1 курс

ПЛАН – КОНСПЕКТ
проведения вводного занятия по теме 3.4
по дисциплине «Информатика»

Раздел 3. «Информационное моделирование.»

Тема 3.4:
«Понятие алгоритма и основные алгоритмические структуры.»

Подготовил: преподаватель
В.Н. Борисов

Рязань 2024

Тема №3.4. «Понятие алгоритма и основные алгоритмические структуры»

Цели занятий: изучить со студентами основные сведения об алгоритмах, алгоритмических конструкциях: понятие алгоритма, свойства алгоритма, способы записи алгоритма, основные алгоритмические структуры, запись алгоритмов на языке программирования, анализ алгоритмов с помощью трассировочных таблиц.

Виды занятий: классно-групповые, комбинированные (по проверке знаний, умений по пройденному материалу, по изучению и первичному закреплению на практике нового материала).

Метод проведения занятий: практические занятия.

Время проведения практических занятий: 6 ч (3 занятия по 2 часа)

Основные вопросы:

1. Технологии обработки информации.
2. Этапы подготовки и решения задач на ВТ.
3. Понятие алгоритма. Свойства алгоритма. Способы записи алгоритма.
4. Основные алгоритмические структуры.
5. Запись алгоритмов на языке программирования.
6. Расчет результатов выполнения алгоритма. Анализ алгоритмов с помощью трассировочных таблиц.

Литература:

1. [5 учебник раздела «Дополнительной учебной литературы» рабочей программы изучения дисциплины]: Трофимов, В. В. Информатика в 2 т. Том 2 : учебник для среднего профессионального образования / В. В. Трофимов. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 406 с. — (Профессиональное образование). — ISBN 978-5-534-02519-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/513266, главы 21-24;
2. Гаврилов, М. В. Информатика и информационные технологии : учебник для среднего профессионального образования / М. В. Гаврилов, В. А. Климов. — 5-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 355 с. — (Профессиональное образование). — ISBN 978-5-534-15930-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/viewer/informatika-i-informacionnye-tehnologii-510331#page/1, главы 2,4 ;
3. 5 учебник раздела «Основной учебной литературы» рабочей программы изучения дисциплины: Босова, Л. Л. Информатика. 11 класс. Базовый

уровень : учебник / Л.Л. Босова, А. Ю. Босова. — М. : БИНОМ. Лаборатория знаний, 2022. — 200 с. , ISBN 978-5-9963-3142-0, § 5-7 главы 2.

Примерный расчет времени (по каждому практическому занятию):

1. Вступительная часть – 20 мин.
2. Основная часть – 60 мин.
3. Заключительная часть – 10 мин.

Вступительная часть:

Занятия начать с объявления темы занятия, основных рассматриваемых вопросов, времени изучения темы (нового материала), перечисление литературы, проведения опроса по изученному ранее (пройденному) материалу.

Основная часть (по каждому занятию):

Первый вопрос: Технологии обработки информации.

В ходе информационного процесса информация, циркулирующая на предприятии или в организации, подвергается той или иной обработке в зависимости от рода их деятельности. По месту возникновения выделяют входящую и выходящую, внутреннюю и внешнюю информацию. В процессе обработки информация может быть первичной и вторичной, промежуточной и результатной, при этом обрабатываемые данные преобразуются из одного вида в другой. По мере развития информационного общества трудозатраты на обработку данных возрастают и требуют совершенствования применяемых технологий.

Технология (гр. techne – мастерство, logos – учение, учение о мастерстве) – совокупность знаний о способах и средствах производственных процессов, при которых происходит необходимое качественное изменение обрабатываемых объектов.

Обработка информации — вся совокупность операций (сбор, ввод, запись, преобразование, считывание, хранение, уничтожение, регистрация), осуществляемых с помощью технических и программных средств, включая обмен по каналам передачи данных. При современном развитии программного обеспечения существует множество различных программных средств обработки информации, написанных на разных языках программирования на основе выше перечисленных методов.

Обработка информации подразумевает переработку информации определённого типа (текстовой, звуковой, графической и др.) и преобразования её в информацию другого определённого типа. Так, например, принято различать обработку текстовой информации, изображения (графики, фото, видео и мультипликация) и звуковой информации (речь, музыка, другие

звуковые сигналы). Использование новейших технологий обеспечивает их комплексное представление. При этом человеческое мышление может рассматриваться как процесс обработки информации.

Технологией обработки информации называют взаимосвязанные действия, выполняемые в строго определённой последовательности с момента возникновения информации до получения заданных результатов.

Информационная технология обработки предназначена для решения хорошо структурированных задач, по которым имеются необходимые входные данные, известны алгоритмы и другие стандартные процедуры их обработки.

