический процесс с линейной организацией, организованный по счетчику

Вход в цикл – блок 3

Тело цикла – блоки 4, 5, 6

Выход из цикла – блоки 7, 8



Анализ работы алгоритма по контрольной задаче:

Номера блоков	Результат работы блоков
2	K=4
3	I=1
4	A=2, X=3
5	Y=4
6	Y=4
7	I=1+1=2

A=-1,X=6 A=4, X=1 A=0,X=9

Y=-2 Y=3 Y=1

Y=-2 Y=3 Y=1

I=2+1=3 I=3+1=4 I=4+1=5

8 2≤4? Да ——3≤4? Да 4≤4? Да — 5≤4? Нет—

Выход

из цикла

При анализе работы алгоритма ваше внимание постоянно должно быть обращено к контрольной задаче.

На вставочке «нет» из восьмого блока мы оказались при I=5. Опишем этот алгоритм с помощью операторов языка Бейсик двумя способами.

I способ (с помощью операторов переходов):

```
Бл.2    10 INPUT «Количество значений A, X»; K
Бл.3    20 I=1
Бл.4    30 INPUT «Введите значения A, X»; A, X
Бл.5    40 Y=(A*X+2)/2
Бл.6    50 PRINT «Y=»;Y
Бл.7    60 I=I+1
Бл.8    70 IF I≤K THEN 30
Бл.9    80 END
```

Проанализируем работу некоторых операторов программы по контрольной задаче:

Строка 10: В результате работы оператора INPUT исходная переменная К получает значение К=4.

Строка 20: В результате работы оператора присваивания переменная I получает значение равные 1.

Строка 30: В результате операции ввода переменные A и X получают значения.

Строка 60: Происходит переприсваивание значения переменной I на 1 (старое значение стирается из ячейки памяти).

Строка 70: Работает оператор условного перехода: проверяется условие I≤ K?, если условие выполняется, то работает оператор безусловного перехода на строку с номером 30, то есть на начало тела цикла, иначе переход на следующую строку, то есть – выход из цикла.

2.3 Лабораторная работа 3

Тема: Массивы. Обработка одномерных массивов на АЯ высокого уровня.

2.3.1.Цель работы: Научится составлять простйшие программы на АЯ высокого уровня

2.3.2 Задачи работы:

1. Изучить основные операторы АЯ.

2.3.3 Перечень приборов, материалов, используемых в лабораторной работе:

1. Персональный компьютер.

2.3.4 Описание (ход) работы:

Задание 1. Реализовать алгоритмы на АЯ высокого уровня, рассмотренные на ЛР-8,9.

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

3.1 Практическое занятие №1 (2 часа)

Тема: «Алгоритмизация вычислительных процессов»

3.1.1 Задание для работы:

1. Построение алгоритмов различной структуры.

3.2.2 Краткое описание проводимого занятия:

1. Понятие алгоритмизации вычислительных процессов

Разработке алгоритма предшествуют такие этапы, как формализация и моделирование задачи. Формализация предполагает замену словесной формулировки решаемой задачи краткими символьными обозначениями, близкими к обозначениям в языках программирования или к математическим. Моделирование задачи является важнейшим этапом, целью которого является поиск общей концепции решения. Обычно моделирование выполняется путем выдвижения гипотез решения задачи и их проверке любым рациональным способом (прикидочные расчеты, физическое моделирование и т.д.). Результатом каждой проверки является либо принятие гипотезы, либо отказ от нее и разработка новой.

При разработке алгоритма используют следующие основные принципы.

Принцип поэтапной детализации алгоритма (другое название - "проектирование сверху-вниз"). Этот принцип предполагает первоначальную разработку алгоритма в виде укрупненных блоков (разбиение задачи на подзадачи) и их постепенную детализацию.

Принцип "от главного к второстепенному", предполагающий составление алгоритма, начиная с главной конструкции. При этом, часто, приходится "достраивать" алгоритм в обратную сторону, например, от середины к началу.

