**Двумерные массивы - матрицы. Метапредметный подход**

**Метапредметность на уроках информатики.**

Установленные стандартом новые требования к результатам обучающихся вызывают необходимость в изменении содержания обучения на основе принципов метапредметности как условия достижения высокого качества образования. Учитель сегодня должен стать конструктом новых педагогических ситуаций, новых заданий, направленных на использование обобщенных способов деятельности и создание учащимися собственных продуктов в освоении знаний.

"Мета "–(«за», «через», «над»), всеобщее, интегрирующее: метадеятельность, метапредмет, метазнание, метаумение (метаспособ). Иногда это называют универсальными знаниями и способами. Иногда - мыследеятельностью.

**Метадеятельность** - универсальная деятельность, которая является "надпредметной". Предметная - это любая деятельность с предметом. В любой предметной деятельности есть то, что делает ее осознанной и ответственной, то есть: стратегической; исследовательской; проектировочной; сценирующей; моделирующей; конструирующей; прогнозирующей.

Метадеятельность как универсальный способ жизнедеятельности каждого человека определяется уровнем владения им метазнаниями и метаспособами, т.е. уровнем развития личности.

**Метазнания** - знания о знании, о том, как оно устроено и структурировано; знания о получении знаний, т.е. приёмы и методы познания (когнитивные умения) и о возможностях работы с ним (смотри философия, методология, многоотраслевая метанаука). Понятие «метазнания» указывает на знания, касающиеся способов использования знаний, и знания, касающиеся свойств знаний. Метазнания, выступают как целостная картина мира с научной точки зрения, лежат в основе развития человека, превращая его из «знающего» в «думающего».

Метазнания включают в себя философию предмета и общую философию. Философия предмета включает в себя понятие, границы и методологию предмета как части науки. Философия физики, например, анализирует, проблему несовпадения онтологической и физической проекций: понимание физикой времени как течения наиболее стабильного процесса и онтологическое понимание времени как течения времени вообще или смены фаз: прошлое, настоящее, будущее.

Философские проблемы географии заключаются в рамках ключевых аспектов взаимодействия общества и природы, проблем экологии, строящейся на основе принципов философии природы, признающей целостность материального и духовного Мира. К числу общих проблем, относятся также: эволюция Земли и жизни на ней, пространственное разнообразие природных условий, влияние природы на человека и общества на природу. Все эти проблемы - предмет как философии, так и географии. Проблема сохранения жизни на Земле становится краеугольным камнем формирования географической культуры.

Философия искусства исследует сущность и смысл искусства на основе искусства в целом и на основе частных видов искусства, учитывая при этом содержание, смысл и его функции внутри культуры и всей сферы ценностей.

Философия музыки – это понимание ее сущности и особенности, история зарождения и развития, современное состояние и значение в обществе и духовной жизни человека, как творящего музыку, так и воспринимающего ее.

**Метаспособы** - методы, с помощью которых человек открывает новые способы решения задач, строит нестереотипные планы и программы, позволяющие отыскать содержательные способы решения задач. (Ю. Н. Кулюткин)

**Метаумения** – присвоенные метаспособы, общеучебные, междисциплинарные (надпредметные) познавательные умения и навыки. К ним относятся:

- теоретическое мышление (обобщение, систематизация, определение понятий, классификация, доказательство и т.п.);

- навыки переработки информации (анализ, синтез, интерпретация, экстраполяция, оценка, аргументация, умение сворачивать информацию);

- критическое мышление (умения отличать факты от мнений, определять соответствие заявления фактам, достоверность источника, видеть двусмысленность утверждения, невысказанные позиции, предвзятость, логические несоответствия и т.п.);

- творческое мышление (перенос, видение новой функции, видение проблемы в стандартной ситуации, видение структуры объекта, альтернативное решение, комбинирование известных способов деятельности с новыми);

- регулятивные умения (задавание вопросов, формулирование гипотез, определение целей, планирование, выбор тактики, контроль, анализ, коррекция свей деятельности);

-качества мышления (гибкость, антиконфоризм, диалектичность, способность к широкому переносу и т.п.).