Определение информационных технологий

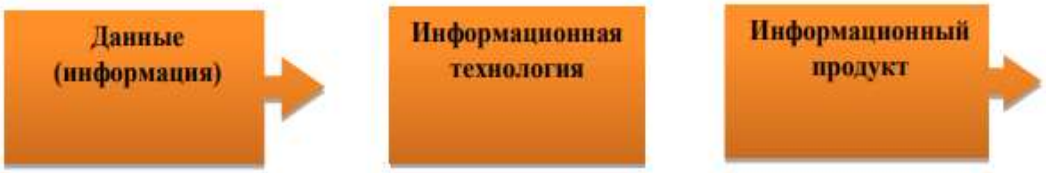
Создание и функционирование информационных систем в управлении экономикой тесно связано с развитием информационных технологий, их главней составной частью.

Процесс определяется выбранной стратегией и реализуется совокупностью различных средств и методов. Технология изменяет качество или первоначальное состояние материи в целях получения материального продукта.

Информация является одним из ценнейших ресурсов общества наряду с традиционными материальными ресурсами: нефтью, газом, полезными ископаемыми и пр.

Значит, процесс ее переработки - информационный процесс по аналогии с процессами переработки материальных ресурсов называется технологией (рисунок 2).

Цель технологии материального производства - выпуск продукции, удовлетворяющей потребности человека или системы.



Цель информационной технологии - производство информации для ее анализа человеком и принятия на его основе решения по выполнению какого- либо действия.

Рисунок 2 - Информационная технология как аналог технологии переработки материальных ресурсов

Информационные технологии в управлении - это комплекс методов переработки разрозненных исходных данных в надежную и оперативную информацию, механизма принятия решений с помощью аппаратных и программных средств с целью достижения оптимальных рыночных параметров объекта управления.

Автоматизированные информационные технологии - это системно-организованная для решения задач управления совокупность методов и средств реализации операций сбора, регистрации, передачи, накопления, поиска, обработки и защиты информации на базе применения развитого программного обеспечения, используемых средств вычислительной техники и связи, а так же способов, с помощью которых информация предлагается клиентам.

Инструментарий информационной технологии - один или несколько взаимосвязанных программных продуктов для определенного типа компьютера, технология работы в котором позволяет достичь поставленную пользователем цель.

В качестве инструментария используются: текстовый процессор (редактор), настольные издательские системы, электронные таблицы, системы управления базами данных, электронные записные книжки, электронные календари, информационные системы функционального назначения (финансовые, бухгалтерские, для маркетинга и пр.), экспертные системы и др.

Информационная технология тесно связана с информационными системами, которые являются для нее основной средой.

Информационная система является средой, составляющими элементами которой являются компьютеры, компьютерные сети, программные продукты, базы данных, люди, различного рода технические и программные средства связи и пр., т. е. это человеко - компьютерная система обработки информации, основная цель которой организация хранения и передачи информации.

Реализация функций информационной системы невозможна без знания ориентированной на нее информационной технологии. Информационная технология может существовать и вне сферы информационной системы.

Для реализации этапов технологического процесса могут использоваться разные программные среды.

Информационная технология, как и любая другая, должна обеспечивать высокую степень расчленения всего процесса обработки информации на этапы (фазы), операции, действия и включать весь набор элементов, необходимых для достижения поставленной цели.

Классификация автоматизированных информационных технологий

Развитие автоматизированных информационных технологий шло параллельно с появлением новых видов технических средств обработки и передачи информации, совершенствованием организационных форм использования ЭВМ, созданием новых средств коммуникаций.

Современные автоматизированные информационные технологии классифицируются по ряду признаков.

1. По способу реализации:

- традиционно сложившиеся информационные технологии использовались в условиях централизованной обработки данных и были ориентированы главным образом на снижение трудоемкости при формировании регулярной отчетности;
- новые информационные технологии связаны с информационным обеспечением процесса управления в режиме реального времени.

2. По степени охвата задач управления:

- электронная обработка данных на ЭВМ по решению отдельных экономических задач без пересмотра методологии и организации процессов управления;
- автоматизация управленческой деятельности - ЭВМ используется для комплексного решения функциональных задач, формирования регулярной отчетности и работы в информационно-справочном режиме для подготовки управленческих решений;
- информационные технологии поддержки принятия решения, предусматривающие широкое использование экономико-математических методов, моделей и прикладных программных продуктов для аналитической работы и формирования прогнозов, составления бизнес-планов, обоснованных оценок и выводов по изучаемым процессам и явлениям производственно-хозяйственной практики;
- информационные технологии электронного офиса, ориентированные на использование последних достижений в области интеграции новейших подходов к автоматизации работы специалистов и руководителей, создание для них наиболее благоприятных условий труда за счет полного автоматизированного набора управленческих процедур;
- информационные технологии экспертной поддержки решений, используемые для автоматизации труда специалистов - аналитиков, которые исследуют ситуации по сбыту продукции, услуг, финансовому положению предприятия, по финансово-кредитной организации.