Принцип структурирования, т.е. использования только типовых алгоритмических структур при построении алгоритма. Нетиповой структурой считается, например, циклическая конструкция, содержащая в теле цикла дополнительные выходы из цикла. В программировании нетиповые структуры появляются в результате злоупотребления командой безусловного перехода (GoTo). При этом программа хуже читается и труднее отлаживается.

Говоря о блок-схемах, как о средстве записи алгоритма, можно дать еще один совет по их разработке. Рекомендуется после внесения исправлений в блок-схему аккуратно перерисовывать ее с учетом этих исправлений. Аккуратность записи есть аккуратность мысли программиста. Аккуратно записанный и детализованный алгоритм упрощает его программирование и отладку.

2. Понятие алгоритма и его свойства, способы описания алгоритмов

"Алгоритм" является фундаментальным понятием информатики. Представление о нем необходимо для эффективного применения вычислительной техники к решению

практических задач. Алгоритм - это предписание исполнителю (человеку или автомату) выполнить точно определенную последовательность действий, направленных на достижение заданной цели. Алгоритм - это сформулированное на некотором языке правило, указывающее на действия, последовательное выполнение которых приводит от исходных данных к искомому результату. Значение слова алгоритм очень схоже со значением слов рецепт, процесс, метод, способ. Однако любой алгоритм, в отличие от рецепта или способа, обязательно обладает следующими свойствами.

Свойства алгоритма (отличающие его от любых других предписаний): понятность (для конкретного исполнителя); дискретность (команды последовательны, с точной фиксацией моментов начала и конца выполнения команды); точность (после выполнения каждой команды точно известно, завершено ли исполнение алгоритма или же какая команда должна выполниться следующей); результативность (после конечного числа шагов задача решается или же становится ясно, что процесс решения не может быть продолжен): массовость (алгоритм единым образом применяется к любой конкретной формулировке задачи, для которой он разработан).

1. Дискретность - разбиение алгоритма на ряд отдельных законченных действий - шагов. Выполнение алгоритма разбивается на последовательность законченных действий - шагов. Каждое действие должно быть закончено исполнителем алгоритма прежде, чем он приступит к исполнению следующего действия.

2. Точность - однозначные указания. На каждом шаге однозначно определено преобразование объектов среды исполнителя, полученной на предыдущих шагах алгоритма. Если алгоритм многократно применяется к одному и тому же набору исходных данных, то на выходе он получает каждый раз один и тот же результат. Запись алгоритма должна быть такой, чтобы на каждом шаге его выполнения было известно, какую команду надо выполнять следующей.

3. Понятность - однозначное понимание и исполнение каждого шага алгоритма его исполнителем. Алгоритм должен быть записан на понятном для исполнителя языке.

4. Результативность - обязательное получение результата за конечное число шагов. Каждый шаг (и алгоритм в целом) после своего завершения дает среду, в которой все объекты однозначно определены. Если это по каким-либо причинам невозможно, то алгоритм должен сообщать, что решение задачи не существует. Работа алгоритма должна быть завершена за конечное число шагов. Информатика оперирует только с конечными объектами и конечными процессами, поэтому вопрос о рассмотрении бесконечных алгоритмов остается за рамками теории алгоритмов.

5. Массовость - применение алгоритма к решению целого класса однотипных задач.

Порядок выполнения алгоритма:

- Действия в алгоритме выполняются в порядке их записи
- Нельзя менять местами никакие два действия алгоритма
- Нельзя не закончив одного действия переходить к следующему

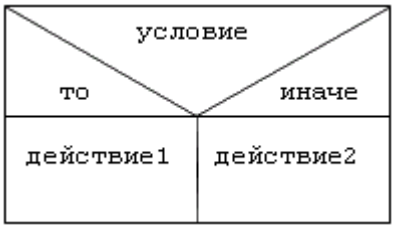
Для записи алгоритмов используются специальные языки:

1. Естественный язык (словесная запись)
2. Формулы
3. Псевдокод
4. Структурограммы
5. Синтаксические диаграммы
6. Графический (язык блок-схем)

