ОГБПОУ «Сасовский индустриальный колледж

имени полного кавалера ордена Славы В.М.Шемарова»

*Доклад*

*на тему: «Возможности*

 *использования ТРИЗ в информатике»*

Подготовила: преподаватель

информатики

Ларюшина И.А.

Город Сасово

2017г.

Возможности использования ТРИЗ в информатике

Все педагогические приёмы можно разделить на две большие группы:

приёмы, рассчитанные преимущественно на запоминание,

и приёмы, рассчитанные преимущественно на развитие.

 Разумеется, любой педагог стремится использовать оба эти приёма в своей работе, и они объективно оба нужны. Вопрос состоит не в том, какой приём выбрать, а в балансе этих приёмов. Они должны сочетаться, желательно — гармонично. Обычно акцент на том или ином приёме делается на разных этапах обучения. Оба приёма по сути являются двумя сторонами одной медали. Первый приём носит консервативный характер: он фиксирует полученные знания, делает картину ясной, стабилизирует положение, позволяет увидеть отношения между известными фактами. Второй приём раскачивает привычную картину, дестабилизирует её, разрушает целостность имеющихся знаний и выводит на новый уровень взаимодействия фактов, создавая, тем самым, новые знания. Это динамический этап, этап прорыва на следующий уровень понимания. Стало быть, задача состоит в правильном чередовании этих этапов и их комплексном применении.

Мотивация у второго приёма бывает разной. Может быть такой: давай разрушим и посмотрим, а вдруг можно будет получить что-то новое. Так на своих уроках я использую видеоролики, объясняющие устройство вычислительной техники: принтер, сканер, монитор, мышь и т.д., чтобы студенты могли увидеть, что там внутри, т. е. получить новые знания. Но может быть и такой: надо получить вот это, давай разрушим и перестроим так, чтобы получить именно это. Например сравнение структуры и принципов работы мониторов: ЖК и плазменной панели. Первую мотивацию можно назвать «методом тыка» (или «научного тыка»), а вторую — «движением на свет маяка». Эффективной может оказаться любая из них, но вторая как-то больше нравится: она более осмысленная, более целеустремлённая, более экономная, в конце концов, она прямо нацелена на результат. Есть даже смутная догадка, что каждая эпоха склонна к одному определённому приёму обучения: либо консервативному, либо динамическому. Нынешняя эпоха, на мой взгляд, начинает тяготеть к динамическому

Технологий, стимулирующих получение новых знаний, к сожалению, немного, и используются они не так активно, как хотелось бы. К тому же многие из этих технологий не ставят прямо задачу: добыть новые знания. Чаще эта задача решается опосредованно. К примеру, метод проектов не ставит прямо задачу добыть знания. Цель другая — получить новый продукт (не важно — материальный, интеллектуальный или какой-то другой), а вот если потребуются для этого новые знания — добывай их! Или, скажем, игровые технологии также не ставят прямо задачу получить новые знания, игра — это только мотивация к мыслительной деятельности (но, правда, не только к ней — эмоциональная, коммуникативная и другие сферы тоже развиваются). Вот технология ТРИЗ (Теория решения изобретательских задач) напрямую ставит задачу получения новых знаний, идей. Причём, даёт методы эффективного, малозатратного, прямого их получения. Вообще, её автор Генрих Саулович Альтшуллер поставил фантастическую задачу — ввести искусство изобретательства в рамки науки («поверить алгеброй гармонию»). Однако, в области массового образовании Теория решения изобретательских задач, которую прозвали «русским чудом», до сих пор не получила должного распространения. В чём причина не могу сказать. Возможно, это из-за необычности метода, возможно — из-за мощи его инструментария, возможно — из-за недостаточной проработанности. Но оно и понятно: загнать творчество в рамки науки непросто, а обучить этой науке — вдвойне сложнее. Попытки делаются, однако, и небезуспешные.

