Conference on Computer Simulation and Information Technology in Education (KMITO 1999) 19-21 April 1999, Kryvyi Rih, Ukraine / Edited by : Vladimir N. Soloviev, Illia O. Teplytskyi, Serhiy O. Semerikov // ACNS Conference Series: Social Sciences and Humanities.– 2022.– Volume 1.– 49 p.– DOI : https://doi.org/10.55056/cs-ssh/1

Belief revision as a problem of scientific epistemology

Yaroslav V. Shramko[1]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

Abstract. The fundamental question that must be answered by any theory of knowledge that claims to be adequate is the question of how it is possible to change our knowledge. The very fact of change undoubtedly takes place, and the problem is to theoretically explicate this fact. The methodological significance of this issue is due to the fact that changing knowledge means nothing more than its development, namely, the question of the ways and means of developing our knowledge is of central importance both for the logic and methodology of science, and for general epistemology. This work is of a review character, and aims to draw the reader's attention to a new promising direction in the modern theory of knowledge called "belief revision".

Keywords: belief revision, scientific epistemology

Основополагающим вопросом, на который должна ответить всякая теория познания, претендующая на адекватность, является вопрос о том, как возможно и каким образом осуществляется изменение нашего знания. Сам факт изменения несомненно имеет место, и проблема заключается в том, чтобы теоретически эксплицировать данный факт. Методологическая значимость этого вопроса обусловлена тем обстоятельством, что изменение знания означает не что иное, как его *развитие*, а именно вопрос о путях и способах развития нашего знания имеет центральное значение как для логики и методологии науки, так для общей эпистемологии.

Настоящая работа носит обзорный характер и имеет целью привлечь внимание читателя к новому перспективному направлению в современной теории познания, за которым в англоязычной литературе закрепилось название "belief revision".

Прежде всего представляется уместным уточнить некоторые ключевые термины, которые использованы в заглавии. Под "ревизией" знания следует понимать его *пересмотр*. Очевидно, что время от времени в силу различных причин (например, в результате изменения нас самих или изменения окружающей нас действительности) мы подвергаем наши знания пересмотру с целью решить, какие из этих знаний устарели и должны быть отброшены, а какие нужно сохранить и, быть может, развить дальше. Таким образом, если мы хотим получить ответ на вопрос, как происходит изменение нашего знания, мы должны объяснить, каковы те правила и методы, по которым осуществляется его пересмотр. Второе, еще более важное уточнение относится к самому термину "знание". Очевидно, что ограничение сущностей образующих то, что может быть названо *эпистемическим состоянием* субъекта, только сферой

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

shramko@rocketmail.com (Y. V. Shramko)
https://kdpu.edu.ua/shramko (Y. V. Shramko)

ACNS Conference Series: Social Sciences and Humanities

знания, означало бы неоправданное сужение самого предмета теории познания. При построении теоретической модели познавательной деятельности необходимо учитывать не только те ее результаты, которые обладают стопроцентной достоверностью и являются несомненно истинными (то есть, собственно знание), но также и все те положения, которые субъект только *считает истинными*. Совокупность таких положений может быть названо "мнением" субъекта (по тому или иному вопросу), или его "убеждениями". По-видимому, правильным будет предположить, что знания субъекта составляет собственное подмножество множества его убеждений. Здесь важно отвлечься от той эмоциональной окрашенности, с которой обычно связано употребление слова "убеждение" в обыденном (русском) языке (в таких словосочетаниях как "идейные убеждения", "стойкие убеждения" и т.п.). В контексте настоящей работы данный термин лишен какой бы то ни было этической или идеологической нагрузки и понимается исключительно в смысле "те положения, которые субъект в данный момент считает истинными". Учитывая, однако, то обстоятельство, что в некоторых случаях бывает крайне трудно полностью отвлечься от нежелательных ассоциаций эмоционально-этического характера, время от времени мы все-же будем употреблять термин "знание", придавая ему тот расширительный смысл, о котором речь шла выше.

Итак, далее речь пойдет о том, каким образом мы пересматриваем наши убеждения. При этом мы не будем затрагивать вопрос, *почему* мы это делаем. Вопрос о причинах, которые иногда побуждают нас пересмотреть (подвергнуть ревизии) то, во что мы ранее верили, выходит за рамки теории познания и не является собственно эпистемологическим вопросом. Также второстепенное значение для чистой теории познания имеет вопрос о той "действительности", к которой относятся наши знания. Вопрос о природе такого рода действительности равно как и вопрос о самом ее существовании представляет собой в сущности философски открытую проблему, которая вряд ли может быть решена окончательно. По крайней мере, она не может быть решена в рамках одной лишь теории познания. Поэтому при построении эпистемологических моделей целесообразно вообще не поднимать такого рода метафизические проблемы. Нас также не будут интересовать психолингвистические (или психофизические) процессы, происходящие в голове субъекта когда он меняет свое мнение, и обеспечивающие психологический механизм такого изменения. Эпистемология рассматривает знания и убеждения субъекта как некоторое *объективированное знание*, как мир объективного содержания мышления ("третий мир" Карла Поппера). Понимаемое таким образом знание представляет собой некоторую (и довольно сильную) эпистемологическую идеализацию необходимую для создания когнитивной модели познавательной деятельности. При таком понимании, вопрос о конкретном материальном носителе убеждений перестает быть существенным и наличие человеческого мозга вообще не является обязательным. Такого рода знание может быть смоделировано как результат деятельности "идеального субъекта", например, как состояние компьютера или компьютерной программы.

Основными понятиями, образующими каркас рассматриваемой когнитивной модели, являются понятие "эпистемического состояния" и понятие "познавательной операции". Первое из этих понятий служит для представления возможного состояния познающего субъекта в некоторый момент времени. Мы предполагаем, что такого рода состояние

является заданным, если нам известны все те положения, которые индивид принимает в данный момент времени, то есть, в истинности которых он убежден. Таким образом, эпистемическое состояние субъекта есть ни что иное как множество его убеждений. С логической точки зрения это множество может быть описано как некоторое множество высказываний (а именно, множество всех тех высказываний, относительно которых субъект *верит*, что они являются истинными). Важно отметить, что эпистемология имеет дело с *рациональным* субъектом, то есть, субъектом, познавательная деятельность которого организована рациональным образом. В этой связи возникает вопрос о том, когда субъект может считаться рациональным, а это есть вопрос о критериях рациональности, которым должны подчиняться эпистемические состояния субъекта. Обычно принимаются следующие два требования рациональности:

(1) Множество убеждений субъекта должно быть непротиворечивым.
(2) Субъект обязан принимать все логические следствия принимаемых им убеждений.

Убеждения, удовлетворяющие данным требованиям, считаются рациональными. Эти требования являются очевидно довольно сильными идеализациями. Так, например, в действительности убеждения субъекта иногда (а возможно и часто) могут противоречить друг другу. Мы однако считаем, что противоречивые убеждения не представляют особого теоретического интереса, поскольку не совсем ясно, каким образом такого рода убеждения могут быть подвергнуты рациональному анализу. Поэтому, если вдруг обнаруживается, что множество убеждений индивида является противоречивым, то такое положение дел считается ненормальным и рациональный индивид, в соответствии с требованием непротиворечивости, обязан предпринять все необходимые действия для устранения противоречия, или, по крайней мере, для его изоляции. Что касается второго требования, то его не следует понимать в том смысле, что субъект действительно *осознает* все логические следствия своих убеждений. Скорее, это требование отражает *эпистемические обязательства* рационального субъекта. Так, например, если индивид верит, что все люди смертны, а также верит, что Сократ человек, то тогда он *обязан* принять утверждение, что Сократ смертен, даже если он явным образом никогда не задумывался над этим последним вопросом. Если же этот индивид, вопреки своим первым двум убеждениям, будет отказываться принять истинность последнего утверждения, то такой индивид будет признан нами нерациональным (или иррациональным), что, очевидно, является вполне обоснованным.

Второе требование иногда формулируется еще и следующим образом:

(2') Множество убеждений субъекта должно быть замкнуто по отношению логического следования.

Формально это может быть представлено с помощью особой *операции замыкания* – Cn. Пусть X есть некоторое множество высказываний. Тогда $Cn(\mathrm{X})$ есть множество всех логических следствий из X, которое называется замыканием X. Cn должна удовлетворять следующим стандартным условиям:

(a) $\mathrm{X} \subseteq Cn(\mathrm{X})$;

(b) Если $\mathrm{X} \subseteq \mathrm{Y}$, то $Cn(\mathrm{X}) \subseteq Cn(\mathrm{Y})$;
(c) $Cn(\mathrm{X}) = Cn(Cn(\mathrm{X}))$.

Используя операцию замыкания, мы вводим следующее понятие "системы убеждений":

Определение 1.
X есть (неабсурдная) система убеждений если и только если $\mathrm{X} = Cn(\mathrm{X})$.

Следующее важное понятие – это понятие "познавательной операции" или "познавательного действия". Именно это понятие дает нам возможность отразить основные типы изменения наших систем убеждений. Пусть K есть некоторая система убеждений. Тогда относительно K возможны следующие познавательные операции, которые приводят к изменению K:

1. *Расширение.* Эта операция применяется, когда мы хотим расширить наши убеждения за счет добавления новых убеждений к уже имеющимся. При этом мы надеемся, что получающаяся в результате новая система убеждений является непротиворечивой, хотя одна лишь операция расширения знаний *сама по себе*, конечно, не может этого гарантировать. Обозначим операцию расширения знаний посредством "+". То есть, если K – имеющаяся система убеждений, а A – некоторое высказывание, то $\mathrm{K} + A$ есть результат расширения K посредством высказывания A.

2. *Сокращение.* Эта операция применяется, когда мы считаем нужным отказаться от некоторого убеждения, то есть, когда мы удаляем это убеждение из нашей системы убеждений. Эта операция обозначается посредством "÷": $\mathrm{K} \div A$ есть результат сокращения системы убеждений K за счет высказывания A.

3. *Ревизия.* Эта операция применяется, если мы пришли к необходимости признать истинность некоторого высказывания, которое является несовместимым с нашей прежней системой убеждений. В этом случае мы добавляем данное высказывание к нашей системе убеждений, и одновременно осуществляем *пересмотр* (ревизию) наших старых убеждений с целью сделать их совместимыми с вновь принятым высказыванием. Если операцию ревизии обозначить посредством "$*$", то тогда $\mathrm{K} * A$ будет результатом ревизии системы убеждений K относительно высказывания A.