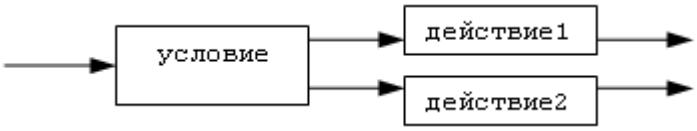
1. Естественный язык:

если условие то действие1 иначе действие2

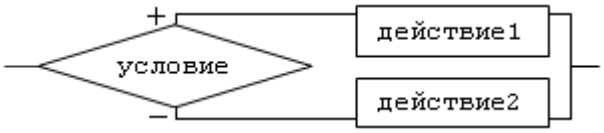
2. Структурограмма:



3. Синтаксическая диаграмма:



4. Графический язык:



Составление алгоритмов графическим способом подчиняется двум ГОСТам:

1. ГОСТ 19.002-80, соответствует международному стандарту ИСО 2636-73. Регламентирует правила составления блок-схем.

2. ГОСТ 19.003-80, соответствует международному стандарту ИСО 1028-73. Регламентирует использование графических примитивов.

Название	Символ (рисунок)	Выполняемая функция (пояснение)
1. Блок вычислений		Выполняет вычислительное действие или группу действий
2. Логический блок		Выбор направления выполнения алгоритма в зависимости от условия
3. Блоки ввода/вывода		Ввод или вывод данных вне зависимости от физического носителя
		Вывод данных на печатающее устройство
4. Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
5. Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
6. Блок модификации		Выполнение действий, изменяющих пункты алгоритма
7. Соединитель		Указание связи между прерванными линиями в пределах одной страницы
8. Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Правила построения блок-схем:

Блок-схема выстраивается в одном направлении либо сверху вниз, либо слева направо

Все повороты соединительных линий выполняются под углом 90 градусов

i. Задания к занятию

Построить алгоритмы решения задач

1. Даны три действительных числа a, b, c. Вывести на экран те, которые принадлежат интервалу (x,y).

2. Даны два действительных числа x и y. Если и x и y отрицательные, то каждое значение заменить его модулем; если отрицательное только одно из них, то x и y увеличить на 0.5; если оба неотрицательные, то x и y увеличить в 10 раз.

3. Известны время старта и время финиша двух лыжников. Если один из лыжников выигрывает, то он получает поощрительный приз и слова поздравления. А проигравший лыжник ничего не выигрывает.

4 В книжном магазине продажей газет занимается ЭВМ. Составить программу, запрашивающую стоимость товара и сумму денег, внесенных покупателем. Определить причитающую сдачу, или выдать сообщение, что денег внесено недостаточно или достаточно.

5. Составить программу, запрашивающую имя и год рождения пользователя. ЭВМ должна выдать сообщение, имеете ли вы право голоса на выборы. Если нет, то выдать сообщение, через сколько лет вы это право получите.

6. Даны два действительных числа a и b. Если числа не равны, то заменить каждое из них числами равными (a+b) и (a-b) соответственно, а если равны, то заменить их нулями.

Протабулировать функции:

7. $K = a * b^3 * c + \frac{\sqrt{c-4}}{b+4}$, где c изменяется на заданном интервале с заданным шагом, a,b – любые значения.

8. $R = x^2 - \sqrt{b + 2*c} + \frac{1}{c}$, где x изменяется на заданном интервале с заданным шагом, c,b – любые значения.

9. $Z = a * x^a + \cos\sqrt{(x+b)}$, где x изменяется на заданном интервале с заданным шагом, a,b –любые значения.

10. $W = a * b * c^2 + \frac{\sqrt{a-2}}{c}$, где a изменяется на заданном интервале с заданным шагом, c,b –принимают одно любое значение.

11. $T = \sqrt{x^2 - 12} + \frac{y-x}{2*y}$, где x изменяется на заданном интервале с заданным шагом, y – одно любое значение.

12. $Q = 10 * a + \sqrt[3]{b-c} - \frac{1}{c}$, где a изменяется на заданном интервале с заданным шагом, c,b –любые значения.