В настоящее время формирование метаумений становится центральной задачей любого обучения.

Метапредметный подход обеспечивает переход от существующей практики дробления знаний на предметы к целостному образному восприятию мира, к метадеятельности. Метапредметность как принцип интеграции содержания образования, как способ формирования теоретического мышления и универсальных способов деятельности обеспечивает формирования целостной картины мира в сознании ребёнка. При таком подходе у учащихся формируется подход к изучаемому предмету как к системе знаний о мире, выраженном в числах и фигурах (математика), в веществах (химия), телах и полях (физика), художественных образах (литература, музыка, изобразительное искусство) и т.д.

Возможности формирования метадеятельности заложены в ряде методик, подходов и технологий: развивающее обучение Эльконина-Давыдова; мыследеятельностная педагогика; коммуникативная дидактика; эвристическое обучение; логико-смыслового моделирования; школа М.Щетинина и др.

Метапредметный подход обеспечивает целостность общекультурного личностного и познавательного развития и саморазвития ребенка, преемственность всех ступеней образовательного процесса, лежит в основе организации и регуляции любой деятельности ученика независимо от ее специально-предметного содержания.

Метапредмет — учебный предмет нового типа, в основе которого лежит мыследеятельностный тип интеграции учебного материала. Метапредметы — это новая образовательная форма, которая выстраивается поверх традиционных учебных предметов, это учебный предмет нового типа, в основе которого лежит мыследеятельностный тип интеграции учебного материала, каковыми являются метазнание, метаспособы, метадеятельность.

Механизмом развития метадеятельности может стать система инновационных творческих проектов. При их создании у учеников формируются понятия, факты, идеи, законы, общие для всех наук, развивается способы, действия, которые они приобретают в процессе обучения, появляется привычка мыслить и действовать в соответствии с принципами метапредметности, то есть происходит интеграция знаний, приобретается опыт творческой деятельности.

Принцип «метапредметности» состоит также в обучении школьников общим приемам, техникам, схемам, образцам мыслительной работы, которые лежат над предметами, поверх предметов, но которые воспроизводятся при работе с любым предметным материалом (Ю.Громыко). Это составление ментальных карт, деревьев понятий, кластеров, денотатные графы, схем «фишбоун» (рыбьи косточки – технология «за и против»), различные техники графические модели знания, приемы сворачивания информации (конспект, таблица, схема) и пр.

Метапредметные образовательные результаты предполагают, что у учеников будут развиты: уверенная ориентация в различных предметных областях за счет осознанного использования при изучении школьных дисциплин философских и общепредметных; владение основными общеучебными умениями информационно-логического характера, умениями организации собственной учебной деятельности, основными универсальными умениями информационного характера, информационным моделированием как основным методом приобретения знаний, широким спектром умений и навыков использования средств информационных и коммуникационных технологий для сбора, хранения, преобразования и передачи различных видов информации, базовыми навыками исследовательской деятельности, проведения виртуальных экспериментов, способами и методами освоения новых инструментальных средств, основами продуктивного взаимодействия и сотрудничества со сверстниками и взрослыми.

Информатика, как учебная дисциплина, прочно завоевала место в базовом образовании. Поэтому целью обучения информатике является не только и ни сколько написание «программных кодов», сколько привитие методологических и технологических подходов и навыков, воспитание  соответствующего способа думать, ставить и решать задачу. Такой подход к обучению позволяет сформировать думающего исследователя. Программный инструментарий в преподавании курса информатики вместе с многообразием форм учебного процесса призваны обеспечить исследовательскую и творческую его направленность.

Обучение информатике – это искусство, направленное не на весь класс одновременно, а на каждого ученика в отдельности. Так вот получается, что урок – как музыкальное произведение, рождается и умирает со звонком, но в душе и в уме каждого ученика оставляет свой собственный след, рождает тропинку, по которой еще предстоит пробираться через дебри и овраги к пониманию, к истине, а значит, и к радости.