Теория решения изобретательских задач задумывалась для изобретательской сферы. Однако, идеи и методы, заложенные в ней, оказались настолько универсальными, что их стали использовать в других областях знаний. Разумеется, информатика не стала исключением. А.Гин утверждает: «Обучение, построенное на усвоении конкретных фактов, изжило себя в принципе, ибо факты быстро устаревают, а их объем стремится к бесконечности». Это означает, что картина мира пришла в движение, стала менее устойчивой. Для образования это значит, что «от педагогики памяти она будет переходить к педагогике развития», и в основе преподавания будет лежать обучение мышлению. Цель ТРИЗ — выявление и использование законов, закономерностей и тенденций развития технических систем. При этом полагается, что такие законы объективно существуют, и, стало быть, поддаются постижению. Коротко задачу ТРИЗ можно обозначить следующим образом: – основным методом решения творческих задач до сих пор был перебор вариантов («метод тыка»); – существующие методики (мозговой штурм, метод контрольных вопросов, морфологический разбор и т. д.) позволяли просто ускорить этот перебор; – ТРИЗ ставит задачу: без сплошного перебора вариантов выйти сразу на так называемые сильные решения.

Соответственно, ТРИЗ-педагогика призвана сделать процесс мыслительной деятельности:

а) целенаправленным,

б) осознанным,

в) управляемым.

Перечислим основные принципы ТРИЗ:

– принцип объективности законов развития систем — строение, функционирование и развитие систем подчиняются объективным законам. (Сильные решения — это решения, соответствующие объективным законам и явлениям);

– принцип противоречия — под воздействием внешних и внутренних факторов возникают и разрешаются противоречия. Системы развиваются, преодолевая противоречия. (Сильные решения — это решения, преодолевающие противоречия);

– принцип конкретности — каждый класс систем имеют свои особенности, облегчающие или затрудняющие изменение этой системы. (Сильные решения — это решения, учитывающие индивидуальные особенности систем).

Отметим также, что ТРИЗ включает в себя:

– механизмы трансформирования проблемы в будущее решение;

 – механизмы блокирования психологической инерции, препятствующей поиску решений (нестандартные решения невозможно находить без преодоления господствующих стереотипов);

– обширный информационный фонд — концентрированный опыт решения проблем.

Наиболее важные понятия ТРИЗ следующие:

– Противоречие. (Система должна обладать свойством А, чтобы повысить эффективность, и не должно обладать этим свойством, чтобы не навредить). Требуется точная формулировка противоречия, ибо она позволит понять корень проблемы.

– Ресурсы. Вначале пробуем использовать те ресурсы, которые уже есть.

Для нас важны:

а) информационные ресурсы,

б) ресурсы времени,

в) человеческие ресурсы.

– ИКР (Идеальный Конечный результат). Решать задачу проще, когда сформулирован идеальный ответ.

Наиболее эффективны варианты:

• система сама выполняет нужную функцию;

• системы нет, а функции ее выполняются (с помощью ресурсов);

• функция не нужна.

В итоге алгоритм решения задачи с помощью методов ТРИЗ вырисовывается следующий:

– определяем тип задачи (изобретательская, исследовательская, конструкторская, прогнозная);

– формулируем к задаче противоречие и идеальный конечный результат (содействуем обострению проблемы и продвигаемся к сильным решениям);

– выявляем ресурсы (сначала — внутренние, потом — дешёвые, потом — остальные);

– применяем приёмы и принципы решения задач (если до этого пункта решение ещё не появилось);

– анализируем решение (с позиции идеальности).

Итак, поскольку ни один из названных принципов и алгоритмов не отвергается ни информатикой вообще, ни программированием в частности, пробуем применить их в этой области.

Пример 1.

Переход от линейного к процедурному программированию.

При написании сложных программ возникает противоречие: усложнение задачи требует всё большего программного кода. Программа разрастается как снежный ком. При этом падает обозреваемость кода, стирается ясность алгоритмических решений, затрудняется взаимодействие частей программы, возрастает вероятность появления ошибок, которые трудно найти.