3. По классам реализуемых технологических операций:

- текстовые процессоры для обработки информации;
- электронные таблицы для автоматизированной обработки данных;

- программные продукты для работы с графической информацией;
- базы и банки данных для обработки больших массивов информации;
- мультимедийные системы, используемые для вывода высококачественного звука и видеоизображения;
- гипертекстовые и другие системы.

4. По типу пользовательского интерфейса:

- пакетные автоматизированные информационные технологии, не позволяющие пользователю участвовать в процессе обработки информации в автоматическом режиме, так как организация обработки данных основана на выполнении программно-заданной последовательности;
- операций над заранее накопленными в системе и объединенными в пакет данными;
- диалоговые автоматизированные информационные системы, предоставляющие пользователю в реальном масштабе времени взаимодействовать с информационными ресурсами, хранящимися в системе;
- сетевые информационные системы, дающие пользователю средства доступа к территориально распределенным информационным и вычислительным ресурсам.

5. По способу построения сети: локальные, многоуровневые и распределенные информационные технологии.

Интерфейс сетевой автоматизированной информационной технологии предоставляет пользователю средства доступа к территориально распределенным информационным и вычислительным ресурсам благодаря развитым средствам связи, что делает их многофункциональными и широко используемыми.

В настоящее время наблюдается тенденция к объединению различных типов информационных технологии в единый компьютерно-технологический комплекс, который называется интегрированным.

При объединении различных типов информационных технологий в единый интегрированный комплекс используются средства коммуникации, которые обеспечивают технологические возможности автоматизации управленческой деятельности и являются основой для создания разно- образных сетевых вариантов информационной технологии.

6. По обслуживаемым предметным областям:

- информационные технологии бухгалтерского учета;
- банковской деятельности;
- налоговой деятельности;
- страховой деятельности и др.

Второй вопрос: <u>Этапы подготовки и решения задач на вычислительной технике (ВТ).</u>

На ЭВМ могут решаться задачи различного характера, например: научно-технические; управления производственными процессами; разработки системного, программного обеспечения; обучения и др. Значительную долю в указанном перечне составляют научно-технические задачи. В процессе подготовки и решения их на ЭВМ можно выделить следующие этапы:

- постановка задачи;
- математическое описание задачи;
- выбор и обоснование метода решения;
- алгоритмизация вычислительного процесса;
- составление программы;
- отладка программы;
- решение задачи на ЭВМ и анализ результатов.

В задачах другого класса некоторые этапы могут отсутствовать, например, в задачахразработки системного программного обеспечения отсутствует математическое описание и т. д.

Перечисленные этапы связаны друг с другом. Например, анализ результатов может показать необходимость внесения изменений в программу, алгоритм или даже в постановку задачи. Для уменьшения числа подобных изменений необходимо на каждом этапе по возможности учитывать требования, предъявляемые последующими этапами.

В некоторых случаях связьмежду различными этапами, например, между постановкой задачи и выбором метода решения, между составлением алгоритма и программированием, может быть настолько тесной, что разделение их становится затруднительным.

Постановка задачи.

Математическое описание задачи.

Настоящий этап характеризуется математической формализацией задачи, при которой существующие соотношения между величинами, определяющими результат, выражаются посредством математических формул.

Так формируется математическая модельявления с определенной точностью, допущениями и ограничениями. При этом в зависимости от специфики решаемой з задачи могут быть использованы различные разделы математики и других дисциплин. Построить математическую модель решаемой задачи – это значит представить в математической форме все ее существенные свойства, выделенные при постановке.

Математическая модель должна удовлетворять требованиям реалистичности, реализуемости и корректности. Под реалистичностью понимается правильное отражение модельюнаиболее существенных черт исследуемого явления. Реализуемость достигается разумной абстракцией, отвлечением от второстепенных деталей, сведением задачи к проблеме с известным решением.

Условием реализуемости является возможность практического выполнения необ ходимых вычислений за отведенное время при доступных затратах требуемых ресурсов.

Корректность модели предполагает некоторые изменения выходных результатов при незначительном изменении входных данных.

Выбор и обоснование метода решения.

Модель решаемой задачи с учетом ее особенностей должна быть доведена до решения при помощи конкретныхметодов решения. Само по себе математической описание задачи в большинстве случаев трудно перевести на язык машины. Выбор и использование метода решения задачи позволяет привести решение задачи к конкретным машинным операциям. При обосновании выбора метода необходимо учитывать различные факторы и условия, в том числе точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти и другие.

Одну и ту же задачу можно решить различными методами, при этом в рамках каждого метода можно составить различные алгоритмы.

Алгоритмизация вычислительного процесса.