Ни один навык не формируется без устойчивого интереса. Познавательный интерес является одним из значимых факторов активизации учебной деятельности. Только в этом случае учение становится личностно – значимой деятельностью, в которой сам обучающийся заинтересован.

Принятие обучающимся роли активного субъекта учебного процесса является условием формирования познавательных интересов и потребностей, более высокого развития личности. Благодаря интересу создается эмоционально – положительная атмосфера  обучения, более интенсивно протекает познавательная деятельность. А применение компьютера в качестве инструмента, обладающего огромным потенциалом возможностей и подразумевая его интенсивное  использование, это более живой и интересный путь, дающий новые возможности для творчества.

На уроках  информатики использование компьютеров позволяет учащимся заниматься исследовательской работой при решении задач из различных областей (например, физические, математические, экономические задачи). При этом они должны научиться четко формулировать задачу, решать ее и оценивать результат.

Еще на заре становления европейской науки Г. В. Лейбницем была сформулирована мысль, что познание сути вещей равносильно раскрытию их внутренней формы. Информатика, по сути, и есть дисциплина, занимающаяся построением и изучением этих «форм».  Поэтому, универсальность, «метапредметность» информатики не является чем-то насильственным или внешним, а вытекает из самого существа науки.

Системно-информационная концепция определяет интегрирующую роль информатики среди всех школьных дисциплин. За счет организации межпредметных связей, реализуемых в процессе решения на уроках информатики разноплановых задач, появляется возможность закреплять и углублять знания, полученные на других предметах. При этом акцент делается на развитие мышления, которое определяет способность человека оперативно обрабатывать информацию и принимать обоснованные решения. Развитием мышления занимаются практически, во всех школьных предметах, но на базе системного подхода нигде, только на информатике. Информатика, позволяющая аккумулировать знания из разных предметных областей, это именно та дисциплина, где реально можно воплотить идею развития системного мышления, у каждого учащегося. Принцип межпредметных связей может быть реализован на практических занятиях. Это достигается в процессе решения многочисленных задач из разных предметных областей.

В качестве примера можно остановиться на следующих темах, отражающих межпредметные связи между курсом информатики и математическими курсами:

Целые и рациональные алгебраические выражения – предлагается составить комплекс программ, реализующих в нем операции сложения, вычитания и деление с остатком.

Делимость чисел – решение задачи, связанной с теоремой Лагранжа (математические формулы и теоремы используются для анализа алгоритма).

Решение алгебраических уравнений с рациональными коэффициентами – задача демонстрирует связь представления многочлена как алгебраической структуры и функциональной зависимости, а также практическое применение этой связи.

Комбинаторика. Одним из важнейших применений комбинаторики является программирование, где перестановки и их свойства существенно используются для анализа различных алгоритмов сортировки информации.

Выпуклые фигуры. Рассматриваются методы аналитической геометрии, когда точка задается своими координатами, а линии и поверхности – уравнениями, решениями которых являются соответствующие множества точек. Например, задачи линейного программирования, где появляется необходимость строить выпуклую оболочку множества точек.

Методически продуманный отбор заданий для практики по программированию позволяет наряду с изучением информатики активизировать и углубить знания учащихся по математике. При этом математические понятия и теоремы используются для разработки и доказательства правильности алгоритмов и для их анализа, т.е. приобретают практический навык и носят прикладной характер.

Развитие познавательного интереса учащихся к информатике, программированию – задача чрезвычайной важности, от решения которой в значительной мере зависит успех овладения учащимся второй компьютерной грамотностью. Поэтому одной из важных форм укрепления интереса учащихся к информатике является правильная мотивация в обучении. Мотивационный компонент должен в разнообразной форме присутствовать на протяжении  всего времени обучения информатике при решении различных задач, в том числе и прикладных. Интересно, что с использованием межпредметных связей, можно построить и сами уроки. Знания основ информатики не только способствуют развитию познавательного мышления, но и закладывают основы успешного овладения всем курсом информатики, способствуют развитию алгоритмического мышления, помогают в овладении любыми знаниями.