13. $$M=\begin{cases}\sqrt{x}+a*x^3, & \text{если x>0}\\ \dfrac{a*x^2+1}{a-2}, & \text{если x<=0}\end{cases}$$

при $x=\sqrt{y+4}-10$, где y изменяется на заданном интервале с заданным шагом, a - принимает одно любое значение, x=9.2.

14. $$B=\begin{cases}13.7*a^3+z*\sqrt{a+z}, & \text{если z>=0}\\ \dfrac{1}{\sqrt{a-\sqrt[3]{z}}}+z^2, & \text{если z<0}\end{cases}$$

при z=x+6 , где x изменяется на заданном интервале с заданным шагом, a – любые значения.

15. $$Z=\begin{cases}\dfrac{4*d\sqrt{b}}{7*a+d^2}-x, & \text{если x>=0}\\ a+x*d-1/b, & \text{если x<0}\end{cases}$$

при $x=(y-2)^2$, где y изменяется на заданном интервале с заданным шагом, a, b, d – любые значения.

16. $$Y=\begin{cases}a*b+1.4*z, & \text{если z=0}\\ z+\dfrac{b*\sqrt{a+2}}{1+z^2}, & \text{если z>0}\\ \sqrt{a+2}-10*a+1/b, & \text{если z<0}\end{cases}$$

при z=2*x-4, где x изменяется на заданном интервале с заданным шагом, a , b – любые значения.

17. $$D=\begin{cases}\sqrt{b}+a*x^3, & \text{если x>0}\\ \dfrac{b*a^2}{a+2}-x, & \text{если x=0}\\ x-\sqrt{x-a}, & \text{если x<0}\end{cases}$$

при x=10-4*y, , где y изменяется на заданном интервале с заданным шагом, a, b – любые значения.

18. Q= $\begin{cases} 10*a*b+1/z, & \text{если } a>=0 \\ \dfrac{1}{a^2*b}+\sqrt[3]{z}, & \text{если } a<0 \end{cases}$

при a=5-2*x, , где x изменяется на заданном интервале с заданным шагом, b, z – любые значения.

19. P= $\begin{cases} x-2*y+4, & \text{если } x<=0 \\ 1/y+2*y+x/3, & \text{если } x>0 \end{cases}$

при x=$\sqrt{2+z}-4$, где z изменяется на заданном интервале с заданным шагом, y – одно любое значение.

20. B= $\begin{cases} \dfrac{b*x^2}{a+2}+15, & \text{если } x<=3 \\ \sqrt{x-a}+a*b, & \text{если } x>3 \end{cases}$

при x=2*y+3, где y изменяется на заданном интервале с заданным шагом, a, b – любые значения.

21. S= $\begin{cases} \dfrac{1}{a*x^4}+b^3, & \text{если } x<3 \\ \sqrt[3]{\sqrt{2*a-b}}-4, & \text{если } x>=3 \end{cases}$

при x=$\frac{1}{y}+9$, где y изменяется на заданном интервале с заданным шагом, a, b – любые значения.

.22. K= $\begin{cases} 6*a^3+\dfrac{2*a-b}{b}, & \text{если } t<=0.1 \\ t^2-\sqrt{a+b}, & \text{если } t>0.1 \end{cases}$

при t=$\sqrt{x}+4$, где x изменяется на заданном интервале с заданным шагом, a , b – любые значения.

23. R= $\begin{cases} x*y*z^2+\sqrt{x+y}, & \text{если } z>0 \end{cases}$

$$\frac{2+z}{\sqrt[3]{x*y-z}}-\frac{4*x-y}{3}$$, если z<=0

где Z изменяется на заданном интервале с заданным шагом, x-любые значения, y – одно любое значение

24. Дано количество студентов. Ввести имена студентов и их год рождения. Определить, через сколько лет они получат право голоса или сделать вывод, что они имеют право голоса.

25. Рассчитать $y=\frac{a*b+2}{4*b}$ для известного количества любых значений a и b.

26. Рассчитать $S=\sqrt{2*a-3*b}+4$ для известного количества любых значений a и b.

27. Рассчитать $Q=\frac{b*x^2}{\ell^x-1}$ для известного количества любых значений x и b.