На данном этапе составляется алгоритм решения задачи согласно действиям, за даваемым выбранным методом решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные блоки и устанавливается последовательность выполнения блоков. Разрабатывается блок-схема алгоритма.

Составление программы.

При составлении программы алгоритм решения задачи переводится на конкретный язык программирования. Для программирования обычно используются языки высокого уровня, поэтому составленная программа требует перевода ее на машинный язык ЭВМ. После такого перевода выполняется уже соответствующая машинная программа.

Отладка программы.

Отладка заключается в поискеи устранении синтаксических и логических (семантических, алгоритмических) ошибок в программе.

В ходе синтаксического контроля программы транслятором выявляются конструкции и сочетаниясимволов, недопустимые с точки зрения правил их построения или написания, принятых в данном языке. Сообщения об ошибках ЭВМ

выдает программисту, при этом вид и форма выдачи подобных сообщений зависят от типа языка и версии используемого транслятора. После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными. Для этого используются специальные методы, например, в программе выбираются контрольные точки, для которых вручную рассчитываются промежуточные результаты. Эти результаты сверяются со значениями, получаемыми ЭВМ в данных точках при выпол-

нении отлаживаемой программы. Кроме того, для поиска ошибок могут быть использованы отладчики, выполняющие специальные действия на этапе отладки, например, удаление, замена или вставка отдельных операторов или целых фрагментов программы, вывод или изменение значений заданных переменных.

Решение задачи на ЭВМ и анализ результатов.

После отладки программу можно использовать для решения прикладной задачи. При этом обычно выполняется многократное решение задачи на ЭВМ для различных наборов исходных данных. Получаемые результаты интерпретируются и анализируются специалистом или пользователем, поставившим задачу.

Разработанная программа длительного использования устанавливается на ЭВМ, как правило, в виде готовой к выполнению машинной программы. К программе приалагается документация, включая инструкцию для пользователя.

Часто при установке программы на диск для ее последующего использования устанавливаются, помимо файлов с исполняемым кодом, различные вспомогательные программы (утилиты, справочники, настройщики и т. д.), а также необходимые для работы программ разного рода файлы с текстовой, графической, звуковой и другой информацией.

Третий вопрос: Понятие алгоритма. Свойства алгоритма. Способы записи алгоритма.

Понятие и свойства алгоритмов.

Алгоритм – *заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.*

Это – не определение в математическом смысле слова, а, скорее, описание интуитивного понятия алгоритма, раскрывающее его сущность.

Понятие алгоритма является не только одним из главных понятий математики, но одним из главных понятий современной науки.

Основные **свойства алгоритмов** следующие:

1. **Понятность** для исполнителя – исполнитель алгоритма должен понимать, как его выполнять. Иными словами, имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

2. **Дискретность** (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

3. **Определенность** – каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

4. **Результативность** (или конечность) состоит в том, что за конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из–за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.

5. **Массовость** означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

Способы описания (записи) алгоритма.

На практике наиболее распространены следующие формы представления алгоритмов:

· *словесная* (запись на естественном языке);

· *графическая* (изображения из графических символов);

· *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).

Словесный способ записи алгоритмов 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;

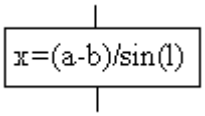
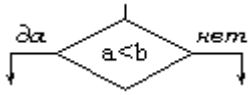
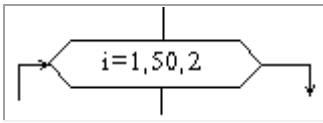
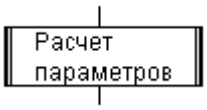
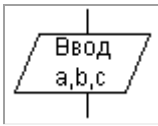
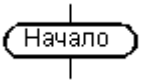
· допускают неоднозначность толкования отдельных предписаний.

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным. *При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.*

Такое 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.

Таблица блочных символов.

В таблице приведены наиболее часто употребляемые символы:

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод–вывод		Ввод–вывод в общем виде
Пуск–останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

Блок **"процесс"** применяется для обозначения действия или последовательности действий, изменяющих значение, форму представления или размещения данных. Для улучшения наглядности схемы несколько отдельных блоков обработки можно объединять в один блок. Представление отдельных операций достаточно свободно.

Блок **"решение"** используется для обозначения переходов управления по условию. В каждом блоке "решение" должны быть указаны вопрос, условие или сравнение, которые он определяет.

Блок **"модификация"** используется для организации циклических конструкций. (Слово модификация означает видоизменение, преобразование). Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.

Блок **"предопределенный процесс"** используется для указания обращений к вспомогательным алгоритмам, существующим автономно в виде некоторых самостоятельных модулей, и для обращений к библиотечным подпрограммам.