**Двумерные массивы Паскаля – матрицы**

Двумерный массив в Паскале трактуется как одномерный массив, тип элементов которого также является массивом (массив массивов). Положение элементов в двумерных массивах Паскаля описывается двумя индексами. Их можно представить в виде прямоугольной таблицы или матрицы.

Рассмотрим двумерный массив Паскаля размерностью 3\*3, то есть в ней будет три строки, а в каждой строке по три элемента:



Каждый элемент имеет свой номер, как у одномерных массивов, но сейчас номер уже состоит из двух чисел – номера строки, в которой находится элемент, и номера столбца. Таким образом, номер элемента определяется пересечением строки и столбца. Например, a 21 – это элемент, стоящий во второй строке и в первом столбце.

**Описание двумерного массива Паскаля.**

Существует несколько способов объявления двумерного массива Паскаля.

Мы уже умеем описывать одномерные массивы, элементы которых могут иметь любой тип, а, следовательно, и сами элементы могут быть массивами. Рассмотрим следующее описание типов и переменных:

**Пример описания двумерного массива Паскаля**

Type
Vector = array [1..5] of <тип\_элементов>;
Matrix= array [1..10] of vector;
Var m: matrix;

Мы объявили двумерный массив Паскаля m, состоящий из 10 строк, в каждой из которых 5 столбцов. При этом к каждой i -й строке можно обращаться m [ i ], а каждому j -му элементу внутри i -й строки – m [ i , j ].

Определение типов для двумерных массивов Паскаля можно задавать и в одной строке:

Type
Matrix= array [1..5] of array [1..10] of < тип элементов >;
или еще проще:
type
matrix = array [1..5, 1..10] of <тип элементов>;

Обращение к элементам двумерного массива имеет вид: M [ i , j ]. Это означает, что мы хотим получить элемент, расположенный в i -й строке и j -м столбце. Тут главное не перепутать строки со столбцами, а то мы можем снова получить обращение к несуществующему элементу. Например, обращение к элементу M [10, 5] имеет правильную форму записи, но может вызвать ошибку в работе программы.

**Основные действия с двумерными массивами Паскаля**

Все, что было сказано об основных действиях с одномерными массивами, справедливо и для матриц. Единственное действие, которое можно осуществить над однотипными матрицами целиком – это присваивание. Т.е., если в программе у нас описаны две матрицы одного типа, например,

type
matrix= array [1..5, 1..10] of integer;
var
   a , b : matrix ;

то в ходе выполнения программы можно присвоить матрице *a* значение матрицы *b* ( a := b ). **Все остальные действия выполняются поэлементно, при этом над элементами можно выполнять все допустимые операции, которые определены для типа данных элементов массива.** Это означает, что если массив состоит из целых чисел, то над его элементами можно выполнять операции, определенные для целых чисел, если же массив состоит из символов, то к ним применимы операции, определенные для работы с символами.

**Ввод двумерного массива Паскаля.**

Для последовательного ввода элементов одномерного массива мы использовали цикл for, в котором изменяли значение индекса с 1-го до последнего. Но положение элемента в двумерном массиве Паскаля определяется двумя индексами: номером строки и номером столбца. Это значит, что нам нужно будет последовательно изменять номер строки с 1-й до последней и в каждой строке перебирать элементы столбцов с 1-го до последнего. Значит, нам потребуется два цикла for , причем один из них будет вложен в другой.