Перед нами очевидное противоречие: код усложнять нельзя, так как программа становится слишком громоздкой; код усложнять надо, так как этого требует уровень сложности задачи. Как быть?

Попробуем использовать методы ТРИЗ.

 Идеальный конечный результат должен представляться так: код может быть сложным, но негативные стороны этого усложнения не проявляются.

 Исходим из того, что любая сложная задача состоит из более простых подзадач, т. е. любая сложная задача внутренне структурирована. А что, если внутреннюю структуру задачи отразить в программе, т. е. простые подзадачи оформлять отдельными программными кусками. В программировании есть понятие подпрограммы — это автономная часть программы, которая как раз решает некоторую частную задачу. В большинстве языков программирования эту роль играют процедуры и функции. Можно попробовать использовать эти возможности.

 Встаёт, однако, вопрос, каков должен быть механизм взаимодействия процедуры и основной программы? И насколько независимой от программы должна быть процедура? Так вот — максимально независима. Почему? Чем меньше зависит подпрограмма от программы, тем выше её универсальность, т. е. возможность использования в других программах.

Это парадигма процедурного программирования. А информационная связь процедуры и программы осуществляется через механизм передачи параметров (список параметров обычно ставится в скобках за именем процедуры). Информацию через параметры можно передавать как в процедуру (при её вызове), так и из процедуры (при возврате из процедуры). Использование процедур и функций даёт возможность разрабатывать программу по частям. Так легче и писать программу, и отлаживать её. Кроме того, легко можно распараллелить работу (т. е. отдельные части программы могут писать и отлаживать разные люди).

Наконец, из процедур и функций можно составлять библиотеки для использования в других программах. В итоге программирование сложной задачи сводится к адекватному разбиению её на отдельные подзадачи и корректное оформление их в виде процедур и функций.

Пример 2.

Исполнитель преобразует некое целое число. Он может выполнять три команды: 1) прибавить 1; 2) прибавить 2; 3) прибавить предыдущее.

Сколько существует программ, которые число 2 преобразуют в число 15?

Стандартный подход — перебор всех возможных вариантов. Недостатки метода очевидны: высокие трудозатраты и высокая вероятность ошибки (возможность пропустить какой-либо вариант). Кроме того, объём вычислений резко возрастает даже при небольшом увеличении входных данных. Упорядочивание же перебора только немного облегчит мучения, но не отменит их полностью. В данном случае предпочтительнее применить так называемый динамический метод. Действуем строго по правилам ТРИЗ: производим дробление задачи, решаем её пошагово, промежуточные результаты не пересчитываем. И самое главное, работаем только с количеством вариантов, а не с их генерацией (ведь в задаче не сказано сгенерировать варианты, а требуется определить только количество вариантов). Решение задачи сводится только к заполнению динамической таблицы по очевидным формулам: для чётного числа: Ai=Ai-1+Ai-2 для нечётного числа: Ai=Ai-1+Ai-2+A(i+1)/2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 1 | 2 | 4 | 6 | 12 | 18 | 34 | 52 | 92 | 144 | 248 | 392 | 658 |

Пример 3.

Фишка может двигаться по полю из N клеток только вперёд. Длина хода фишки не более K клеток за один раз. Найти число различных вариантов пути, с помощью которых фишка достигнет конца поля.

Например, для N=3, K=2 возможны пути: 1 1 1; 1 2; 2 1. Ответ: 3.

Так же, как и в предыдущем примере, прямой перебор вариантов совершенно бесперспективен (хотя и может быть приемлем при небольших размерностях). Просмотр вариантов также можно организовать с помощью приёма «перебор с возвратом».

 procedure go(i:integer); {в i - текущий номер поля с фишкой}

var j:integer; {j - длина очередного хода фишки}

begin if i=n then inc(c) {считаем вариант, если дошли до конца поля}

else for j:=1 to k do if i+j<=n then go(i+j); {генерацияслед. варианта}

end;

Недостаток метода бросается в глаза сразу: ненужная генерация вариантов (которая в задаче не требуется, но которая съедает время) и многочисленный пересчет уже обработанных вариантов, причём второй недостаток резко возрастает при увеличении размерности. И вновь использование методов ТРИЗ позволит приблизить решение задачи к Идеальному Конечному Результату. Выходя за рамки привычного, опять применим динамический метод: разбиваем общую задачу на частные для каждой клетки, решаем их, пошагово приближаясь к конечному результату, данные при этом заносим в таблицу, эти промежуточные результаты больше не пересчитываем. И по-прежнему, работаем только с количеством вариантов.