Ни одна из этих познавательных операций не сводится к простому механическому одноразовому действию. Так, например, если мы расширяем имеющуюся систему убеждений за счет некоторого высказывания, недостаточно просто добавить это высказывание к множеству старых убеждений. Ведь то, что получится в результате, также должно быть системой убеждений, то есть по *определению 1* она должна быть замкнута по отношению логического следования. Иными словами, при добавлении нового убеждения к уже имеющимся, мы должны добавить сюда также и все логические следствия, которые отсюда вытекают. С другой стороны, если мы осуществляем сокращение наших знаний, недостаточно просто удалить некоторое высказывание из нашей системы убеждений. Дело в том, что мы должны также исключить и все те высказывания, из которых удаляемое высказывание логически следует, поскольку если этого не сделать, то удаляемое высказывание фактически вовсе не будет удалено,

а неявным образом сохранится в системе убеждений. Далее, если два различных высказывания совместно влекут удаляемое убеждение, то одно из этих высказываний также должно быть удалено, и здесь мы оказываемся в ситуации выбора, который далеко не всегда является тривиальным.

Очевидно, что расширение и сокращение знания представляют собой в значительной степени *идеальные* познавательные действия, которые в чистом виде встречаются довольно редко. По-видимому наиболее типичной эпистемической операцией является ревизия, и процесс развития наших убеждений чаще всего происходит именно путем их пересмотра. В этой связи возникает интересный теоретический вопрос – является ли ревизия независимой познавательной операцией и нельзя ли попробовать свести ее к двум другим, то есть определить ревизию через расширение и сокращение. Оказывается, что такое сведение вполне возможно. По существу, операция ревизии представляет собой некоторое комплексное действие, заключающееся в том, что мы должны (1) включить некоторое новое высказывание A в нашу систему убеждений и (2) принять все необходимые меры к тому, чтобы наша новая система убеждений была непротиворечивой. Первое из этих действий достигается путем *расширения* имеющейся системы убеждений за счет A, в то время как вторая цель может быть достигнута посредством *предварительного* удаления $\neg A$ (отрицание A) из нашей системы убеждений (*сокращение*). Иными словами, операция ревизии может быть эксплицирована как результат последовательного осуществления двух подопераций: (1) сокращение посредством $\neg A$ и (2) расширение за счет A. Таким образом, приходим к следующему определению, известному в литературе как “равенство Леви”:

Определение 2.
$\mathrm{K} * A = (\mathrm{K} \div \neg A) + A.$

Это определение имеет очень большое эвристическое значение, поскольку с его принятием проблема теоретической экспликации процесса изменения наших знаний сводится к рассмотрению двух сравнительно простых познавательных операций – расширению и сокращению знаний. Рассмотрим первую из этих операций. Очевидно, что расширение можно довольно легко определить, используя аппарат теории множеств. А именно, если мы хотим расширить нашу систему убеждений K за счет высказывания A, мы должны “механически” добавить это высказывание к K (осуществить теоретико-множественное объединение), а затем замкнуть получившееся множество высказываний K посредством операции замыкания Cn:

Определение 3.
$\mathrm{K} + A = Cn(\mathrm{K} \cup \{A\})$

Посредством данного определения операция расширения знаний вводится однозначным образом, не оставляя пространства для различных ее истолкований. А это значит, что вся проблема ревизии знаний фактически эквивалентна проблеме определения операции сокращения. Как ни парадоксально это звучит, но если мы хотим получить ответ на вопрос о том, каким образом осуществляется изменение (а

значит и развитие) нашего знания, мы должны ответить на вопрос, как происходит его сокращение. Принимая же оптимистическую точку зрения, в соответствии с которой в процессе развития знания происходит его *рост*, мы приходим к следующему кардинальному выводу: *проблема роста знания сводима к проблеме его сокращения.* И если бы нам удалось найти для этой операции такое же четкое определение, как определение 3, то тогда проблема теоретической экспликации развития знания была бы решена однозначным образом.

К сожалению (а может быть и к счастью) однозначно определить операцию сокращения не удается. Основной причиной этого является отмеченная выше возможность "альтернативных ходов", неизбежное появление при осуществлении сокращения ситуации неопределенности, когда мы оказываемся перед выбором, какое из нескольких высказываний удалить из системы наших убеждений, а какое оставить, и при этом не существует никаких чисто логических предпочтений в пользу того или иного высказывания.

Остановимся кратко на некоторых возможных подходах к определению операции сокращения. Пусть X есть некоторое множество высказываний и A – некоторое высказывание. Определим множество $\mathrm{X}\bot A$ (читается "X без A") как множество всех максимальных подмножеств X, таких что они не влекут A. Формально:

Определение 4.

$$\mathrm{У} \in \mathrm{X}\bot A \iff \begin{array}{l}(1)\ \mathrm{У} \subseteq \mathrm{X} \\ (2)\ A \notin Cn(\mathrm{У}) \\ (3)\ \text{не существует множества У' такого, что: } \mathrm{У} \subset \mathrm{У'} \subseteq \mathrm{X} \text{ и } A \notin Cn(\mathrm{У'})\end{array}$$

Теперь можно было бы попробовать определить результат сокращения некоторой системы убеждений К посредством высказывания A как пересечение всех элементов принадлежащих множеству $\mathrm{K}\bot A$:

Определение 5.
$\mathrm{K} \div A = \cap(\mathrm{K}\bot A)$

Определенная таким образом операция сокращения получила в литературе название "сокращение полного пересечения". Нетрудно видеть, что такого рода операция является излишне "перестраховочной", она требует удалять из наших убеждений *слишком* многое, даже если мы этого не хотим. Например, если мы стоим перед выбором – отказаться от одного из каких-либо двух высказываний, то сокращение полного пересечения требует от нас удалить *оба* эти высказывания, что далеко не всегда представляется оправданным. Более того, легко может быть доказана следующая лемма:

Лемма 1. (Алчуррон и Макинсон)
Если "$\div$" есть сокращение полного пересечения и $A \in \mathrm{K}$, то имеем:
$B \in \mathrm{K} \div A \iff B \in \mathrm{K}$ и $B \in Cn(\neg A)$

Доказательство:

Доказательство предоставляется любознательному читателю в качестве упражнения. ■

Иными словами, результатом сокращения наших убеждений в соответствии с определением 5 будет лишь множество тех убеждений, которые логически следуют из одного только высказывания $\neg A$! Ясно, что такое сокращение не может быть признано удачным.

Другое возможное предложение заключается в том, чтобы выбрать из множества $\mathrm{K}\bot A$ один элемент и рассмотреть его как результат применения операции сокращения. Это значит, что мы вводим на множестве $\mathrm{K}\bot A$ некоторую функцию выбора, скажем δ, которая выбирает из этого множества ровно один элемент – $\delta(\mathrm{K}\bot A)$ для каждого A. Тогда имеем следующее определение, посредством которого вводится так называемое "сокращение максимального выбора":

Определение 6.
$\mathrm{K} \div A = \delta(\mathrm{K}\bot A)$

Сокращение максимального выбора также имеет существенный недостаток – оно не оставляет возможности действовать достаточно осторожно. Так, если мы находимся перед выбором – удалить либо высказывание A, либо – высказывание B и при этом не имеем абсолютно никаких резонов предпочесть одно из этих высказываний, может оказаться полезным отбросить оба эти высказывания, чтобы быть полностью уверенным в наших убеждениях. Например, пусть мы полагали, что госпожа Иванова имеет ровно два ребенка – мальчика и девочку, а затем узнали, что на самом деле ребенок у Ивановой только один, при этом о поле ребенка ничего не было сказано. Естественно, мы не можем сохранить оба имевшиеся ранее у нас убеждения "Иванова имеет мальчика" и "Иванова имеет девочку". И хотя "объективно" одно из этих высказываний является истинным, но, поскольку мы не получили достаточно точной информации, будет разумным отбросить (по крайней мере пока, до получения необходимых уточняющих данных) *оба* эти убеждения и признать, что мы не уверены ни в том, что госпожа Иванова имеет мальчика, ни в том, что она имеет девочку. Такого рода стратегия к сожалению оказывается невозможной в рамках определения 6.

Более разумным представляется следующий путь. Мы вводим на множестве $\mathrm{K}\bot A$ некоторую *функцию предпочтения*, скажем γ, которая отбирает те элементы этого множества, которые являются более "предпочтительными", более "достойны сохранения", по сравнению с остальными множествами убеждений. Результатом сокращения будет тогда пересечение всех элементов из $\gamma(\mathrm{K}\bot A)$. Это есть "сокращение частичного пересечения".

Определение 7.
Если $\mathrm{K}\bot A$ непусто, то $\gamma(\mathrm{K}\bot A) \subseteq \mathrm{K}\bot A$ и $\mathrm{K}\bot A$ также непусто;
Если $\mathrm{K}\bot A$ пусто, то $\gamma(\mathrm{K}\bot A) = \{\mathrm{K}\}$.

Определение 8.

$$\mathrm{K} \div A = \cap\gamma(\mathrm{K}\perp A)$$

Нетрудно видеть, что сокращения полного пересечения и максимального выбора суть частные случаи сокращения частичного пересечения. Оказывается также, что свойства данной операции могут быть охарактеризованы посредством некоторого набора постулатов, которые должны для нее выполняться. Иными словами, операция частичного сокращения допускает построение определенной аксиоматической теории. Опишем кратко эти постулаты:

1. "*Постулат замыкания*" (closure): если K является системой убеждений, то $\mathrm{K} \div A$ также есть система убеждений.
   (Иными словами, $\mathrm{K} \div A$ должно быть замкнуто по отношению логического следования, если таковым является само K.)
2. "*Постулат успеха*" (success): если $A \notin Cn(\varnothing)$, то $A \notin \mathrm{K} \div A$.
   (Успех сокращения очевидно заключается в том, что удаляемое высказывание не должно принадлежать результирующей системе убеждений. Однако, сокращение не может быть успешным, если мы попытаемся удалить из наших убеждений логически истинное высказывание (то есть закон логики). Тот факт, что высказывание A является логической теоремой можно обозначить посредством $A \notin Cn(\varnothing)$, поэтому постулат успеха имеет в качестве условия требование, что A не является теоремой логики.)
3. "*Постулат включения*" (inclusion): $\mathrm{K} \div A \subseteq \mathrm{K}$.
   (Получившаяся в результате сокращения система убеждений должна составлять подмножество исходной системы убеждений.)
4. "*Постулат пустоты*" (vacuity): если $A \notin \mathrm{K}$, то $\mathrm{K} \div A = \mathrm{K}$.
   (Если мы попытаемся "удалить" из нашей системы убеждений высказывание, которое в действительности вовсе не принадлежит этой системе, то наша система убеждений просто останется без изменения – никакого сокращения не произойдет.)
5. "Постулат восстановления" (recovery): $\mathrm{K} \subseteq (\mathrm{K} \div A) + A$.
   (В соответствии с этим постулатом, *все* наши знания должны быть восстановлены, если мы вначале сократим систему убеждений посредством высказывания A, а затем возвратим A в нашу систему убеждений.)
6. "*Постулат экстенсиональности*" (extensionality): если $A \Leftrightarrow \mathrm{B} \subseteq Cn(\varnothing)$, то $\mathrm{K} \div A = \mathrm{K} \div \mathrm{B}$.

И в заключение может быть сформулирована важная репрезентационная теорема:

Теорема 1. (Алчуррон, Герденфорс, Макинсон)
Операция "÷" есть сокращение частичного пересечения, если и только если для него выполняются постулаты 1 – 6.