Четвёртый вопрос: <u>Основные (базовые) алгоритмические структуры.</u>

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных **базовых** (т.е. основных) **элементов**. Естественно, что при таком подходе к алгоритмам изучение основных принципов их конструирования должно начинаться с изучения этих базовых элементов. Для их описания будем использовать язык схем алгоритмов и простейший псевдокод.

Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: следование, ветвление, цикл. Характерной особенностью базовых структур является наличие в них **одного входа и одного выхода.**

1.***Базовая структура "следование".*** Образуется последовательностью действий, следующих одно за другим:

Псевдокод	Язык блок–схем
действие 1 действие 2 . . . . . действие n	

2.***Базовая структура "ветвление"***. Обеспечивает в зависимости от результата проверки условия (**да** или **нет**) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к *общему выходу*, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

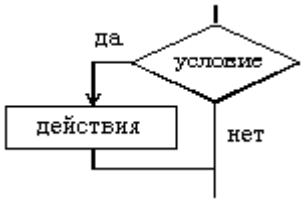
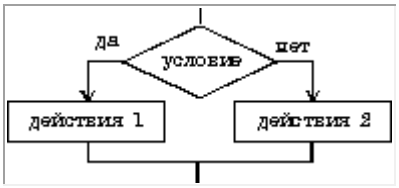
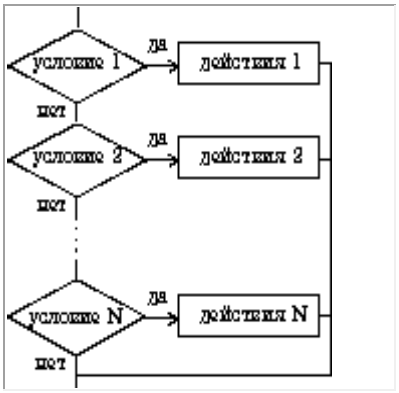
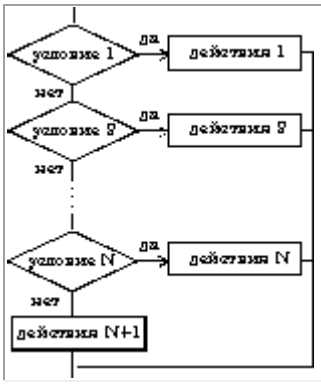
Структура *ветвление* существует в четырех основных вариантах:

· если–то;

· если–то–иначе;

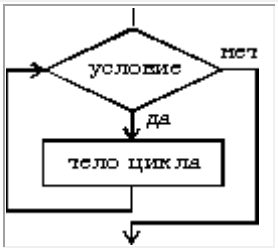
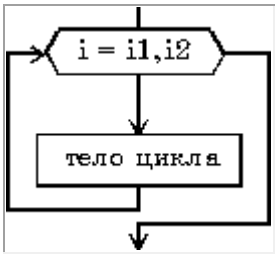
· выбор;

· выбор–иначе.

Псевдокод	Язык блок–схем
1. если–то	
если условие то действия все	
2. если–то–иначе	
если условие то действия 1 иначе действия 2 все	
3. выбор	
выбор при условие 1: действия 1 при условие 2: действия 2 . . . . . при условие N: действия N все	
4. выбор–иначе	
выбор при условие 1: действия 1 при условие 2: действия 2 . . . . . при условие N: действия N иначедействия N+1 все	

3. ***Базовая структура "цикл" (повторение).*** Обеспечивает *многократное выполнение некоторой совокупности действий***,** которая называется *телом цикла*. Основные разновидности циклов представлены в таблице:

Псевдокод	Язык блок–схем
Цикл типа пока. Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока.	

нц пока условие тело цикла (последовательность действий) кц	
Цикл типа для. Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.	
нц для i от i1до i2 тело цикла (последовательность действий) кц	

Пятый вопрос: <u>Запись алгоритмов на языке программирования.</u>

Как мы говорили (вопрос 4 данного План-конспекта), одним из способов описания (записи) алгоритмов является программный, т.е. запись алгоритма на одном из языков программирования.

Язык программирования – это формализованный искусственный язык для записи программ – последовательностей действий на каком-либо устройстве-исполнителе (без вмешательства внешнего естественного или искусственного интеллекта в процесс исполнения).

Примеры языков программирования:

- язык программирования станков с числовым программным управлением;
- язык программирования электронно-вычислительной машины (ЭВМ, компьютера).

Существуют тысячи компьютерных языков программирования, как универсальных по своему назначению и практике применения, так и предметно ориентированных (для научно-технических вычислений, веб-разработки и др.).

Система программирования – это система для разработки новых программ на конкретном языке программирования.

Система программирования обычно включает в себя следующие **компоненты:**

1. Компилятор или интерпретатор.
2. Интегрированная среда разработки.
3. Средства создания и редактирования текстов программ.
4. Библиотеки стандартных программ и функций.
5. Отладочные программы, помогающие находить и устранять ошибки.