Решение будет выглядеть примерно так:

for i:=1 to n do a[i]:=0; {очисткамассива}

a[1]:=1; {число вариантов для первой клетки}

for i:=2 to n do {заполнение динамической таблицы} if i<=k then begin {пока длина хода меньше k} a[i]:=1;

for j:=1 to i-1 do a[i]:=a[i]+a[j];

end else begin {длина хода больше k} a[i]:=0;

for j:=i-k to i-1 do a[i]:=a[i]+a[j];

end; write(a[n]); {печать результата}

Заметим, что вычислительная ёмкость этого метода пропорциональна размерности входных данных, т. е. имеет очень хороший показатель. В заключении, подводя итоги констатируем:

– методы ТРИЗ вполне применимы в такой науке как информатика, в том числе в разделе «программирование», т. е. их можно смело использовать на уроках;

– методы ТРИЗ пробуждают творческое отношение студентов к стоящим задачам и проблемам;

– методы ТРИЗ подходят ко всем студентам, т. е. они не требуют каких-то исключительных знаний или необыкновенных способностей;

– тем не менее нужно отметить, что эти методы больше востребованы студентами с высоким творческим потенциалом; при использовании этих методов, они как бы попадают в родную стихию и с большей свободой и увлечением пользуются её арсеналом;

 – неоспоримым преимуществом ТРИЗ является его прямое побуждение к действию нестандартными методами, широкий охват привлекаемых для решения задач ресурсов, т. е. всего того, что даёт мощный стимул для развития мыслительной деятельности подростка.

Пример 4.

Задачи по методу словотворчества

Цель: закрепить, проверить знания учащихся по пройденной теме, развитие памяти, внимания.

**Описание:** в “ящике” с буквами спрятано слово. Надо найти это слово, для чего необходимо правильно совместить понятия по вертикали и горизонтали.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | пробел | Морзе | диалог | поле | оператор | слово | Евклид  | Паскаль Си и др. |
| Самая длинная клавиша | А | К | Р | О | Л | Д | Ж | Т |
| Изобретатель системы кодирования информации | Ф | Б | Л | Д | Т | Ь | И | Я |
| Два смежных байта оперативной памяти | Ф | Ц | У | К | Е | Л | Й | З |
| Автор самого древнего алгоритма | Р | Е | П | М | Т | А | О | Э |
| Язык программирования высокого уровня | Л | Ж | Д | Э | Х | Щ | Ш | Т |
| Форма взаимодействия пользователя с компьютером | Ц | У | И | Е | Н | Г | Ш | Б |
| Элементарная единица программы | Ю | К | О | А | К | П | Д | Р |
| Компонент данных типа запись | Б | Ж | Ъ | Й | С | В | Л | О |

Ответ: килобайт.

Литература:

Альтшуллер Г. С. Творчество как точная наука. — М.: Советское радио, 1979.

Альтшуллер Г. С. Найти идею. — Новосибирск: Наука, 1991 (2-е изд.).

Гин А. А. Приёмы педагогической техники. — М.: «Вита-Пресс», 1999. — 88 с.; 6-е изд., доп., 2005. — 112 с., 9-е изд., 2009. — 112 с.

Гин, А. А. ТРИЗ для всех (опыт кружковой работы). Журнал «Школьные технологии», М.: Народное образование, 2007. № 5. С. 127–128.

Злотин Б. Л., Зусман А. В. Использование аппарата ТРИЗ для решения исследовательских задач. — Кишинев: Картя молдовеняскэ, 1985. — 211 с.