Доказательство:
Доказательство не представляет особенных затруднений.
■

References

[1] Alchourrón, C.E., Gärdenfors, P. and Makinson, D., 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2), pp.510–530. Available from: http://www.jstor.org/stable/2274239.

[2] Gärdenfors, P., 1988. *Knowledge in flux. Modeling the dynamics of epistemic states*. Cambridge, Mass., and London: The MIT Press.

[3] Rott, H., 1991. Two methods of constructing contractions and revisions of knowledge systems. *Journal of Philosophical Logic*, 20(2), pp.149–173. Available from: https://doi.org/10.1007/BF00284973.

Physics models in the course “The basics of computer simulation”

Illia O. Teplytskyi[1,2]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*
[2]*Kryvyi Rih Tsentralno-Miska gymnasium of the Kryvyi Rih City Council in Dnipropetrovsk oblast, 16 Pershotravneva Str., Kryvyi Rih, 50000, Ukraine*

Abstract. A review of the methodological literature on computer modeling shows the existence of different approaches to its teaching in secondary and higher education. It is a common approach in which the construction of models is carried out using the apparatus of higher mathematics, which is possessed mainly by senior students. This leads to the transfer of the course “Methods of mathematical modeling” for 7-8, and sometimes 9-10 semesters, which reduces its role in shaping the worldview of the future specialist, which takes place in high school and junior high school. This state of affairs forced us to create a propaedeutic course “The basics of computer simulation”, which was developed by the joint efforts of the Department of Informatics and Applied Mathematics of Kryvyi Rih State Pedagogical University and the Department of Informatics of the Kryvyi Rih Tsentralno-Miska gymnasium. The methodological support of the course is a textbook designed for high school students and junior high school students.

Keywords: physics models, computer simulation

Огляд методичної літератури з комп’ютерного моделювання показує існування різних підходів до його викладання у середній та вищій школі. Загальноприйнятим є підхід, при якому побудова моделей здійснюється з використанням апарату вищої математики, яким володіють переважно студенти старших курсів. Це зумовлює перенесення курсу “Методи математичного моделювання» на 7-8, а іноді і на 9-10 семестри, що знижує його роль у формуванні світогляду майбутнього спеціаліста, яке відбувається у старших класах школи та на молодших курсах вузу.

Такий стан справ змусив вдатися до створення пропедевтичного курсу “Основи комп’ютерного моделювання”, який було розроблено спільними зусиллями кафедри інформатики та прикладної математики Криворізького педуніверситету та кафедри інформатики Центрально-Міської гімназії м. Кривого Рогу. Методичним забезпеченням курсу є навчальний посібник, призначений для учнів старших класів школи та студентів молодших курсів вузу. Концепція курсу та результати його апробації було викладено у працях [2, 3].

При створенні посібника однією з основних задач був підбір змістовних моделей, побудова яких не вимагає від учнів знань, що виходять за межі шкільної програми. Основна увага при вивченні моделей приділена технології моделювання. Кожна модель

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

hs@mg.dp.ua (I. O. Teplytskyi)

ACNS Conference Series: Social Sciences and Humanities

передбачає не менше трьох версій, що поступово ускладнюються. Робота починається з аналізу досліджуваного предмета (явища), виділення його суттєвих (з точки зору дослідника) якостей, що призводить до висунення певних спрощуючих припущень. Далі, на основі висунутих припущень, відбувається формалізація моделі – запис сформульованих залежностей у вигляді математичних співвідношень. Обчислювальний експеримент – один з основних етапів роботи з побудованою моделлю, який дозволяє, змінюючи її параметри, отримати уявлення про поведінку досліджуваного об'єкта у різних ситуаціях та зробити певні висновки щодо її адекватності.

Суттєвим є питання про вибір середовища для моделювання, яке повинне давати можливість відстежувати результати моделювання як у числовій (у вигляді часових рядів), так і у графічній формі (у вигляді графіків залежностей обчислюваних величин). Традиційно таким середовищем є програми, написані однією з мов високого рівня, проте для пропедевтичного курсу, яким є наш, цілком прийнятним середовищем є електронні таблиці, які дозволяють, не відволікаючись на інтерфейс користувача, сконцентруватися безпосередньо на роботі з моделлю.

Чильне місце в обговорюваному курсі посідають динамічні моделі з шкільного курсу фізики, що описуються законами Ньютона. Відомо, що велику кількість динамічних задач засобами елементарної математики аналітично розв'язати неможливо. Проте чисельне розв'язання – методом скінченних різниць – для школярів цілком доступне і не викликає утруднень. Р. Фейнман, піонер використання такого підходу для роботи з молодшими студентами, на своїх лекціях використовував саме таблиці, які «є, звичайно, просто зручною формою запису результатів, отриманих з рівнянь, і фактично повністю замінюють їх» [1, p. 170]. Однією з задач, яку він пропонував студентам, був рух планети навколо центрального тіла. Розглянемо її реалізацію у електронних таблицях.

Нехай наша система складається з чотирьох тіл з масами m_0, m_1, m_2 та m_3 відповідно (figure 1).

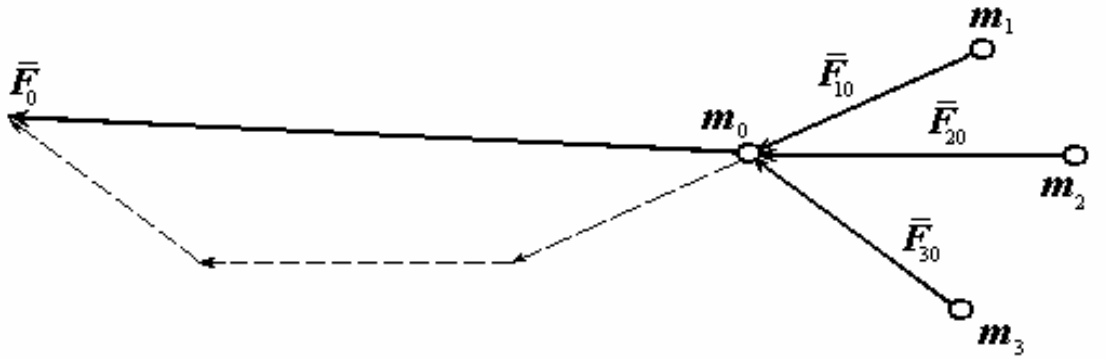


Figure 1: Система чотирьох тіл.

Згідно закону всесвітнього тяжіння, сила, що діє на тіло з масою m_i з боку всіх інших,

дорівнює векторній сумі парних взаємодій:

$$\boldsymbol{F}_i = \sum_{\substack{j \\ (i \neq j)}} \boldsymbol{F}_{ji} = \sum_{\substack{j \\ (i \neq j)}} G\frac{m_j m_i}{r_{ji}^3}\boldsymbol{r}_{ji} = Gm_i \sum_{\substack{j \\ (i \neq j)}} \frac{m_j}{r_{ji}^3}\boldsymbol{r}_{ji} \tag{1}$$

Ця сила, згідно другого закона Ньютона, надає тілу прискорення:

$$\boldsymbol{F}_i = m_i \boldsymbol{a}_i \tag{2}$$

Прирівнюючи формули (1) та (2), отримуємо формулу для визначення прискорення:

$$m_i \boldsymbol{a}_i = Gm_i \sum_{\substack{j \\ (i \neq j)}} \frac{m_j}{r_{ji}^3}\boldsymbol{r}_{ji} \Rightarrow \boldsymbol{a}_i = G \sum_{\substack{j \\ (i \neq j)}} \frac{m_j}{r_{ji}^3}\boldsymbol{r}_{ji} \tag{3}$$

Зауважимо, що у нашому випадку ми, взагалі кажучи, не можемо користуватися класичними формулами для визначення швидкості та координати, бо, згідно (3), прискорення залежить від координати, тобто *рух тіла під дією сили тяжіння не є рівноприскореним.* Як можна подолати цю перешкоду? Скористаємося чисельним методом – розіб'ємо весь час руху тіла на дуже малі проміжки і будемо вважати, що на кожному з цих елементарних проміжків прискорення є постійним.

Нехай на початку руху i-те тіло має координати (x_{i0}, y_{i0}), прискорення $\boldsymbol{a}_{i0}$ та швидкість $\boldsymbol{v}_{i0}$. Наприкінці першого проміжку часу тіло набуде прискорення $\boldsymbol{a}_{i1}$ за (3); його швидкість обчислюватиметься за формулою:

$$\boldsymbol{v}_{i1} = \boldsymbol{v}_{i0} + \boldsymbol{a}_{i0}\Delta t \tag{4}$$

Знаючи швидкість, ми можемо обчислити нові координати тіла:

$$\begin{aligned} x_{i1} &= x_{i0} + v_{i1x}\Delta t \\ y_{i1} &= y_{i0} + v_{i1y}\Delta t \end{aligned} \tag{5}$$

Змінюючи i, визначаємо прискорення, координати та швидкості всіх інших тіл наприкінці першого проміжку часу. Повторюючи цю процедуру, ми врешті-решт одержимо їх координати, дискретизовані проміжком часу Δt, що дасть нам змогу побудувати графіки їх руху. Для тестування візьмемо спочатку лише два тіла, а далі вдосконалюватимемо нашу модель, поступово вводячи до розгляду інші тіла.

Отже, ми можемо записати остаточний

АЛГОРИТМ РОБОТИ З МОДЕЛЛЮ:

1. Створимо електронну таблицю за таким зразком:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	a_{1x}	a_{1y}	a_{2x}	a_{2y}	υ_{1x}	υ_{1y}	υ_{2x}	υ_{2y}	x_1	y_1	x_2	y_2	***Дано:***	
2													G=	
3													Δt=	
4													m_1=	
5													m_2=	
6													υ_{1x0}=	
7													υ_{1y0}=	
8													υ_{2x0}=	
9													υ_{2y0}=	
10													x_{10}=	
11													y_{10}=	
12													x_{20}=	
13													y_{20}=	
14														

2. Занесемо у другий рядок початкові дані:

комірки	**формули / числа**
A2	=$N$2*$N$5*(K2-I2)/СТЕПЕНЬ(КОРЕНЬ((K2-I2)^2+(L2-J2)^2);3)
B2	=$N$2*$N$5*(L2-J2)/СТЕПЕНЬ(КОРЕНЬ((K2-I2)^2+(L2-J2)^2);3)
C2	=$N$2*$N$4*(I2-K2)/СТЕПЕНЬ(КОРЕНЬ((K2-I2)^2+(L2-J2)^2);3)
D2	=$N$2*$N$4*(J2-L2)/СТЕПЕНЬ(КОРЕНЬ((K2-I2)^2+(L2-J2)^2);3)
E2	=$N$6
F2	=$N$7
G2	=$N$8
H2	=$N$9
I2	=$N$10
J2	=$N$11
K2	=$N$12
L2	=$N$13

В комірки E2-L2 ми переносимо значення із стовбчика “Дано”, а в A2-D2 заносимо формули для обчислення проєкцій прискорення тіл згідно (3).