Рассмотрим пример ввода двумерного массива Паскаля с клавиатуры:

**Пример программы ввода двумерного массива Паскаля с клавиатуры**

type
   matrix= array [1..5, 1..10] of integer;
var
   a, : matrix;
   i, j: integer; { индексы массива }
begin
   for i :=1 to 5 do {цикл для перебора всех строк}
      for j :=1 to 10 do {перебор всех элементов строки по столбцам}
         readln ( a [ i , j ]); {ввод с клавиатуры элемента, стоящего в i -й строке и j -м столбце}

Двумерный массив Паскаля можно заполнить случайным образом, т.е. использовать функцию random (N), а также присвоить каждому элементу матрицы значение некоторого выражения. Способ заполнения двумерного массива Паскаля выбирается в зависимости от поставленной задачи, но в любом случае должен быть определен каждый элемент в каждой строке и каждом столбце.

**Вывод двумерного массива Паскаля на экран.**

Вывод элементов двумерного массива Паскаля также осуществляется последовательно, необходимо напечатать элементы каждой строки и каждого столбца. При этом хотелось бы, чтобы элементы, стоящие в одной строке, печатались рядом, т.е. в строку, а элементы столбца располагались один под другим. Для этого необходимо выполнить следующую последовательность действий (рассмотрим фрагмент программы для массива, описанного в предыдущем примере):

**Пример программы вывода двумерного массива Паскаля**

for i :=1 to 5 do {цикл для перебора всех строк}
begin
   for j :=1 to 10 do {перебор всех элементов строки по столбцам}
      write ( a [ i , j ]:4); {печать элементов, стоящих в i -й строке матрицы в одной экранной строке, при этом для вывода каждого элемента отводится 4 позиции}
   writeln ; {прежде, чем сменить номер строки в матрице, нужно перевести курсор на начало новой экранной строки}
end ;

**Замечание (*это важно!*):** очень часто в встречается ошибка, когда ввод с клавиатуры или вывод на экран массива пытаются осуществить следующим образом: readln (a), writeln (a), где *а* – это переменная типа массив. При этом их удивляет сообщение компилятора, что переменную этого типа невозможно считать или напечатать. Может быть, вы поймете, почему этого сделать нельзя, если представите N кружек, стоящих в ряд, а у вас в руках, например, чайник с водой. Можете вы по команде «налей воду» наполнить сразу все кружки? Как бы вы ни старались, но в каждую кружку придется наливать отдельно. Заполнение и вывод на экран элементов массива также должно осуществляться последовательно и поэлементно, т.к. в памяти ЭВМ элементы массива располагаются в последовательных ячейках.

**Представление двумерного массива Паскаля в памяти**

Элементы абстрактного массива в памяти машины физически располагаются последовательно, согласно описанию. При этом каждый элемент занимает в памяти количество байт, соответствующее его размеру. Например, если массив состоит из элементов типа integer , то каждый элемент будет занимать по два байта. А весь массив займет S^2 байта, где S – количество элементов в массиве.

А сколько места займет массив, состоящий из массивов, т.е. матрица? Очевидно: S i^S j , где S i - количество строк, а S j – количество элементов в каждой строке. Например, для массива типа

Matrix = array [1..3, 1..2] of integer ;

потребуется 12 байт памяти.

Как будут располагаться в памяти элементы этого массива? Рассмотрим схему размещения массива M типа matrix в памяти.



Под каждый элемент M [i,j] типа integer выделяется две ячейки памяти. Размещение в памяти осуществляется «снизу вверх». Элементы размещаются в порядке изменения индекса, что соответствует схеме вложенных циклов: сначала размещается первая строка, затем вторая, третья... Внутри строки по порядку идут элементы: первый, второй и т.д.

Как мы знаем, доступ к любой переменной возможен, только если известен адрес ячейки памяти, в которой хранится переменная. Конкретная память выделяется для переменной при загрузке программы, то есть устанавливается взаимное соответствие между переменной и адресом ячейки. Но если мы объявили переменную как массив, то программа «знает» адрес начала массива, то есть первого его элемента. Как же происходит доступ ко всем другим элементам массива? При реальном доступе к ячейке памяти, в которой хранится элемент двумерного массива, система вычисляет ее адрес по формуле:

Addr + SizeElem \* Cols \*( I -1)+ SizeElem \*( J -1),

где Addr – фактический начальный адрес, по которому массив располагается в памяти; I , J – индексы элемента в двумерном массиве; SizeElem – размер элемента массива (например, два байта для элементов типа integer ); Cols – количество элементов в строке.

