**1*.*Дайте определения понятиям: Алгоритм, Алгоритмизация. Пример алгоритма. Свойства алгоритма.**

***Алгоpитм***– это точное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

**Алгоритмизация** – это техника разработки алгоритма для решения задач на ЭВМ.

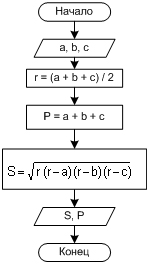
1. ***Массовость.*** Алгоритм служит для решения целого класса задач.
2. ***Понятность.*** Чтобы алгоритм можно было выполнить, он должен быть понятен исполнителю.
3. ***Дискретность.*** Алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов.
4. ***Конечность.*** Выполнение алгоритма заканчивается после выполнения ***конечного числа шагов***.
5. ***Определенность.*** Каждый шаг алгоритма должен быть ***четко и недвусмысленно определен*** и не должен допускать произвольной трактовки исполнителем.
6. ***Эффективность.*** Каждый шаг алгоритма должен быть выполнен точно и за конечное время.

**ПРИМЕР: Зная длины трех сторон треугольника, вычислить площадь и периметр треугольника.**

 Пусть a, b, c - длины сторон треугольника. Необходимо найти S - площадь треугольника. Для нахождения площади можно воспользоваться формулой Герона:

[https://sites.google.com/site/informvecole/_/rsrc/1467890432493/dla-ucenikov/9-klass/osnovy-algoritmizacii/5.jpg](https://sites.google.com/site/informvecole/dla-ucenikov/9-klass/osnovy-algoritmizacii/5.jpg?attredirects=0), где r - полупериметр, P - периметр.

 Входные данные: a, b, c. Выходные данные: S, P.

 ***Знак "=" означает не математическое равенство, а операцию присваивания. Переменной, стоящей слева от оператора, присваивается значение, указанное справа.***

**2. Перечислить основные шаги для описания алгоритма. Привести пример алгоритма.**

***Алгоpитм***– это точное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

**Основные шаги для описания алгоритма:**

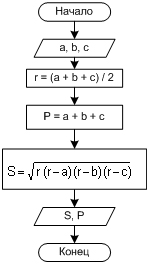
**1**.Изучение задания для алгоритмизации. Оно представляется в описательной форме с использованием формул, таблиц, графиков. Перед разработчиком возникает необходимость глубоко изучить алгоритмируемый процесс, то есть выяснить все закономерности, факторы, оценить степень влияния отдельных явлений на конечные результаты. Необходимо определить вход/выход информации, точность вычислений.

**2**. Входная информация должна быть полной для обеспечения получения всей выходной информации. Входная информация бывает 2-х видов: постоянная и переменная. Универсальность алгоритма зависит от соотношения объемов постоянной и переменной информации.

**3**. Далее выполняется математическая формулировка поставленной задачи: с целью построить массивы арифметических {А} и логических {Р} операторов. Данные массивы являются материалом из которого строится схема алгоритма.

**ПРИМЕР: Зная длины трех сторон треугольника, вычислить площадь и периметр треугольника.**

 Пусть a, b, c - длины сторон треугольника. Необходимо найти S - площадь треугольника. Для нахождения площади можно воспользоваться формулой Герона:

[https://sites.google.com/site/informvecole/_/rsrc/1467890432493/dla-ucenikov/9-klass/osnovy-algoritmizacii/5.jpg](https://sites.google.com/site/informvecole/dla-ucenikov/9-klass/osnovy-algoritmizacii/5.jpg?attredirects=0), где r - полупериметр, P - периметр.

 Входные данные: a, b, c. Выходные данные: S, P.

***Знак "=" означает не математическое равенство, а операцию присваивания. Переменной, стоящей слева от оператора, присваивается значение, указанное справа.***

**3.Средства описания алгоритма: Операторная форма. Привести пример.**

Сложность операторов определяется характером решаемой задачи, чтобы после объединения операторов получилось описание алгоритма необходимо описать взаимосвязь операторов. Обозначение операторов: А-арифметические, Р-логические.

Основные правила операторной формы:

1.Порядковый номер оператора указывается нижним индексом, нумерация-сквозная(,,).

2.Если оператор зависит от параметра, то этот параметр отображается в верхнем индексе.

3.Если знаки двух операторов располагаются в схеме рядом, то оператор слева передает управление оператору, стоящему справа.