3. У третій рядок в комірки A3-D3 скопіюємо вміст комірок A2-D2, а комірки E3-L3 модифікуємо відповідно до формул (4) та (5):

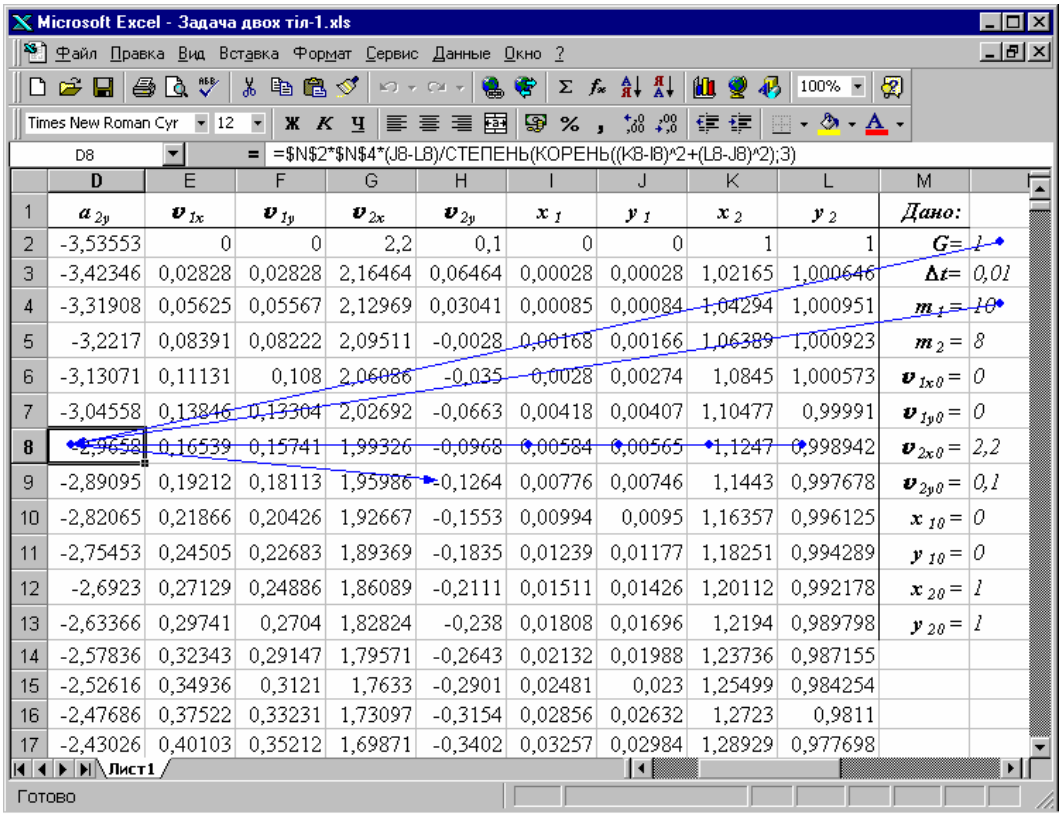
комірки	**формули / числа**
E3	=E2+A2*$N$3
F3	=F2+B2*$N$3
G3	=G2+C2*$N$3
H3	=H2+D2*$N$3
I3	=I2+E3*$N$3
J3	=J2+F3*$N$3
K3	=K2+G3*$N$3
L3	=L2+H3*$N$3

4. Скопіювати третій рядок у наступні рядки (їх кількість добиратимемо експериментально).

Далі ми можемо розпочати

ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ:

Введемо такі початкові дані: $G = 1$; $\Delta t = 0{,}01$; $m_1 = 10$; $m_2 = 8$; $v_{1x0} = 0$; $v_{1y0} = 0$; $v_{2x0} = 2{,}2$; $v_{2y0} = 0{,}1$; $x_{10} = 0$; $y_{10} = 0$; $x_{20} = 1$; $y_{20} = 1$ (figure 2). Легко побачити, що їх вибір нічим не зумовлений, проте для тестування моделі цього цілком достатньо. Зауважимо, що при виборі слід звернути увагу на час дискретизації – він не повинен бути занадто великим (при цьому алгоритм, що включає в себе різницеву схему, втратить стійкість – чим більше проміжок, тим менш достовірні результати), проте і занадто малим його робити не слід – нам буде заважати не лише величезна кількість рядків, які нам треба буде скопіювати, а й постійно зростаюча в операціях сумування похибка округлення.



	D	E	F	G	H	I	J	K	L	M	
1	a_{2y}	v_{1x}	v_{1y}	v_{2x}	v_{2y}	x_1	y_1	x_2	y_2	Дано:	
2	-3,53553	0	0	2,2	0,1	0	0	1	1	G=	1
3	-3,42346	0,02828	0,02828	2,16464	0,06464	0,00028	0,00028	1,02165	1,000646	Δt=	0,01
4	-3,31908	0,05625	0,05567	2,12969	0,03041	0,00085	0,00084	1,04294	1,000951	m_1 =	10
5	-3,2217	0,08391	0,08222	2,09511	-0,0028	0,00168	0,00166	1,06389	1,000923	m_2 =	8
6	-3,13071	0,11131	0,108	2,06086	-0,035	0,0028	0,00274	1,0845	1,000573	v_{1x0} =	0
7	-3,04558	0,13846	0,13304	2,02692	-0,0663	0,00418	0,00407	1,10477	0,99991	v_{1y0} =	0
8	-2,9652	0,16539	0,15741	1,99326	-0,0968	0,00584	0,00565	1,1247	0,998942	v_{2x0} =	2,2
9	-2,89095	0,19212	0,18113	1,95986	-0,1264	0,00776	0,00746	1,1443	0,997678	v_{2y0} =	0,1
10	-2,82065	0,21866	0,20426	1,92667	-0,1553	0,00994	0,0095	1,16357	0,996125	x_{10} =	0
11	-2,75453	0,24505	0,22683	1,89369	-0,1835	0,01239	0,01177	1,18251	0,994289	y_{10} =	0
12	-2,6923	0,27129	0,24886	1,86089	-0,2111	0,01511	0,01426	1,20112	0,992178	x_{20} =	1
13	-2,63366	0,29741	0,2704	1,82824	-0,238	0,01808	0,01696	1,2194	0,989798	y_{20} =	1
14	-2,57836	0,32343	0,29147	1,79571	-0,2643	0,02132	0,01988	1,23736	0,987155		
15	-2,52616	0,34936	0,3121	1,7633	-0,2901	0,02481	0,023	1,25499	0,984254		
16	-2,47686	0,37522	0,33231	1,73097	-0,3154	0,02856	0,02632	1,2723	0,9811		
17	-2,43026	0,40103	0,35212	1,69871	-0,3402	0,03257	0,02984	1,28929	0,977698		

Figure 2: Обчислювальний експеримент.

На figure 3 – графік руху тіл, отриманий при введених початкових даних.
Проведемо

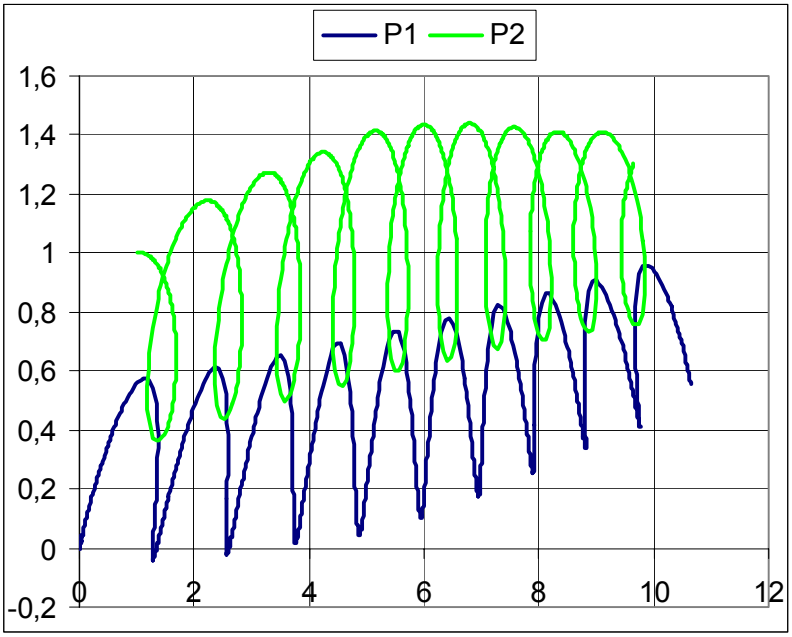


Figure 3: Графік руху тіл за уведених початкових даних.

АНАЛІЗ РЕЗУЛЬТАТІВ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ:

З figure 3 видно, що тіла дійсно рухаються згідно закону всесвітнього тяжіння: хоча траєкторія їх спільного руху і звивиста, проте можна побачити, що у моменти зближення прискорення, відповідно до закону всесвітнього тяжіння, збільшується, що, у свою чергу, призводить до різкого збільшення швидкості. Тіла "розлітаються", але із збільшенням відстані прискорення зменьшується до 0, аж поки не змінює свій знак. Цей факт означає, що тіла повинні знову зближуватися і т.д.

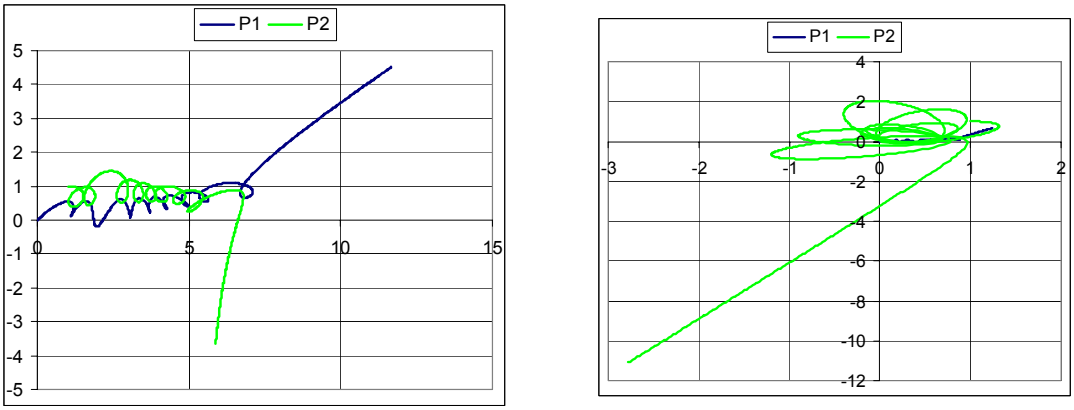
Поставимо питання:

1. А що буде, якщо тіла занадто зблизяться? Якщо їх маси співрозмірні, то вони на великій швидкості віддалятимуться одне від одного у нескінченність (figure 4a), якщо ж ми маємо справу з системою планета-супутник, то супутник передасть свій імпульс планеті і вирветься з її «гравітаційних обіймів» (figure 4b).
2. Які початкові умови треба задати, щоб тіла рухались у одному напрямку? Для відповіді на це питання скористаємося законом збереження імпульсу: задамо початкові дані такими, щоб виконувалося, наприклад, співвідношення:

$$m_1 v_{1x0} = m_2 v_{2x0}.$$

Результати підтверджують нашу здогадку (figure 5).