Выражение SizeElem \* Cols \*( I -1)+ SizeElem \*( J -1) называют смещением относительно начала массива.

**Сколько памяти выделяется для массива?**

Рассмотрим не столько вопрос о том, сколько памяти выделяется под массив (это мы разобрали в предыдущем разделе), а о том, каков максимально допустимый размер массива, учитывая ограниченный объем памяти.

Для работы программы память выделяется сегментами по 64 Кбайт каждый, причем как минимум один из них определяется как **сегмент данных**. Вот в этом-то сегменте и располагаются те данные, которые будет обрабатывать программа. Ни одна переменная программы не может располагаться более чем в одном сегменте. Поэтому, даже если в сегменте находится только одна переменная, описанная как массив, то она не сможет получить более чем 65536 байт. Но почти наверняка, кроме массива в сегменте данных будут описаны еще некоторые переменные, поэтому реальный объем памяти, который может быть выделен под массив, находится по формуле: 65536- S , где S – объем памяти, уже выделенный под другие переменные.

Зачем нам это знать? Для того чтобы не удивляться, если при компиляции транслятор выдаст сообщение об ошибке объявления слишком длинного массива, когда в программе встретит описание (правильное с точки зрения синтаксиса):

Type myArray= array [1..50000] of integer;

Вы уже знаете, что, учитывая двухбайтовое представление целых чисел, реально можно объявить массив с количеством элементов равным 65536/2 –1=32767. И то лишь в том случае, если других переменных не будет. Двумерные массивы должны иметь еще меньшие границы индексов.

**Примеры решения задач с двумерными массивами Паскаля**

**Задача:** Найти произведение ненулевых элементов матрицы.

**Решение:**

* Для решения данной задачи нам потребуются переменные: матрица, состоящая, например, из целочисленных элементов; P – произведение элементов, отличных от 0; I , J – индексы массива; N , M – количество строк и столбцов в матрице.
* Входными данными являются N , M – их значения введем с клавиатуры; матрица – ввод матрицы оформим в виде процедуры, заполнение матрицы осуществим случайным образом, т.е. с помощью функции random ().
* Выходными данными будет являться значение переменной P (произведение).
* Чтобы проверить правильность выполнения программы, необходимо вывести матрицу на экран, для этого оформим процедуру вывода матрицы.
* Ход решения задачи:

обсудим сначала выполнение основной программы, реализацию процедур обговорим чуть позже:

* введем значения N и M ;
* Введем двумерный массив Паскаля, для этого обращаемся к процедуре vvod ( a ), где а – матрица;
* Напечатаем полученную матрицу, для этого обращаемся к процедуре print ( a );
* Присвоим начальное значение переменной P =1;
* Будем последовательно перебирать все строки I от 1-й до N -й, в каждой строке будем перебирать все столбцы J от 1-го до M -го, для каждого элемента матрицы будем проверять условие: если a ij ? 0, то произведение P будем домножать на элемент a ij ( P = P \* a ij );
* Выведем на экран значение произведения ненулевых элементов матрицы – P ;

А теперь поговорим о процедурах.

**Замечание *(это важно!)*** Параметром процедуры может быть любая переменная предопределенного типа, это означает, что для передачи в процедуру массива в качестве параметра, тип его должен быть описан заранее. Например :

Type
Matrix=array [1..10, 1..10] of integer;
..............................
procedure primer (a: matrix);
..............................

Вернемся теперь к нашим процедурам.