28. Рассчитать все возможные значения $x=\frac{2*a-b^3}{2}$, если a изменяется на заданном интервале с заданным шагом, b изменяется на заданном интервале с заданным шагом.

29. Рассчитать все возможные значения $f=\frac{(a+b)^2-4}{2}$, если a изменяется на заданном интервале с заданным шагом, b изменяется на заданном интервале с заданным шагом.

30. Рассчитать все возможные значения $k=\frac{4*x-(y-2)}{3}$, если x изменяется на заданном интервале с заданным шагом, y принадлежит известное любое количество значений.

31. Рассчитать все возможные значения $p=\frac{(a+b)^2-12}{3}$, если a изменяется на заданном интервале с заданным шагом, b изменяется на заданном интервале с заданным шагом.

32. Сохранить в памяти ЭВМ индексы элементов массива C(N), значения которых попадают в интервал от A1до A2.

33. Рассчитать $T = \sqrt{x^2 - 12} + \frac{y - x}{2 * y}$ для известного количества любых значений y и x.Сохранить в памяти ЭВМ только положительные значения T.

34. Рассчитать $X = a^2 - \sqrt{b + 4} - \frac{b}{a}$ для известного количества любых значений a и b.Сохранить в памяти ЭВМ только отрицательные значения X.

35. Дано: список фирм и их уставной фонд. Найти сумму уставных фондов фирм, стоящих на нечетных местах.

36. Дано: список фирм и их уставной фонд. Найти сумму фондов, попадающих в интервал от B1до B2.

37. Дано: список фирм и их уставной фонд. Подсчитать количество фирм, уставной фонд которых выше K, или вывести сообщение, что таких фирм нет.

38. Подсчитать сумму и количество элементов, попадающих в интервал от B1до B2 в массиве A(N).

39. Даны массивы A(N) и B(N). Найти разность значений A и B и переписать в массив C. В массиве C(N) упорядочить элементы по возрастанию.

40. Дано: список фирм и их уставной фонд. Вывести название 3 фирм с наибольшим уставным фондом.

41. Удалить первый отрицательный элемент в массиве B(K) или сделать вывод, что отрицательных элементов в массиве нет.

42. В массиве B(V) исключить все элементы с отрицательными значениями.

43. Вычислить сумму положительных элементов каждой строки матрицы A(M,N).

44. Переписать элементы двумерного массива в одномерный.

45. Вычислить сумму элементов главной диагонали матрицы B(K,L).

46. Задан массив A(M,N). Сформировать новый массив B по формуле: B(I,J)=I*J*A(I,J).

47. Определить количество положительных и отрицательных элементов матрицы A(K,L).

48. Сформировать одномерный массив B из положительных элементов главной диагонали матрицы A(M,N).

49. Вычислить сумму отрицательных элементов по второй диагонали матрицы C(Q,Z).

50. Найти наибольший элемент матрицы V(N,M) и номер строки и столбца, на пересечении которых он стоит.

51. Сформировать одномерный массив из элементов матрицы D(K,L), попадающих в промежуток от A до B.

52. Определить, если в массиве M(A,B) отрицательный элемент.

53. Определить, есть ли в массиве K(L,A) строка, состоящая только из положительных элементов.

54. Определить, есть ли в массиве S(K,L) столбец, состоящий только из отрицательных элементов.

55. Определить, есть ли в массиве T(A,B) столбец, состоящий только из элементов, принадлежащих промежутку от K до L.

56. В матрице X(A,B) все ненулевые элементы заменить противоположными по знаку.

57. Матрица K(M,N) состоит из единиц и нулей. Найти в ней номера (индексы) хотя бы одной строки, не содержащих единицы, либо сообщить, что таковых нет.

58. Матрица K(M,N) состоит из единиц и нулей. Найти в ней номера (индексы) хотя бы одного столбца, не содержащих нулей, либо сообщить, что таковых нет.

59. Найти наибольшие элементы каждого столбца матрицы M(A,B) и записать их в массив N.

60. Поменять местами строки массива, содержащие max и min элементы массива B(K,L). Если эти элементы находятся в одной строке – выдать об этом сообщение.