Незважаючи на довільність вихідних даних, під час тестування ми впевнилися у *якісній відповідності* нашої моделі припущенням, покладеним у її основу (якими у нашому



(a) (b)

Figure 4: Моделювання зближення тіл.

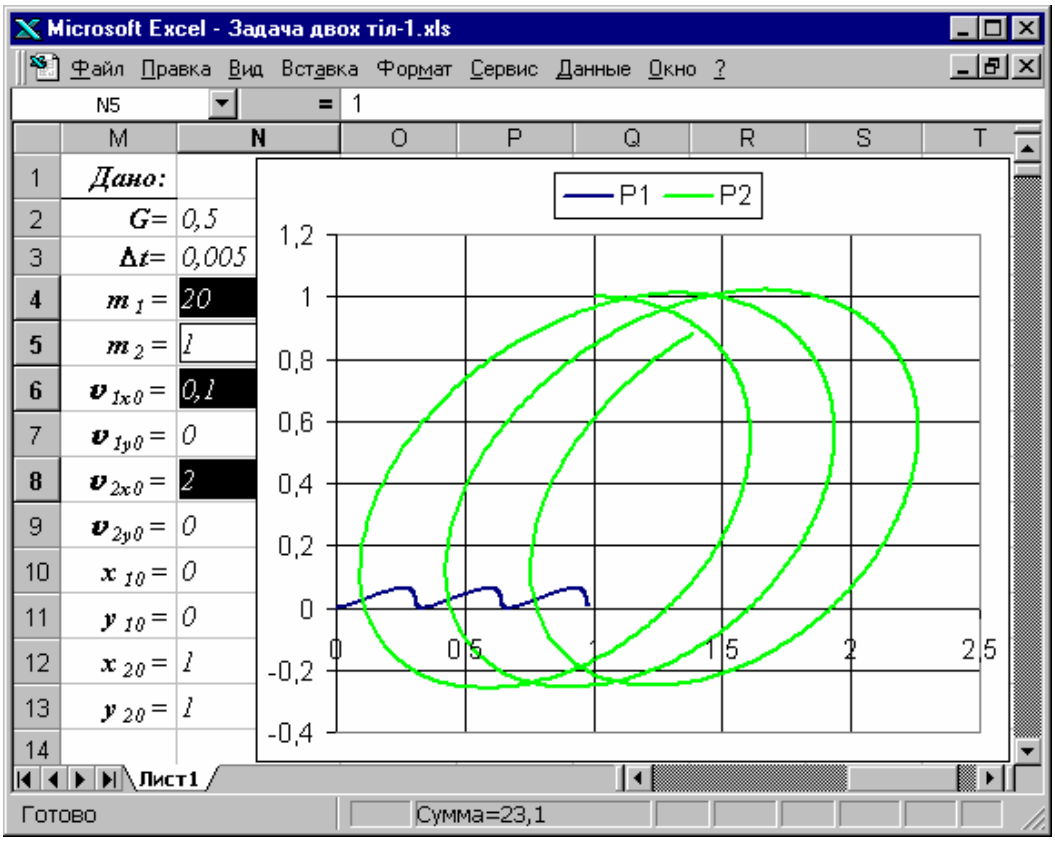


Figure 5: Моделювання руху тіл в одному напрямі.

випадку виступають закони динаміки). Завдяки використанню зручного середовища для моделювання, у обчислювальному експерименті ми з легкістю змінювали параметри моделі, миттєво отримуючи результати. І, нарешті, наявність зручного засобу візуалізації результатів моделювання дозволило нам провести аналіз експерименту і скорегувати початкові дані у відповідності до тих результатів, які ми хотіли отримати.

References

[1] Feynman, R.P., Leighton, R.B. and Sands, M.L., 1989. *The Feynman lectures on physics*, vol. I, Mainly mechanics, radiation, and heat. Redwood City, Calif.: Addison-Wesley. Available from: https://www.feynmanlectures.caltech.edu/I_toc.html.

[2] Soloviev, V.N., Semerikov, S.O. and Teplytskyi, I.O., 1998. Osnovy kompiuternoho modeliuvannia v serednii shkoli ta pedahohichnomu vuzi [Basics of computer simulation in high school and pedagogical university]. *Doprofesiina pidhotovka uchnivskoi molodi v konteksti realizatsii tsilovoi kompleksnoi prohramy "Vchytel"*. Dnipropetrovsk, vol. 2, pp.53–56. Available from: http://elibrary.kdpu.edu.ua/handle/0564/683/.

[3] Teplytskyi, I.O. and Semerikov, S.O., 1998. Vyvchennia fraktalnykh klasteriv za dopomohoiu imitatsiinykh kompiuternykh modelei [Study of fractal clusters using simulation computer models]. *Zbirnyk naukovykh prats Skhidnoukrainskoho derzhavnoho universytetu. Seriia "Mashynobuduvannia"*, pp.276–289. Available from: http://elibrary.kdpu.edu.ua/handle/0564/695.

Concept of the course “Numerical methods in object methodology”

Aleksandr P. Polishchuk[1], *Sergei A.* Semerikov[1]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

Abstract. The tasks for which computers were created – routine calculations of an industrial, scientific and military nature – required the creation of a whole class of new methods focused not on manual but on machine calculations. The first programming languages did not have convenient means for reflecting such objects often used in computational mathematics as matrices, vectors, polynomials, etc. Further development of programming languages followed the path of embedding mathematical objects into languages as data types, which led to their complication. So, for example, an attempt to make a universal language Ada, in which there are even such data types as dictionaries and queues, led to the fact that the number of keywords in it exceeded 350, making it almost unusable for learning and use. The compromise solution between these two extremes can be the following: let the programmer himself create the data types that he needs in his professional work. Programming languages that implement this approach are called object-oriented. This, on the one hand, makes it possible to make the language quite easy by reducing the number of keywords, and on the other, expandable, adapting to specific tasks by introducing keywords for creating and using new data types.

Keywords: C++, numerical methods, mathematical classes

Задачи, ради которых и были созданы компьютеры – рутинные расчёты производственного, научного и военного характера, – потребовали создания целого класса новых методов, ориентированных не на ручные, а на машинные вычисления. Первые языки программирования не обладали удобными средствами для отражения таких часто используемых в вычислительной математике объектов, как матрицы, вектора, полиномы и т.д. Дальнейшее развитие языков программирования шло по пути встраивания математических объектов в языки как типов данных, что вело к их усложнению. Так, например, попытка сделать универсальный язык Ада, в котором есть даже такие типы данных, как словари и очереди, привела к тому, что количество ключевых слов в нём превысило 350, сделав его практически непригодным для изучения и использования.

Компромиссным решением между этими двумя крайностями может быть следующее: пусть программист сам создаёт типы данных, которые ему необходимы в его профессиональной деятельности. Языки программирования, в которых реализован такой подход, называют объектно-ориентированными. Это, с одной стороны, позволяет сделать язык достаточно лёгким путём уменьшения количества ключевых слов, а

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

apol@cabletv.dp.ua (A. P. Polishchuk); cc@kpi.dp.ua (S. A. Semerikov)

https://kdpu.edu.ua/semerikov (S. A. Semerikov)

ACNS Conference Series: Social Sciences and Humanities

с другой – расширяемым, приспосабливаемым к конкретным задачам введением ключевых слов для создания и использования новых типов данных.

Наиболее удобным инструментом для создания классов математических объектов является объектно-ориентированное программирование и его поддержка в языке C++. Этот язык дает возможность варьировать методы создания математических объектов путем определения в классе необходимого количества конструкторов, осуществить переопределение стандартных операций для вновь созданных классов, использовать мощный механизм одиночного и множественного наследования свойств базовых классов в производных классах, создавать параметризованные классы и функции с подстановкой типов параметров в процессе конструирования соответствующих объектов. Последовательное наращивание иерархии математических типов на базе уже созданных позволяет существенно снизить трудоемкость программирования за счет исключения повторяющихся последовательностей действий и избежать внесения в программы новых ошибок.

Одной из самых важных учебно-методических линий в курсе алгебры средней школы является линия уравнений и неравенств, а также их систем. В высшей школе эта линия продолжается на более высоком уровне в курсе линейной алгебры и векторных пространств, что позволило нам выделить следующие математические объекты: вектора, матрицы и многочлены. Наш выбор обуславливался ещё тем, что все они тесно связаны друг с другом. Так, например, матрицу можно рассматривать как упорядоченный кортеж арифметических векторов, а многочлен можно задать вектором его коэффициентов.

Для каждого из этих типов в процессе написания работы понадобилось определить множество процедур, в большинстве своём – бинарных алгебраических операций над элементами соответствующих множеств. Результатом работы явилось создание библиотеки классов параметризованных векторных, матричных и полиномиальных объектов, расширяющих возможности языка C++ по работе с такими объектами. Рассмотрим детальнее механизм реализации этой библиотеки.

Базовым в нашей иерархии классов является класс для работы с арифметическими векторами. В нашей интерпретации вектор – это упорядоченный кортеж некоторых объектов (или, иными словами, массив определённой длины). Тип чисел, составляющих компоненты вектора, не является строго фиксированным – этот тип можно использовать как параметр конструктора векторного объекта, то есть вы можете иметь целочисленные, вещественные, комплексные и т.д. вектора различных длин. Основные операции, определённые для векторов – это сложение, вычитание, отрицание, скалярное умножение, умножение вектора на скаляр, сравнение векторов, вывод вектора в поток и ввод его из потока, нахождение модуля вектора и нормирование его по модулю, а также ряд операций сокращённого сложения, вычитания и т.д. Разумеется, большинство этих операций определены только для векторов совпадающих размерностей.

Для реализации арифметических операций был использован механизм перегрузки операций, благодаря которому, например, при сложении двух векторов в программе достаточно поставить знак “+” между объектами векторного класса точно так же, как это делается при сложении стандартных целых и вещественных чисел. Такая запись является более естественной, чем вызов функции Add, хотя, по сути, ничем от неё не отличается. Это свойство является особенностью именно C++, в отличие от других

объектно-ориентированных языков.

Следующим классом, разработанным нами, был класс, предназначенный для хранения и использования, пожалуй, самых важных алгебраических объектов – матриц. Структура матричного класса подобна структуре векторного, но более содержательная. Это обусловлено множеством операций, которые необходимо было запрограммировать для эффективной работы с новым типом данных – "матрица". Как и векторный класс, матричный богат конструкторами, которые позволяют создавать матрицы, инициализируя их данными из памяти, из файла, из другой матрицы и так далее. Кроме того, допустимо создание «пустой», т.е. нулевой матрицы заданного размера.