6. Диалоговая среда.
7. Многооконный режим работы.
8. Мощные графические библиотеки.
9. Утилиты для работы с библиотеками.
10. Ассемблер.
11. Справочная служба.

Компилятор — это особый вид транслятора, который переводит тексты с языка программирования высокого уровня (с того языка, которым пользуется программист при написании текста программы) на машинный язык (в машинный код, который понятен компьютеру).

Интерпретатор — это исполняемый файл, который поэтапно читает программу, а затем обрабатывает, сразу выполняя ее инструкции. Он осуществляет программу поэтапно как часть собственного исполняемого файла.

Интегрированная среда разработки — это набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы.

Библиотеки стандартных программ и функций состоят из совокупности подпрограмм, составленных на одном из языков программирования и удовлетворяющих определенным единым требованиям к структуре, организации их входов и выходов, описаниям подпрограмм.

Важным компонентом понятия системы программирования являются отладочные программы.

Программный модуль отладки позволяет выполнить основные задачи, связанные с мониторингом процесса выполнения результирующей прикладной программы. Отладка позволяет последовательно и пошагово выполнять итоговые программы, просматривать значения объявленных переменных, устанавливать контрольные точки, трассировку для того, чтобы идентифицировать места и виды ошибок в разработке.

Справочная система, входящая в состав системы программирования, предназначена для предоставления пользователю справочной информации по конкретной системе программирования.

Машинно–ориентированные системы — это системы, в которых язык программирования, наборы операторов и изобразительные средства существенно зависят от особенностей архитектуры компьютера.

Машинно-независимые системы программирования — системы, позволяющие описывать алгоритмы решения задач и информацию, подлежащую обработке. Системы часто используются в широких кругах пользователей и не требуют особых знаний организации функционирования ЭВМ.

Виды языков программирования в машинно-независимых системах:

- процедурно-ориентированные;
- проблемно-ориентированные языки;
- объектно-ориентированное программирование.

Процедурно-ориентированные являются основными языками описания алгоритмов, которые обеспечивают математические функции многих современных вычислительных машин.

Они включают в себя такие популярные языки как:

1. **Fortran —** один из старейших языков программирования высокого уровня, который используется для приложений с интенсивными вычислениями. Fortan часто применяется в процессе научного и инженерного вычисления. Он удобен благодаря большой программной базе, возможностью работы с документами и библиотекам с открытым исходным кодом, доступных под свободными лицензиями. Язык может осуществлять интуитивную запись в виде массива, которая упрощает запись быстрых векторизованных вычислений.
2. **Бейсик** является одним из самых простых языков программирования. Он был создан с целью обучения студентов основам решения задач с помощью написания кода, поэтому программа ориентировалась на пользователей, для которых скорость выполнения программ была не очень важна, и которым первостепенно необходима возможность использовать компьютер для решения своих задач, не имея специальной подготовки. В России сегодня наиболее популярна разновидность **Turbo-Basic** фирмы Borland.
3. Язык программирования **Си** был создан как язык высокого уровня для разработки операционной системы **UNIX** и стал популярен благодаря своей простоте и эффективности. Си существенно повлиял на развитие индустрии программного обеспечения; его синтаксис стал основой для современных и востребованных языков **C++, C#, Java.**
4. **Паскаль** — язык высокого уровня общего назначения, который был первоначально разработан Никлаусом Виртом в начале 1970-х годов. Он разрабатывался с целью создания платформы для **обучения** программированию, поэтому Паскаль недостаточно удобен при решении сложных задач и широко используется для математических операций. Разновидность языка **Паскаль АБС** стала полноценной системой для начинающих, которая сегодня используется для обучения студентов и школьников.

Проблемно-ориентированные языки — это формальные языки, предназначенные для описания данных (информации) и алгоритмов их обработки (программ) на вычислительной машине.

Основные проблемно-ориентированные языки:

1. **ЛИСП** — семейство языков программирования, программы и данные в которых представляются системами линейных списков символов. Так как исходный код состоит из списков, программы на ЛИСПе позволяют его изменять как структуру данных и создавать макросистемы, позволяющие программистам формировать новый синтаксис или новые предметно-ориентированные языки, встроенные в ЛИСП. В настоящее время ЛИСП применяется в экспертных системах, системах аналитических вычислений и т.д.
2. **Prolog** — язык логического программирования, который обеспечивает решение задач, выраженных в терминах объектов и отношений между ними. Для того чтобы инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь».