4.Если между двумя операторами стоит точка с запятой, тогда оператор слева не передает управление оператору справа.

5.Передача управления оператору, стоящему не рядом справа, обозначается стрелкой.

Использование операторной формы рассчитано на получение следующих преимуществ:

1. Облегчается работа по составлению программы.

2. Уменьшается количество ошибок при программировании.

3. Облегчает проверку готовых программ.

4. Появляется возможность возобновить работу над программой после длительного перерыва или поручить её другому лицу.

Пример:   


Где М-это количество указанных чисел меньших К  


1. М:=  0; Б:=  0
2. I:=1
3. Если  , тогда переходим к указателю 6
4. Б:= Б+1
5. Переход к указателю 7
6. М:=М+1
7. Если I⩾  n, тогда переход к указателю 9
8. I:=I+1
9. Переход к указателю 3

10.Конец

Операторная форма:

**4.Средства описания алгоритмов: схема алгоритма. Привести пример.**

Схема алгоритма служит как формальный язык для записи содержания логических связей в задаче. Она сранительно легко преобразуется в программу на ЭВМ.

Схемы алгоритмов- это функционально-ориентированные графические изображения, с помощью которых описывают последовательность шагов алгоритмизации.

Преимущества схем алгоритмов:

1.Не требуется специальных знаний для понимания

2. Для удобства чтения схем алгоритмов в ряде стран разработаны стандартные графические изображения операторов.

Пример:   


Где М-это количество указанных чисел меньших К  


1. М:=  0 ; Б:=  0
2. I:=1
3. Если  , тогда переходим к указателю 6
4. Б:= Б+1
5. Переход к указателю 7
6. М:=М+1
7. Если I⩾  n, тогда переход к указателю 9
8. I:=I+1
9. Переход к указателю 3
10. Конец

Схема алгоритма:

**5.Средства описания алгоритмов: граф- схема алгоритма. Привести пример.**

Граф-схема алгоритма-это конечный связный ориентированный граф, вершины которого соответствуют операторам, а дуги задают порядок следования вершин(операторов) алгоритма.

Основные свойства граф-схемы:

1.Состоит из конечного числа точек, называемых вершинами, которые соединяются стрелками.

2.В граф-схеме есть 2 особых узла: 1-входной, в который не входит ни одна стрелка,2- выходной, из которого не выходит ни одна стрелка

3. Все вершины граф-схемы отмечены своим оператором (А или Р)

4. Из каждого узла с символом А выходит только одна стрелка, из символа Р как минимум 2. Если в граф-схеме вместо операторов поставить их описания, то она превратиться в схему алгоритма

Пример:   


Где М-это количество указанных чисел меньших К  


1. М:=  0 ; Б:=  0
2. I:=1
3. Если  , тогда переходим к указателю 6
4. Б:= Б+1
5. Переход к указателю 7
6. М:=М+1
7. Если I⩾  n, тогда переход к указателю 9
8. I:=I+1
9. Переход к указателю 3
10. Конец

Граф-схема:

1. **Численные методы реализации алгоритмов.**

**Численный метод** – метод приближенного или точного решения математических задач ,основанный на построении конечной последовательности действия над конечным множеством чисел **Выбор численных методов:** 1) Исследовались вопросы получения погрешностей из-за непрерывных методов вычислений дискретными операциями 2) Начали изучать погрешности представления данных и накопления систематических ошибок при большом числе простейших операций над приближёнными числами Выбирая численный метод необходимо проводить проверку правильности вычислений при решении задач на машине . Способ контроля при помощи повторных расчетов не всегда эффективен. Более экономным и эффективным может оказаться контроль путём проверок каких-либо заранее известных соотношений между вычисленными величинами.

**7.Описание абстрактной машины Поста, система команд, пример программы.**

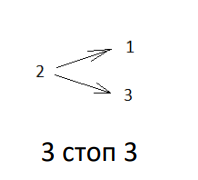
**Маши́на По́ста** — абстрактная машина Поста состоит из бесконечной ленты, разделенной на разные секции, а также считывающую головку. В каждой секции м.б. ничего не записано либо записана метка, такая секция называется отличной. Информация о заполнении метками секции ленты характеризует состояние ленты.