В качестве внутреннего представления матрицы был выбран упорядоченный кортеж векторов – объектов векторного класса, рассмотренного нами ранее, что позволяет использовать для матричных операций перегруженные операции класса "вектор". Например, для сложения двух матриц совпадающих размерностей нам достаточно сложить вектора, их составляющие, что позволяет вместо двух циклов сложения элементов матриц использовать один цикл сложения векторов, составляющих строки матрицы.

Для матриц совпадающей размерности определены операции сложения, вычитания, а также их сокращённые аналоги; умножение матрицы на скаляр, транспонирование, сравнение матриц, вывод в поток и ввод из потока. Для отдельных типов матриц определена операция умножения.

Практический интерес при работе с матрицами представляет решение систем линейных алгебраических уравнений, задаваемых соответственно матрицами коэффициентов при неизвестных и вектор-столбцом свободных членов. В связи с этим были реализованы следующие методы решения СЛАУ: метод Гаусса с выбором главного элемента, метод ортогонализации векторов матрицы, метод обратной матрицы, метод Крамера.

Вычисление обратной матрицы и детерминанта можно производить либо аналитически, либо численно. Нами были реализованы оба этих подхода. Так, Вы можете воспользоваться рекурсивной функцией вычисления детерминанта разложением его по какой-либо строке и функцией обращения матрицы с использованием алгебраических дополнений. При этом с ростом порядка матрицы количество операций сложения и умножения возрастает настолько, что эти методы стают малоэффективными. Более эффективным является вычисление детерминанта и обратной матрицы косвенно, путём решения системы уравнений.

Перегрузка перечисленных операций даёт нам возможность записи операций над матрицами в близкой к алгебраической форме. К примеру, решение системы уравнений может быть записано как решение матричного уравнения $\mathbf{AX} = \mathbf{B}$, $\mathbf{X} = \mathbf{A^{-1}} * \mathbf{B}$, где операция возведения в степень "-1" не что иное, как перегруженная функция обращения матрицы, а "$*$" – операция умножения матриц.

Наличие, наряду с умножением и обращением, операции транспонирования, позволяет нам одной строчкой программы записать решение задачи МНК – метода наименьших квадратов. Пусть $\mathbf{X}$ и $\mathbf{Y}$ – соответственно матрицы независимых и зависимых переменных, $\mathbf{A}$ – неизвестный вектор оценки МНК. Тогда $\mathbf{A} = X$ транспонированное, умноженное на $\mathbf{X}$ (всё в минус первой степени), умноженное

на произведение транспонированной матрицы независимых переменных на вектор-столбец зависимых переменных:

*matrix a=(&(x*x))*(x*y);*

Класс для работы с многочленами от одной переменной базируется на векторе – действительно, операции над многочленами сводятся к действиям над коэффициентами при соответствующих степенях неизвестной. Это даёт возможность использовать арифметический вектор для представления многочлена. Как и в предыдущих классах, полиномиальные объекты имеют набор методов для конструирования, сложения, вычитания, умножения полинома на полином и полинома на скаляр, сравнения, деления с остатком и т.п.

Для полиномиальных объектов мы можем найти производную любого порядка и неопределённый интеграл любой кратности в аналитической форме; результатом будет полином, коэффициенты которого получаются по соответствующим правилам. Кроме того, в любой точке мы можем найти функциональное значение полинома.

В задачах прикладной математики полиномиальные объекты, как правило, используются при решении алгебраических уравнений, что побудило нас к реализации ряда методов решения уравнений разного порядка. Согласно основной теореме алгебры, полином n-ной степени имеет ровно n корней. Это позволяет считать, что решением полиномиального уравнения в комплексной области является комплексный вектор решений с размерностью, равной степени многочлена, компонентами которого являются его корни.

Для многочленов с действительными коэффициентами нами были рассмотрены метод Кардано-Тартальи для решения уравнений 3-ей степени и базирующийся на нём метод Феррари решения уравнений 4-ой степени. Более универсальными методами для комплексных многочленов являются методы решения квадратных уравнений и метод Ньютона для поиска корней многочлена любой степени. В последнем методе мы, находя очередной корень, отделяем его, понижая степень многочлена по схеме Горнера, ищем корень многочлена меньшей степени и т.д., до отделения всех корней.

Алгоритмы и программы решения проблемы собственных значений для несимметричных комплексных матриц, реализованные в традиционной методологии, всегда отличались громоздкостью. Для её решения используются все разработанные нами классы – вектора, матрицы и полиномы, существенно сокращая объём программы и повышая её наглядность.

В стройном здании математики более сложные математические объекты строятся из более простых. Так, комплексные числа представляются в виде пары вещественных координат вектора по вещественной и мнимой осям комплексной плоскости. Многомерный числовой вектор может быть представлен совокупностью его вещественных или комплексных координат по осям многомерной координатной системы (или как частный случай матрицы – одностолбцовой или однострочной), матрица может быть представлена как вектор векторов, полином известного порядка может быть задан как вектор его коэффициентов.

Для каждого класса математических объектов определен допустимый набор математических операций и способы их реализации; например, операции умножения

определены для вещественных чисел, векторов и матриц, но имеют, естественно, различный смысл и алгоритмы реализации. Операция определения нулей специфична для полиномов, транспонирования и вычисления собственных значений и собственных векторов – для матриц, определения модуля – для векторов.

Представляется целесообразным построить курс вычислительных методов по принципу определения иерархии математических классов, объекты которых конструировались бы затем в программе путем объявления, то есть синтаксически так же, как и стандартные для используемого языка типы (целые, вещественные и пр.) с определением внутри класса всех необходимых для их использования в вычислениях операций. При этом под термином “операция” можно понимать как общепринятые для простых типов операции, например, арифметические, так и любые, базирующиеся на данном математическом классе вычисления, – например, решение системы линейных алгебраических уравнений или вычисление коэффициентов регрессии для заданной матрицы или вычисление корней полинома наряду с операциями полиномиальной арифметики – сложения, умножения, деления полиномов.

Нам неизвестны пособия с систематическим изложением методов вычислений на базе объектно-ориентированного подхода в программной реализации на языке C++. Уже сейчас во многих средних и высших учебных заведениях осуществляется преподавание языка C и C++ и предлагаемая работа может, по нашему мнению, оказать положительное влияние на эффективность учебного процесса в области вычислительной математики.

Представляемое учебное пособие [1] рассчитано на сравнительно небольшой двухсеместровый курс численных методов (2 часа в неделю лекций + 2 часа лабораторных работ в компьютерном классе), который читается студентам специальности “Математика и информатика” педагогических институтов. Изложение курса предполагает владение основами объектно-ориентированного программирования на языке C++.

В соответствии с уже изложенной концепцией объектно-ориентированной программной реализации многие методы вычислений инкапсулируются в классы математических объектов, с которыми они работают; например, метод наименьших квадратов и метод решения систем линейных алгебраических уравнений будут размещены в классе матриц, а полиномиальная арифметика и методы вычисления полиномиальных нулей – в классе полиномов.

Глава 2 посвящена рассмотрению специальных математических типов (и операций над ними) и определению соответствующих им классов в терминах языка C++. Вначале в качестве иллюстрации рассматривается необходимый при изучении последующего материала (например, методов вычисления корней полиномов при наличии среди них комплексных) предположительно знакомый слушателю и реализованный в библиотеке C++ класс комплексных чисел; его программная реализация взята прямо из среды разработки Borland C++ и по возможности откомментирована – этот материал служит своеобразным образцом в реализации других рассмотренных в этой главе математических классов – векторов, полиномов, матриц. Изучение матричного класса сопровождается изложением методов решения основных задач линейной алгебры – систем линейных уравнений, вычисления собственных значений и векторов матриц.

Глава 3 содержит изложение методов полиномиальной и экспоненциальной аппроксимации функций и их программную реализацию, также инкапсулированную в

виде функций-членов специального класса.

Глава 4 является естественным прикладным продолжением предыдущей и содержит методы численного интегрирования и дифференцирования функций с использованием рассмотренных методов приближения функций.

Глава 5 посвящена методам решения обыкновенных дифференциальных уравнений; при этом рассматриваются не только численные методы, а иллюстрируется применение приближенных численных методов при программной реализации аналитических решений. Например, при решении дифференциальных уравнений методами операционного исчисления возникает задача вычисления корней характеристических уравнений, которая может быть решена численно.

Глава 6 содержит введение в поисковые методы определения экстремумов функций при отсутствии и наличии шумов в определении значения функции и методы программирования соответствующих задач.

Апробация курса “Численные методы в объектной методологии” в Криворожском педуниверситете в течение последних лет свидетельствует о повышении качества усвоения учебного материала за счёт переключения внимания обучаемого с деталей программной реализации на сам метод благодаря приближению программной записи алгоритма к естественной математической и использованию таких типов данных, как векторы, матрицы и полиномы.

Наличие готовой библиотеки математических объектов существенно ускоряет процесс программной реализации метода, сокращая не только время, но и объём программы, делая её более “прозрачной” за счёт повышения уровня абстракции до операций над новыми типами данных (это проявляется, например, в использовании для умножения матриц вместо трёх вложенных циклов знака умножения). Параметризация программ позволяет порождать из шаблонов типов специализированные реализации, делая, к примеру, из параметризованной матрицы действительную, комплексную либо функциональную простой подстановкой типа (double, complex, function) в угловые скобки поле имени параметризованного объекта.

Применение этих типов позволило расширить традиционный курс численных методов разделами, обычно вызывающими трудности в программной реализации в процедурной идеологии (символическое исчисление, линейное и динамическое программирование), по-новому взглянуть на традиционные методы и расширить область их применения.

References

[1] Polishchuk, A.P. and Semerikov, S.A., 1999. *Metody vychislenii v klassakh iazyka C++ [Numerical methods in C++].* Krivoi Rog: Izdatelskii otdel KGPI. Available from: http://elibrary.kdpu.edu.ua/handle/0564/755.

Information technologies of teaching in a secondary school

Mykola I. Zadorozhnii[1]

[1]*Novoiulivska secondary school, 36 Shkilna Str., Novoiulivka, 53150, Ukraine*

Abstract. The conference participants are offered components of the educational and methodological complex in physics, which are obtained not by advanced pedagogical experience, ie by trial and error, but by detailed and complete analysis and calculation of structures and algorithms of the educational process in secondary school and its individual components. It is a guide for students, systematization of educational information, algorithm for solving physical problems, solving standard physical problems on a computer, diagnosing student performance and rating assessment of knowledge.

Keywords: teaching physics, information technologies, secondary school

Комп'ютери і вчитель

Існує загальноприйняте поняття про інформаційні технології навчання, як обов'язкове використання електронно-обчислювальних машин для виконання обчислень, демонстрації зображень, зберігання та обробки текстової інформації, електронних таблиць, баз даних.

Зараз створюються навчально-ігрові програми з використанням сучасних мультимедійних засобів, які дуже добре демонструють можливості сучасної комп'ютерної техніки та здібності програмістів. Ці програми моделюють навчальну діяльність вчителя та учнів, намагаючись замінити вчителя.