Объектно-ориентированное программирование основано на методологии представления программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Примеры объектно-ориентированных языков:

1. **JavaScript** — язык сценариев, который позволяет создавать интерактивные html-документы, производить вычисления, выполнять проверку допустимости данных без обращения к серверу. Скрипты программы позволяют взаимодействовать с сайтами: заполнять формы обратной связи, оставлять комментарии, просматривать всплывающие подсказки и т.д.
2. **Objective-C** — один из языков программирования, который активно используется для разработки мобильных приложений. Он используется корпорацией Apple и необходим для операционных систем OS X и iOS, их программных интерфейсов.
3. **Python** — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Он универсален, поэтому подходит для решения разнообразных задач и многих платформ, начиная с iOS и Android и заканчивая серверными ОС. Python используется в различных сферах IT, таких как машинное обучение, разработка приложений, web, парсинг и другие.
4. **Perl** — это язык программирования общего назначения, который изначально был разработан для работы с текстовой информацией, но в дальнейшем стал способен решать широкий круг задач, например, системное

администрирование, веб-разработка, сетевое программирование, разработка графического интерфейса пользователя и т.д.

Актуальные системы программирования.

1. **Eclipse** — свободная интегрированная среда разработки модульных кроссплатформенных приложений, которая часто используется как платформа для разработки расширений. Eclipse является платформой с особым фундаментом для построения и запуска интегрированных инструментов разработки сквозного программного обеспечения. В силу бесплатности и высокого качества, Eclipse во многих организациях является корпоративным стандартом для разработки приложений.
2. **IntelliJ IDEA** — мощная универсальная среда программирования, поддерживающая язык **Java**. Она позиционирует себя как умная и удобная система программирования для Java (и других языков) с поддержкой всех последних технологий и фреймворков. У этой среды разработки есть мощные аналитические возможности. Система включает в себя набор инструментов для изменения внутренней структуры программы, который позволяет быстро реорганизовывать исходные тексты. Дизайн среды ориентирован на продуктивность работы программистов и позволяет оптимизировать простые рутинные задачи, чтобы дать возможность специалистам сконцентрироваться на достижение функциональных целей.
3. **Delphi** — среда разработки прикладных программ, предназначенных для запуска в ОС Windows, MacOS, а также в мобильных операционных системах — iOS и Android. Delphi отличается простотой и может использоваться в учебных целях. Программы в Delphi пишутся на языке **Object Pascal**, который является преемником и развитием языка **Turbo Pascal**. Программа предназначена, в первую очередь, для разработки приложений в архитектуре клиент-сервер.
4. Система **Borland C++ Builder** – это мощная среда программирования, которая поддерживает принципы визуального объектно-ориентированного программирования для 32-разрядных операционных систем **Microsoft Windows** и позволяет значительно сократить время на разработку приложений.
5. **Symantec Cafe** — первая интегрированная среда визуальной разработки для создания приложений и интернет-страниц. Symantec Cafe позволяет разрабатывать приложения на языке Java, которые могут затем встраиваться в интернет-страницы для повышения их функциональности.

Шестой вопрос: Расчет результатов выполнения алгоритма. Анализ алгоритмов с помощью трассировочных таблиц.

Расчет результатов выполнения алгоритма.

Эффективность. Рассматривая различные алгоритмы решения одной и той же задачи, полезно проанализировать, сколько вычислительных ресурсов они требуют (время работы, память), и выбрать наиболее эффективный.

Под временем работы (running time) алгоритма будем подразумевать число элементарных шагов, которые он выполняет. Положим, что одна строка псевдокода требует не более чем фиксированного числа операций (если только это не словесное описание каких-то сложных действий – типа «отсортировать все точки по ***x***-координате»). Следует также различать *вызов* (*call*) процедуры (на который уходит фиксированное число операций) и её исполнение (*execution*), которое может быть долгим.

Сложность алгоритма – это величина, отражающая порядок величины требуемого ресурса (времени или дополнительной памяти) в зависимости от размерности задачи.

Анализ трудоёмкости алгоритмов.

В качестве критерия оптимальности алгоритма выбирается трудоемкость алгоритма, понимаемая как количество элементарных операций, которые необходимо выполнить для решения задачи с помощью данного алгоритма.
Функцией трудоемкости называется отношение, связывающие входные данные алгоритма с количеством элементарных операций.
Одним из упрощенных видов анализа, используемых на практике, является асимптотический анализ алгоритмов. Целью асимптотического анализа является сравнение затрат времени и других ресурсов различными алгоритмами, предназначенными для решения одной и той же задачи, при больших объемах входных данных.

Используемая в асимптотическом анализе оценка функции трудоёмкости, называемая сложностью алгоритма, позволяет определить, как быстро растет трудоёмкость алгоритма с увеличением объема данных. В асимптотическом анализе алгоритмов используются обозначения, принятые в математическом асимптотическом анализе.

Анализ алгоритмов с помощью трассировочных таблиц.

Для анализа свойств алгоритма и проверки его соответствия решаемой задачи используются трассировочные таблицы.

В линейных алгоритмах присваивание является важнейшей операцией в алгоритмах, работающих с величинами, поговорим о ней более подробно.

Переменная величина получает значение в результате присваивания.