Всего для машины Поста **существует шесть типов команд:**

1. xMy — поставить метку;
2. xCy — стереть метку;
3. x← y — сдвинуться влево;
4. x→ y — сдвинуться вправо;
5. х → y1|y2 — условная передача управления (если нет метки y1, если есть y2);
6. xCТОПx — конец программы.

Пример: Переход к первой метке:

1 → 2



**8. Дайте определения конечного автомата и автономного автомата.**

**Конечный автомат** — это некоторая абстрактная модель, содержащая конечное число состояний чего-либо. Используется для представления и управления потоком выполнения каких-либо команд.

Для описания конечных динамических систем используется 2 множества:

**{C} = {C1, C2, …, Ck}** – состояние дин. системы

**{P} = {P1,P2,…,Pr}** – входы дин. cистемы

Для полного описаниянеобходимо определить закон изменения состояние системы в каждый этап времени. Состояние системы в любом такте **i** обозначается **Ci-1** и входом в текущий такт (**Pi**)

**Ci=F(Ci-1,Pi)**

Т.е. задать конечный автомат – определить множество С, P и функцию F. Если переменную P зафиксировать, т.е. **Pc={P1}**(**Ci=F(Ci-1,PC)**, тогда такой автомат называется автономным.

**Автономный автомат** – автомат, функция переходов и функция выходов которого не зависят от входа.

**9. Дайте определение конечного автомата с выходным преобразователем.**

**Конечный автомат:** система, состоящая из конечного автомата, преобразующего символы алфавита **Р** в символы алфавита **С** в соответственной функции **Сi= F(Ci-1; Pi)** и преобразователя  λ, которая однозначно ставит символ в соответствии из алфавита **А. A=λ(Ci, Pi)** A1…A3 – случайные.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **P1** | **P2** | **P3** |
| **с** | **A1** | **A2** | **A3** |
| **C1** | **A1** | **A3** | **A1** |
| **C1** | **A3** | **A2** | **A2** |

**10. Способ задания конечного автомата с помощью основной таблицы.**

В таблице по вертикали записываются все состояния системы, а по горизонтали все символы.

Чтобы определить текущее состояние находим предыдущее состояние по вертикали и по горизонтали.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **P1** | **Pn** | … | **Pe** | … | **Pr** |
| **С1** | **C3** | **C8** |  |  |  |  |
| **С2** | **C1** | **Cn** |  |  |  |  |
| **…** |  |  |  |  |  |  |
| **СJ** |  |  |  | **CJe** |  |  |
| **…** |  |  |  |  |  |  |
| **Сk** |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | **P1** | **P2** |
| **С1** | **С2** | **С3** |
| **С2** | **С1** | **С2** |
| **С3** | **С4** | **С3** |
| **С4** | **С2** | **С1** |

Пример:

**Ci-1 = C1**

**P = P1**

**Ci = C3**

Автономный автомат представляет собой один столбец из основной таблицы

|  |  |
| --- | --- |
|  | **Pс** |
| **С1** | **С2** |
| **С2** | **С1** |
| **С3** | **С4** |
| **С4** | **С2** |

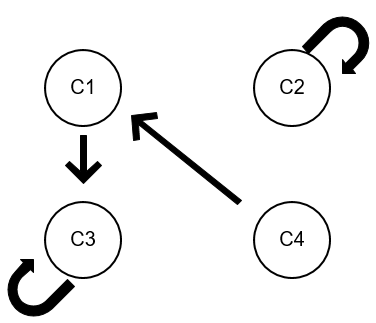
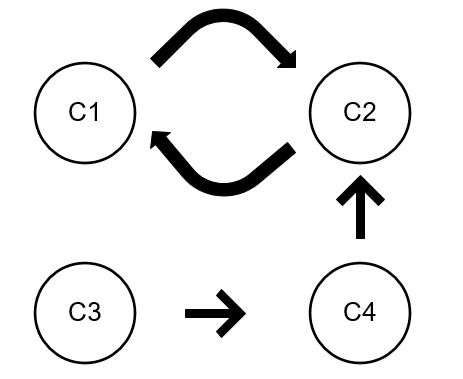
**11. Способ задания конечного автомата с помощью граф-схемы.**

**Граф автомата** - это ориентированный связный граф, вершины которого соответствуют состояниям, а ребра графа – переходам между ними. Две вершины графа автомата соединяются ребром, направление указывается стрелкой. Ребру графа приписывают соответствующие значения входных и выходных сигналов, если они определены. Если переход из состояния aiв состояния aJ происходит под воздействием нескольких входных сигналов, то соответствующему ребру (ai, aJ) присваивают все значения входных и выходных сигналов.