Процедура ввода матрицы называется vvod , параметром процедуры является матрица, причем она должна быть, как результат, передана в основную программу, следовательно, параметр должен передаваться по ссылке. Тогда заголовок нашей процедуры будет выглядеть так:

Procedure vvod ( var m : matrix );

Для реализации вложенных циклов в процедуре нам потребуются локальные переменные-счетчики, например, k и h . Алгоритм заполнения матрицы уже обсуждался, поэтому не будем его повторять.

Процедура вывода матрицы на экран называется print , параметром процедуры является матрица, но в этом случае она является входным параметром, следовательно, передается по значению. Заголовок этой процедуры будет выглядеть следующим образом:

Procedure print ( m : matrix );

И вновь для реализации вложенных циклов внутри процедуры нам потребуются счетчики, пусть они называются так же – k и h . Алгоритм вывода матрицы на экран был описан выше, воспользуемся этим описанием.

**Пример программы двумерного массива Паскаля**

Program proizvedenie;
Type
   Matrix=array [1..10, 1..10] of integer;
Var
   A: matrix;
   N, m, i, j: byte;
   P: integer;
Procedure vvod (var m: matrix);
Var k , h : byte ;
Begin
   For i :=1 to n do {переменная n для процедуры является глобальной, а значит «известной»}
      For j :=1 to m do {переменная m для процедуры является глобальной, а значит «известной»}
         M[i,j]:= random(10);
End;
Procedure print (m: matrix);
Var k, h: byte;
Begin
   For i:=1 to n do
   begin
      For j:=1 to m do
         Write (M[i, j]: 4);
      Writeln;
   end ;
End ;
Begin {начало основной программы}
   Writeln (‘Введите размерность матрицы:’);
   Readln(N, M);
   Vvod(a);
   Print(a);
   P:=1;
   For i:=1 to N do
      For j:=1 to M do
         If a[i, j]<>0 then p:=p\*a[i, j];
   Writeln ( p );
End .

**Используемая литература**

Концепция федеральных государственных образовательных стандартов общего образования: проект / Рос.акад. образования; под ред. А.М. Кондакова, А.А. Кузнецова. — М.: Просвещение, 2008.

http://www.ed.gov.ru/ob-edu/noc/rub**/**standart**/**

Словарь-справочник по педагогике. Автор-составитель В.А. Мижериков, под ред. П.И. Пидкасистого, М. 2004, с.197.

1. Валькова Г., Зайнуллина Ф., Штейнберг В. Логико-смысловые модели - дидактическая многомерная технология / В. // ДИРЕКТОР ШКОЛЫ: науч.-метод. журн. для рук. учеб. заведений и органов образования. - 2009. - № 1. - C.49-54

Громыко Ю.В. "Метапредмет "Знак".- М., 2001.- 285 с.

Громыко Н.В. "Метапредмет "Знание".- М., 2001.- 540с.

Громыко Ю.В. "Метапредмет "Проблема".- М., 1998. – 376 с.

1. Колесина К.Ю. Метапроектное обучение: теория и технологии реализации в учебном процессе: Автореф. дисс. … д-ра пед. наук: 13.00.01. Ростов-на-Дону: ЮФУ, 2009. 35 с.
2. Кузнецов А.А. О школьных стандартах второго поколения / А.А. Кузнецов. // Муниципальное образование: инновации и эксперимент. - 2008. - № **2**. - С. 3-6.
3. Ковалева Г.С., Красновский Э.А., Краснянская К.А., Логинова О.Б., Татур О.А. Модель системы оценки результатов освоения общеобразовательных программ. /www. standart. edu. ru/.

Федорова С.Ш. Технология присвоения метазнаний /http://festival.1september.ru/articles/100689/.

Фоменко И.А. Создание системы формирования нового содержания образования на основе принципов метапредметности/fomenko.edusite.ru/p35aa1.html/.

ХуторскойА.В. Эвристический тип образования: результаты научно-практического исследования // Педагогика. – 1999. - №7. – С.15-22.