Але ніякі комп'ютери і програми не замінять безпосереднього спілкування вчителя і учнів, тому потрібні програми, які б не витісняли вчителя з навчально-виховного процесу, а доломогали йому виконувати технічну, малопродуктивну роботу, підвищуючи цим самим продуктивність навчальної діяльності вчителя та учнів.

Висока продуктивність інформаційних машин

Загальновідомо, що людський мозок за своєю складністю та можливостями на багато порядків перевищує складність та можливості інформаційних машин, в той же час

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

njuscool@ukr.net (M. I. Zadorozhnii)
https://sites.google.com/view/njuschool15/vciteli/zadoroznij-m-i (M. I. Zadorozhnii)

ACNS Conference Series: Social Sciences and Humanities

продуктивність машин при роботі з інформацією на скільки ж порядків вища, ніж людини. Чому?

Можна назвати кілька причин:

- машина працює тільки з певним чином впорядкованою інформацією;
- машина працює за чітко організованою програмою;
- програма забезпечує повноту всіх можливих варіантів дій;
- програма забезпечує однозначність виконуваних дій.

При цьому слід відзначити, що ці чотири умови високої продуктивності інформаційних машин забезпечують не самі машини, а люди, які створюють програми та користуються машинами.

Навчальна діяльність вчителів та учнів – це теж робота з інформацією, і якщо для цієї роботи створити такі ж умови, як і для машин, продуктивність роботи вчителя та результативність навчання учнів стане на порядок вищою.

Інформаційні технології без машин

Одним з перших створив умови для високопродуктивної навчальної діяльності вчителя і учнів Віктор Федорович Шаталов із Донецька більше 20 років тому [4]. Він дійсно користувався:

- впорядкованою навчальною інформацією у вигляді конспектів;
- у нього був чітко організований навчальний процес;
- він описав близько 600 прийомів і методів навчання;
- він міг прогнозувати результати навчання учнів.

Очевидно не випадково саме в Донецьку зараз проводиться десятирічний експеримент АПН України з впровадження системи модульного навчання, що одержав назву “Школа розвитку”, під керівництвом Фурмана А. В. та Калугіна О. І. [2]

У системі модульного навчання реалізована надзвичайно важлива і корисна ідея повного функціонального циклу навчального модуля, що дає можливість технологізувати навчально-розвиваючий процес.

Умови реалізації інформаційних технологій навчання

З моменту виникнення електронно-обчислювальних машин їх намагаються навчити робити те, що вміє людина. Зараз настав час людям вчитися у машин працювати з інформацією.

Отже, можна вважати, що інформаційні технології навчання це такі способи діяльності вчителя та учнів, які мають на меті зберігання, передачу, перетворення та використання навчальної інформації, при цьому забезпечують впорядкування, чітку організацію, повноту та однозначність цієї інформації.

Використання комп’ютерної техніки дає можливість покласти на машину виконання тих елементів інформаційних технологій, які комп’ютер виконує швидше і точніше, ніж людина.

Необхідність використання інформаційних технологій навчання

Кількість навчальної інформації з усіх навчальних предметів постійно зростає. Навчальний час обмежений фізіологічними можливостями учнів, і навіть зменшується через недостатне фінансування. Тому переважна частина учнів та значна частина вчителів просто не встигають відповідним чином опрацювати навчальний матеріал підручника і довести навчально-виховний процес до логічного завершення, тобто учні одержують фрагментарні, неповноцінні знання.

Якщо пересічному вчителеві та учням поряд з традиційним підручником дати впорядковану навчальну інформацію, це значно полегшить і прискорить їх навчальну роботу, дасть можливість учням одержати систематичні, завершені знання.

Фізика – це один із самих складних і насичених навчальною інформацією предметів. Поява нових підручників, введення обов'язкового екзамену з фізики в 10 класі, відсутність системи навчально-методичних посібників зумовлює необхідність підготовки та впровадження навчально-методичних посібників, описаних в [5].

Сучасний підручник – це навчально-методичний комплекс

З точки зору інформаційних технологій сучасний підручник повинен представляти собою навчально-методичний комплекс, що містить крім підручника всі друковані матеріали, необхідні для організації навчального процесу:

- програма з чітко поставленими навчальними завданнями та рівневою диференціацією навчального матеріалу;
- календарно-тематичний план вчителя, з чітко визначеною системою уроків;
- плани уроків вчителя, структура яких відповідає повному функціональному циклу начання;
- довідник для учнів з впорядкованою навчальною інформацією;
- робочий зошит для учнів, що містить запитання, вправи, задачі;
- зошит з фізичного експерименту, що містить лабораторні роботи, досліди, експериментальні задачі;
- матеріали для повторення та систематизації знань учнів;
- збірник задач для самостійних та контрольних робіт учнів;
- комплект таблиць та наочних посібників.

Сучасний підручник – це комп'ютерна база даних

Забезпечити повноту і єдність всіх елементів навчального процесу можна за допомогою комп'ютерної бази даних з відповідним програмним забезпеченням.

Ця база даних повинна містити всі елементи навчального матеріалу: теоретичні відомості, запитання, вправи, задачі, досліди, лабораторні роботи, експериментальні задачі, структурно-логічні схеми для повторення та систематизації знань, контрольні

роботи. Програмні засоби можуть формувати з єдиної бази даних всі необхідні посібники для учнів та вчителів.

Особливо слід відзначити можливість створення таких комп'ютерних програм, які складають тексти задач і розв'язують їх з повним записом розв'язку та визначають рівень складності задачі. Це дасть можливість забезпечити індивідуальність та ефективність навчання.

База даних – це максимально структурована система знань, тому програмні засоби можуть створити багато варіантів посібників різного рівня складності для кожного вчителя, кожного класу, кожного учня.

Довідник для учнів

Цей посібник містить всі поняття, властивості. Формули та їх застосування, які є у підручнику. При цьому збережено означення понять, формулювання властивостей та пояснення явищ та дослідів, які використовуються в підручнику. Окремо виділено групи понять, які характеризують фізичну величину. Також виділено формули та зразки виконання вправ, наведених у підручнику. Крім цього навчальний матеріал систематизовано за такими ознаками:

1) фізичні явища та досліди; (Яв)
2) поняття та величини; (Вел)
3) властивості, формули та закони; (Фор Зак)
4) приклади та застосування явищ, понять, властивостей (Зас)

та розділено на частини, які необхідно вивчати на одному уроці [1].

Тема 1. Фізика – наука про природу

10. Явище всесвітнього тяжіння та маса тіла. (Лабораторна робота 5) §13, 14

Яв 1. **ГРАВІТАЦІЯ** – це гравітаційна взаємодія тіл, наприклад притягання тіл до Землі.
Яв 2. **ЯВИЩЕ ВСЕСВІТНЬОГО ТЯЖІННЯ** – так ще називають гравітацію. Англійський вчений І. Ньютон у 1666 році висунув гіпотезу, що явище гравітації проявляється між усіма тілами у Всесвіті. Протягом двадцяти років він довів це.
Вел 3. **МАСА ТІЛА** – фізична величина, яка є мірою гравітаційної властивості тіла.
Вел 4. Масу **позначають** символом m (ем).
Вел 5. **Основна одиниця**: кілограм (1 кг)
Кілограм – це маса міжнародного еталона маси, знаходиться в м. Севрі поблизу Парижа (Франція)
Вел 6. **Похідні одиниці**: грам: 1 г = 0.001 кг, міліграм: 1 мг = 0.001 г = 1^{-6} кг, тона: 1 т = 1000 кг
Вел 7. **Вимірювальні прилади**: важільні терези, пружинні динамометри.
Зак 8# **ЗАКОН ЗБЕРЕЖЕННЯ МАСИ** – маса тіла дорівнює сумі мас його частин, тобто маса тіла дорівнює масі молекули, помноженій на кількість молекул.
Фор 9. **ФОРМУЛА МАСИ ТІЛА** – $m = m_0 \cdot N$

Зас 10. (ЛР) **ВАЖІЛЬНІ ТЕРЕЗИ** – вимірювальний прилад, призначений для вимірювання маси тіл способом порівняння з тілами відомої маси.
Зас 11. (ЛР) **КОРОМИСЛО** – стержень, який може вільно обертатись навколо своеі осі, розміщеної посередині нього.
Зас 12. (ЛР) **ШАЛЬКИ** – дві тарілки, підвішені до кінців коромисла. Якщо маси тіл, що лежать на шальках, одинакові, терези перебувають у рівновазі.
Зас 13. (ЛР) **НАБІР ГИР** – набір тіл відомої маси.

Повторення та задачі

В цьому посібнику найважливіші поняття, властивості систематизовано навколо основних питань цього курсу фізики, а кілька таких питань, логічно зв'язаних між собою, об'єднуються в навчальний модуль. Так навчальний матеріал першої теми курсу "Фізика – наука про природу" об'єднано в шість модулів:

Фізика	**Всесвіт**	**Речовина**
Що вивчає?	Нескінченна подільність Всесвіту	Молекули
Як вивчає?	Простір і час	Величини
Як обчислює?	Гравітація	Стани речовини
Астрономія		
Що вивчає?	**Фізичний експеримент**	**Величини та формули**
Явища	Вимірювальні прилади	Основні величини
Поняття	Способи вимірювань	Похідні величини
Величини	Експериментальні задачі	Фізичні константи
Властивості	Лабораторні роботи	Формули
Застосування		

Цей посібник використовується вчителем та учнями для систематизації та повторення знань, може служити планом пояснення або відповіді на найважливіші питання курсу фізики. Таблиця навчальних модулів також розміщується на змінному стенді у фізичному кабінеті. Крім цього посібник містить зразки розв'язування задач та котрольні завдання, в яких зібрано всі види вправ та задач підручника, об'єднаних у групи: експериментальні завдання, перетворення одиниць вимірювання фізичних величин та задачі з даної теми, об'єднані навколо певного фізичного явища.

МОДУЛЬ 2. ВСЕСВІТ

НЕСКІНЧЕННА ПОДІЛЬНІСТЬ ВСЕСВІТУ

…???
Космос = світ зірок та планет
Земля = частина Всесвіту
Речовина = складається з частинок
Молекула = найдрібніша частинка речовини

Атом	= складова частина молекул
Субатомні частинки	= складова частина атомів
…???	