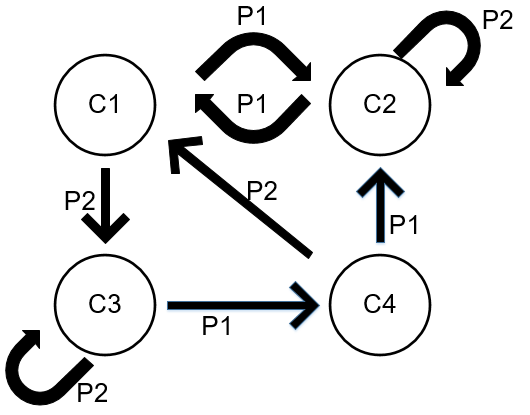
Вершина графа – состояние; Дуга - переход

A(P1) A(P2)

|  |  |  |
| --- | --- | --- |
|  | **P1** | **P2** |
| **С1** | **С2** | **С3** |
| **С2** | **С1** | **С2** |
| **С3** | **С4** | **С3** |
| **С4** | **С2** | **С1** |



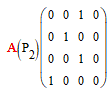
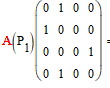
Объединенные графы автомата



**12.Способ задания конечного автомата с помощью с помощью матриц.**

Каждому графу автономного автомата можно однозначно сопоставить квдратную матрицу состоящую из **0** и **1**. **1**- есть связь между вершинами

**0**-нет связи между вершинами.



**13.Метасинтаксический язык Бэкуса.**

Метаязык, предложенный Бэкусом и Науром, впервые использовался для описания синтаксиса реального языка программирования Алгол 60. Наряду с новыми обозначениями метасимволов, в нем использовались содержательные обозначения нетерминалов. Это сделало описание языка нагляднее и позволило в дальнейшем широко использовать данную нотацию для описания реальных языков программирования. Были использованы следующие обозначения:

* символ " ::= " отделяет левую часть правила от правой;
* нетерминалы обозначаются произвольной символьной строкой, заключенной в угловые скобки "<" и ">";
* терминалы - это символы, используемые в описываемом языке;
* каждое правило определяет порождение нескольких альтернативных цепочек, отделяемых друг от друга символом вертикальной черты "|".

Пример: <цифра> :: = 0|1|2|3|4|5|6|7|8|9; Формы Бекуса-Наура используются с двумя основными целями:

* они являются металингвистическим языком, стандартным для описания языков программирования;
* они описывают правила построения текстов или конструкций.

Таким образом, множество терминальных символов, множество нетерминалов с выделенными начальными символами и множество правил вывода входят в формальное определение порождающей грамматики. Язык, порождаемый такими грамматиками, это множество терминальных цепочек, которые можно вывести из начального символа.

**14.нотации бэкуса-наура** **Расширенная форма Бэкуса-Наура** — формальная система определения синтаксиса, в которой одни синтаксические категории последовательно определяются через другие.

[ ] – включённая них конструкция может отсутствовать

{ } – заключённая в них конструкция может повторятся ,возможно 0 раз

{/ /} - обозначает повторения 1 и более раз

( ) - используется для отображения альтернативных конструкций Нетерминальные символы, состоящие из нескольких слов, пишутся слитно, например, чётное\_число, ЧётноеЧисло. Терминальные символы изображаются словами, заключёнными в кавычки. Синтаксические правила начинаются с знака “$”, заканчиваются символом “.”. Левая часть отделяется от правой знаком “=”. $буквы=”A”|”B”|…|”a”|”b”|…|”z”. $цифры=”0”|”1”|”2”|…|”9”. $идентификатор=буква{буква/цифра}.

**15. диаграммы Вирта**

Это графический мета язык для описания языка программирования. Используются обозначения: — терминальный символ принадлежащий алфавиту языка.  — постоянная группа терминальных символов определяющая название лексем, ключевых слов и тд.  — нетерминальный символ, определяющий название правила. — соединительные линии, обеспечивающие связь между терминальными и нетерминальными символами.  — входная дуга с именем правила определяющая ее начало.

**16.подходы к построению алгоритмических языков: основные этапы.**

Алгоритмический язык является средством прочного формулирования вычислительных процессов для их последующей реализации на вычислительных машинах.