Присваивание производится компьютером при выполнении одной из двух команд из представленной выше системы команд: команды присваивания или команды ввода.

Рассмотрим последовательность выполнения четырех команд присваивания, в которых участвуют две переменные: a и b. В приведенной ниже таблице против каждой команды указываются значения переменных, которые устанавливаются после ее выполнения.

Такая таблица называется трассировочной таблицей, а процесс ее заполнения называется трассировкой алгоритма.

Вычисления по алгоритму

Алгоритм

x:=2

*y:=x*x*

*y:=y*y*

*x:=y*x*

s:=x+y

Шаг алгоритма	Переменные		
	x	*y*	*s*
1	2	-	-
2	2	4	-
3	2	16	-
4	32	16	-
5	32	16	48

Ответ: $s = 48$

Прочерк в таблице означает неопределенное значение переменной. Конечные значения, которые получают переменные **a и b**, соответственно равны **2 и 4**.

Трассировочная таблица иллюстрирует три основных свойства присваивания.

Вот эти свойства:

1) пока переменной не присвоено значение, она остается неопределенной;
2) значение, присвоенное переменной, сохраняется вплоть до выполнения следующего присваивания этой переменной нового значения;
3) новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Трассировочная таблица пошаговое исполнение команд алгоритма с указанием значений переменных, которые устанавливаются после выполнения команд.

Трассировка алгоритма – процесс заполнения трассировочной таблицы

Трассировочная таблица

Для трассировки удобно использовать трассировочную таблицу, в которой по вертикали откладывается шаг программы, а по горизонтали значение переменных и показания экрана

Пример

```
Program chetnost;
  var i, x: integer;
  begin
    for i:=1 to 4 do
      begin
        x:= i mod 2;
        if  x = 0
          then  writeln(i, ' – четное')
          else writeln(i, ' – нечетное');
    end;
  end.
```

Трассировочная таблица

i	x	Экран
1	1	1 – нечетное
2	0	2 – четное
3	1	3 – нечетное
4	0	4 – четное

Трассировочная таблица

Для поиска ошибок удобно визуализаровать переменные на каждом шаге работы программы,
Для этого в тело программы вставляют операторы вывода переменных на экран.

Пример

```
Program sravnenie;
  var x, y: integer;
  begin
    x:=2;
    y:=10;
    while x<y do
      begin
        x:=sqr(x);
        y:=2*y;
      end;
    write(x:4,y:4);
end.
```

Добавим в цикл оператор вывода

```
Program sravnenie;
  var x, y: integer;
  begin
    x:=2;
    y:=10;
    while x<y do
      begin
        x:=sqr(x);
        y:=2*y;
        writeln('x=', x,' y=',y);
      end;
    write(x:4,y:4);
end.
```

Трассировочная таблица

x	y	Экран
2	10	
4	20	x=4 y=20
16	40	x=16 y=40
256	80	x=256 y=80
		256 80

Таким образом мы получим аналог трассировочной таблице на экране.

Трассировочная таблица

Если значение какой-то переменной в текущем шаге программы не изменилось,
то соответствующую клетку таблицы оставляют пустой.

Пример

```
Program uslovie;
  var i, x, y: integer;
  begin
    x:=2;
    y:=11;
    for i:=1 to 3 do
          if x mod 4 = 0
            then
              x:=sqr(x);
            else
              begin
                x:=2*x;
                y:=2*y;
              end;
    write(x,' ',y);
end.
```

Трассировочная таблица

i	x	y	Экран
	2	11	
1	4	22	*i=1 4 22*
2	16		*i=2 16*
3	256		*i=3 256*
			256 22

Для удобства трассировки мы добавили вспомогательные операторы вывода,
Но из-за этого нам пришлось добавить и дополнительные begin и end,
Так как в теле операторов for и if стало больше одной команды.

```
Program uslovie;
  var i, x, y: integer;
  begin
    x:=2;
    y:=11;
    for i:=1 to 3 do
      begin
        write('i=',i,' ');
          if x mod 4 = 0 then
            begin
              x:=sqr(x);
              writeln( x, ' ');
            end
          else
            begin
              x:=2*x;
              y:=2*y;
              writeln( x, ' ', y);
            end;
      end;
    write(x,' ',y);
end.
```

В таблице действие дополнительных операторов вывода показано красным курсивом.

Заключительная часть (по каждому занятию).

1. Закончить изложение материала.
2. Ответить на возникшие вопросы.
3. Подвести итоги занятия.
4. Дать задание на самоподготовку (домашние задания).

Задание на самоподготовку (домашние задания):

1. Детально проработать, законспектировать материал занятия, размещенный в данном план-конспекте, в учебниках, указанных на с.2 текущего документа.
2. Подготовиться к опросу по пройденному материалу, защите ранее выполненных практических работ.