3 основных задачи

1)алгоритм численного анализа(расчеты по формулам и т.п.)

2)процесс обработки данных различных типов и структур

3)переработка символьной информации(обработка текстов , моделирование интеллекта)

**17.свойства хорошего языка программирования.**

1) Ясность простота и единообразие понятий. ЯП обеспечивает как систему понятий для обдумывания алгоритмов, так и средства выражения этих алгоритмов. 2) Естественность для приложений. Язык должен иметь такой синтаксис, который при правильном использовании отражает в коде реализуемые алгоритмические структуры. 3) Ортогональность. Любые возможные комбинации различных языковых конструкций будут осмысленными. 4) Поддержка абстракций. Возможной частью работы программиста является разработка конкретных абстракций для решения задачи и их реализация. 5) Удобство верификации программы. Методы верификации программы позволяют без выполнения проверить на наличие ошибок. 6) Переносимость программ. Кроссплатформенные языки, Языки под различные аппаратные архитектуры (виртуальные машины). 7) Стоимость использования: • стоимость выполнения программы(ресурсы), • стоимость трансляции программы (на стадии разработки), • стоимость создания тестирования и использование ПО, • стоимость сопровождения ПО.

**18.синтаксические элементы языка**

1) Набор символов. Определение набора символов — одна из первоочередных задач при разработке синтаксиса языка. 2) Идентификаторы. Различия правил составления идентификаторов в языках сводятся к включению в допустимый набор букв и цифр специальных символов. • run\_motion () — змеиная нотация, • runMotion () — верблюжья нотация. 3) Символы операций. + - \* 4) Ключевые и зарезервированные слова. Ключевое слово – идентификатор, используемый в качестве фиксированной части синтаксиса какой-либо оператора. 5) Комментарии. Являются важной частью документирования программы. //, #, /\* \*/, ‘’’ ‘’’. 6) Разделители и скобки. Разделитель — синтаксический элемент, функция которого обозначить начало и конец некоторой синтаксической конструкции. (), {}. 7) Операторы. Важные синтаксические компоненты, от них зависти удобство записи и чтение программы. 8) Выражения. Это функции, которые обрабатывают какиелибо данные в программе и возвращают некоторые значения.

**19.организация выполнения программ**

1) Трансляция(компиляция). Создаётся транслятор, который будет переводить программу на язык высокого уровня в эквивалентные программы на машинном языке. Типы трансляторов: • загрузчик или редактор связей, •процессор или макропроцессор, •компилятор. 2) Программная имитация (интерпретация). Моделируется на хост-компьютере другой компьютер для которого машинным языком будет ЯВУ. •Медленнее. Можно ускорить (многопоточность, CHDA) •Чаще используется комбинированный подход 1 и 2. Подключаются на этапе загрузки run-time библиотеки для создания выполняемой формы программы. Виртуальные машины.

**20.использование формальных языков для поиска.**

*Информационный поиск:* задачи обработки информации, библиографического поиска и анализа фактографических данных часто объединяют под общим названием информационно-логических задач. Для них характерны логические обработки больших объемов информации и ее хранение. Дескрипторный метод поиска: Содержание документа в общих чертах может быть представлена набором характерных для данного текста слов. Их называют ключевыми словами. Для построения поисковой системы из всего многообразия ключевых слов, выбранных из ряда характерных для данной тематики текстов составляется стандартный набор терминов без синонимов со строго фиксированными значениями. Эти термины называются **дескрипторами**, а полный набор таких терминров – **словарём дискрипторов.** Для каждого документа составляется свій нобор дискрипторов, называемый **поисковым образом документа.** Масив дискрипторных наборов документа может быть построен двумя разными способами: **прямым и инвертным**. При прямом способе в память машины последовательно записываются номера документов и за каждым из них указываются дескрипторы или их коды. Процесс поиска заключается в последовательном сравнении для каждого документа всех дескрипторов запроса с дескрипторами документа и выделение тех документов, для которых выполняется критерий соответствия. При инвертном способе за основные позиции принимают не документы, а дескрипторы. Для каждого дескриптора записываются все номера документов, которые имеют в своих поисковых образах этот дескриптор. Поиск нужных документов осуществляется в словаре дескрипторов путём обращения к тем дескрипторам, которые имеются в запросе, и путём выбора всех номеров документов этих дескрипторов. Отобранными считаются те документы, номера которых оказались общими для всех дескрипторов указанных в запросе.

**21.Дайте определение понятиям: Базы данных и Банки данных.**

Идея создания БД применяется, когда для решения задач исходные данные и промежуточные результаты хранятся отдельно от исполняемых программ.

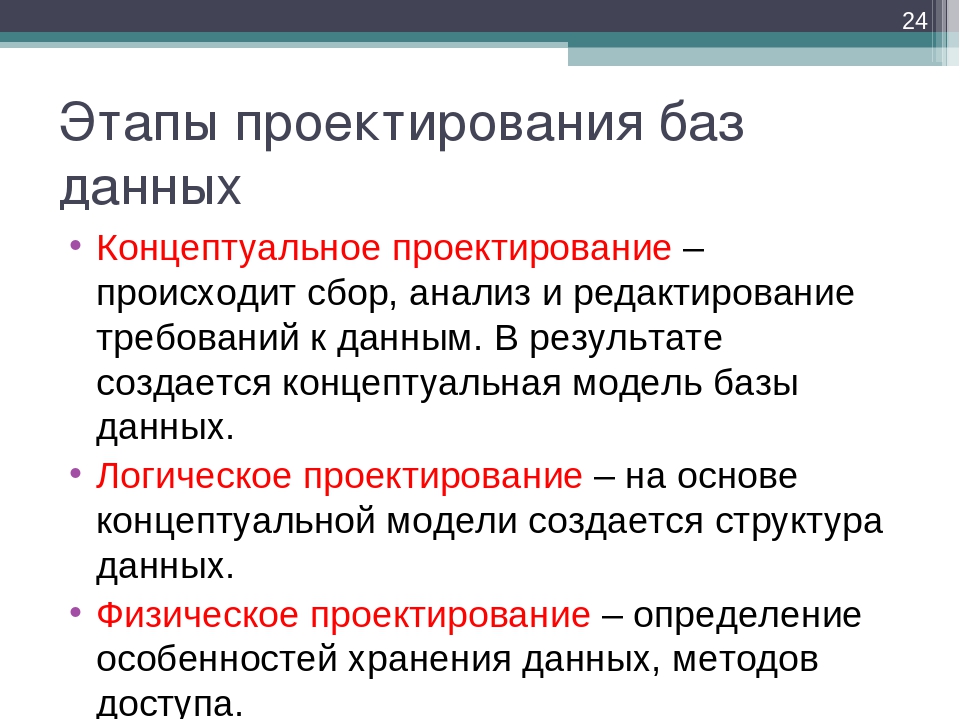
**БД** — совокупность взаимосвязанной информации, организованной по определенным правилам.

**Банк данных (БнД)** – это система специально организованных данных, предназначенных для централизованного накопления и коллективного многоцелевого использования.

**22. основные требования к базам данных**

1) целостность базы данных, т.е. требование полноты и непротиворечивости данных. 2) многократное использование данных. 3) уменьшение избыточности данных. 4) быстрый поиск и получение информации по запросам пользователя. 5) простота обновления данных. 6) защита данных от несанкционированного доступа, искажение и уничтожение.

**23.этапы проектирования БД: концептуальная модель.**



**Концептуа́льная** **моде́ль** — это **модель**, представленная множеством понятий и связей между ними, определяющих смысловую структуру рассматриваемой предметной области или её конкретного объекта.

**24. Модель «Сущность-связь»(ЕR-диаграмма).**

Средством моделирования предметной области на этапе концептуального проектирования является модель "сущность–связь". Часто ее называют ERмоделью (Entity – сущность, Relation – связь). В ней моделирование структуры данных предметной области базируется на использовании графических средств – ER-диаграмм (диаграмм "сущность–связь"). В наглядном виде они представляют связи между сущностями. Основные понятия ER-диаграммы – сущность, атрибут, связь. Сущность - ϶ᴛᴏ некоторый объект реального мира, который может существовать независимо. Сущность имеет экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию. Атрибут - ϶ᴛᴏ свойство сущности. К примеру, сущность КНИГА характеризуется такими атрибутами, как автор, наименование, цена, издательство, тираж, количество страниц. Конкретные книги являются экземплярами сущности КНИГА

На ER-диаграмме сущность изображается прямоугольником, в котором указывается ее имя



На ER-диаграмме связь изображается ромбом.