ПРОСТІР ТА ЧАС

Протяжність	= міра лінійних розмірів
Площа	= міра поверхні
Об'єм	= міра простору
Час	= форма лослідовноі зміни станів
	= міра тривалості подій

ГРАВІТАЦІЯ

Явище всесвітнього тяжіння	= гравітація (або притягання тіл)
Маса	= міра гравітаційної взаємодії
Закон збереження маси	= маса тіла дорівнює масі його частин

Алгоритм розв'язування фізичних задач

Рівень 0	Рівень 1	Рівень 2	Рівень 3
I етап: Явища			
1. Аналіз явищ		2. Явище за графіком 3. Виконання малюнка	
II етап: Величини			
4. Відомі величини 9.Невідомі величини	5. Величини за одиницями	6. Величини за текстом 7.Величини за малюнком	8. Співвідношення величин
III етап: Формули			
10. Вибір формул	11. Перетворення формул	12. Формули з індексами	

Рівень 0	Рівень 1	Рівень 2	Рівень 3
		13. Підстановка формул	14. Алгебраїчні дії 15. Розв'язування рівнянь
IV етап: Обчислення			
19. Підстановка значень 20. Усні обчислення 25. Одиниця результату	16. Основні одиниці 17. Фізичні константи 18. Табличні величини 21. Обчислення з калькулятором 22. Округлення значень	23. Стандартний запис 24. Тригонометричні функції 26. Зручний вигляд	27. Дії з одиницями 28. Дії з векторами 29. Побудова графіків
V етап: Відповідь			
30. Явище-результат 31. Значення невідомих		32. Аналіз варіантів	

Розв'язування розрахункових задач

404 [3]. М'яч масою 400 г, кинутий вертикально вгору з швидкістю 20 м/с, впав у ту саму точку з швидкістю 15 м/с. Визначити роботу сили опору повітря.

Явище: а) рівносповільнений рух м'яча вертикально вгору під дією сили тяжіння та сили опору повітря;

б) рівноприскорений рух м'яча вертикально вниз під дією сили тяжіння та сили опору повітря.

Закон збереження енергії: м'яч за рахунок частини своеі енергії руху виконує роботу проти сил опору повітря.

$$W_{k1} \to W_{k2} + A'_o$$

Величини:
m = 400 г = 0.4 кг
v_1 = 20 м/с
v_2 = 15 м/с
A_o – ?
Формули:

$$\frac{mv_1^2}{2} = \frac{mv_2^2}{2} + A'_o$$

$$A'_o = \frac{m(v_1^2 - v_2^2)}{2}$$

$$A_o = -A'_o$$

Обчислення: $A'_o = \frac{0{,}4(20^2 - 15^2)}{2} = 35$ (Дж), $A_o = -35$ (Дж).
Відповідь: робота сили опору повітря дорівнює -35 Дж.

Розв'язування стандартних фізичних задач на комп'ютері

Комп'ютерна програма ELSA 3.1 написана на мові програмування BASICD з використанням модульної системи програмування, яка дає можливість використовувати окремі модулі програми, що містять до 50 команд. Особливістю цієї програми є також те, що вона працює в автоматичному режимі. В цій програмі використовуються тільки команди вибору файлів. Алгоритм програми повністю відповідає алгоритму розв'язування фізичних задач.

Метою створення цієї програми було перевірити можливість створення таких програм, які розв'язують текстові задачі з фізики та перевірка самого алгоритму розв'язування стандартних текстових фізичних задач.

Для розв'язування задач програма використовує текстовий файл даних, в якому вказується список формул та величин. Тексти задач записуються в текстовий файл задач. Програма утворює текстовий файл розв'язків, в якому містяться тексти задач та повні розв'язки задач.

При роботі програми існує проблема однозначності запису величин в текстах задачі. Так, використовуючи тексти задач із різних збірників без редактування, програма в більшості випадків не може ідентифікувати фізичні величини. Можливо це є також однією з причин труднощів в розв'язуванні фізичних задач учнями.

Файл задач

#Визначити магнітний потік , якщо через котушку площею 12 см2 проходить магнітна індукція 150 мТл ?

#Яка магнітна індукція , створює в котушці площею 25 см2 магнітний потік 240 мВб ?

#Визначити індуктивність котушки , площею 6 см2 , що знаходиться в магнітному полі з індукцією 50 мТл і через неі проходить струм 1.2 А ?

#Знайдіть зміну магнітного потоку , якщо магнітний потік змінився від .25 Вб до .7 Вб ?

Файл розв'язків

Визначити магнітний потік , якщо через котушку площею 12 см2 проходить магнітна індукція 150 мТл ?

Площа витка S= 12 см2 = .0012 м2
Магнітна індукція B= 150 мТл = .15 Тл
Магнітний потік Ф- ?
Магнітний потік Ф=B*S
Ф=(.15*.0012)=.00018 Вб

Яка магнітна індукція , створює в котушці площею 25 см2 магнітний потік 240 мВб ?

Площа витка S= 25 см2 = .0025 м2
Магнітний потік Ф= 240 мВб = .24 Вб
Магнітна індукція B- ?
Магнітний потік Ф=B*8 B=Ф/8 B"(.24/.0025)=96 Тл

#Визначити індуктивність котушки , площею 6 см2 , що знаходиться в магнітному полі з індукцією 50 мТл і через неі проходить струм 1.2 А ?

Площа витка S= 6 см2 = .0006 м2
Магнітна індукція B= 50 мТл == .05 Тл
Сила струму I= 1.2 А
Індуктивність L- ?
Магнітний потік Ф=L*I L=Ф/I
Магнітний потік Ф=B*S
L=(B*S)/I
L=((.05*.0006)/1.2)=(.00003/1.2)=.000025 Гн

#Знайдіть зміну магнітного потоку , якщо магнітний потік змінився від .25 Вб до .7 Вб ?

Магнітний потік Ф1= .25 Вб
Магнітний потік Ф2= .7 Вб
Зміна магнітного потоку dФ- ?
Зміна магнітного потоку dФ=Ф2-Ф1
Ф2=(.7-.25)=.45 Вб

Діагностика успішності учнів

Комп'ютерна програма DIAG 1.1 написана на мові програмування BASICD з використанням модульної системи програмування. Ця програма призначена для діагностики успішності учнів та тематичного контролю знань учнів, Програма визначає тематичну оцінку кожного учня, порівнює з тематичною оцінкою, виставленою вчителем. Потім визначає об'етивність тематичної оцінки та достатність контролю знань учнів. Також визначаються такі показники навчання учнів, як активність та якість знань, відвідування занять.

І нарешті визначається рейтингова оцінка знань учнів по 100-бальній шкалі. В основу ціеі оцінки покладено три положення:

- якщо учень одержав протягом теми тільки оцінки "5", це відповідає рейтинговій оцінці 50 балів;
- якщо учень одержав оцінки "5" на кожному уроці даної теми, тоді це відповідає рейтинговій оцінці 100 балів;
- якщо учень має недостатню кількість оцінок і має пропуски, тоді рейтингова оцінка відповідно зменшується.

Програма DIAG 1.1 створює такий текстовий файл:

#FS9T3 П'ятниця 22 січня 1999 року 17:17:16
Фізика. 9 клас. Тема 3. Основи динаміки.
14 год.24.11- 2.02.96 Задорожній М.І.

Прізвище	Пот.оцін	ТеК	ТО	ТОс	ОБ.	Дс	Акт.	Якіс	ПНав	%вд	РТ
БаранН	455554н	5	5	4.8	+	+	0.5	0.86	1.36	93	57
ГречкаВ	33333н	3	3	3	+	+	0.43	0.0	0.43	93	30
ДехтярукГ	нн33333н	3	3	3	+	+	0.43	0.0	0.43	79	30
ДятловО	444343	4	3	3.8	–	+	0.5	0.36	0.86	100	38
ЄськовІ	ннн3333н	3	3	3	+	+	0.36	0.0	0.36	71	30
ЗадорожнійВ	5нн5555555	5	5	5	+	+	0.64	1.29	1.93	86	73
КадченкоТ	н555555	5	5	5	+	+	0.5	1	1.5	93	63
КараушО	нн33333	3	3	3	+	+	0.43	0.0	0.43	86	30
КвасоваН	4554555	5	5	4.9	+	+	0.57	1	1.57	100	63
КвіткаО	3н3333н	3	3	3	+	+	0.43	0.0	0.43	86	30
КоржО	33333	3	3	3	+	+	0.43	0.0	0.43	100	30
КотовськаТ	н34333	3	3	3.1	+	+	0.43	0.07	0.5	93	31
ТатценкоР	ннннн44344	4	4	3.9	+	+	0.43	0.36	0.79	71	39
ЯловаВ	34344	3	4	3.3	–	+	0.43	0.21	0.64	100	33
ЯловаО	54ннннн	4	4	3.4	–	–	0.21	0.29	0.5	64	34
ЯловийС	нн34333	3	3	3.1	+	+	0.43	0.07	0.5	86	31
Тема(сер)	3.8	3.7	3.7	3.6	81	94	.45	.34	.79	88	40

Сер. поточна оцінка	=3.8	Об'єктивність оцінок в %	=81
Сер. оцінка тем. контролю	=3.7	Достатність контролю в %	=94
Сер. тем. оцінка вчителя	=3.7	Тем. оцінка DIAG 1.1	=3.6
Активність учнів за урок	=.45	Відвідування уроків в %	=88
Якість знань учнів за урок	=.34		
Показник навчання	=.79	Рейтингова оцінка знань	=40

References

[1] Buhaiov, O.I., Martyniuk, M.T. and Smolianets, V.V., 1995. *Fizyka. Astronomiia: probnyi pidruchnyk dlia 7 klasu sered. shkoly [Physics. Astronomy: a trial textbook for 7th grade high school].* 2nd ed. Kyiv: Osvita.

[2] Furman, A.V. and Kaluhin, O.I., 1994. Shkola rozvytku: nepiznani hrani fundamentalnoi idei [School of development: unknown aspects of the fundamental idea]. *Ridna shkola*, (6), p.27.

[3] Rymkevich, A.P., 1998. *Fizika. Zadachnik: 10-11 klassy [Physics. Problem book. 10-11 grades]*, Zadachniki "Drofy". 2nd ed. Moscow: Drofa.

[4] Shatalov, V.F., 1988. "Give Me a School!". *Soviet education*, 30(2), pp.54–67. Available from: https://doi.org/10.2753/RES1060-9393300254.

[5] Zadorozhnii, M.I., 1997. Novyi zmist fizychnoi osvity [New content of physical education]. *Dzherelo*, (24(121)), p.5. Available from: https://drive.google.com/open?id=1MzthQN2LgdIhiY5dIm_t5I3KDYrYqI21.

Instrumental and executive system of training and testing

Aleksandr P. Polishchuk[1]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

Abstract. One of the ways to increase the efficiency of computer support of the educational process is to provide the subject teacher with a simple and easy-to-use interactive instrumental and executive system with a set of educational materials and test tasks for testing knowledge. The presence of two subsystems – the subsystem of learning and the subsystem of testing and performance accounting allows the teacher to solve the problem of simultaneous questioning in the classroom. The student, in turn, gets the opportunity to re-test independently to correct his unsatisfactory grades on certain topics already covered. Pedagogical experience shows that there were many such systems; most of them were focused on learning programming. Today, almost all training systems created in the 1960s, except PLATO (Programmed Logic for Automated Teaching Operations), have no practical significance. In their didactic capabilities, they differed little from the systems that used the simplest technical learning tools and provided a rigid, virtually excluding dialogue determination of student activity. However, it was the first developments that stimulated interest in computer learning, and the development of hardware and software for personal computers led to greater opportunities for their use in learning.

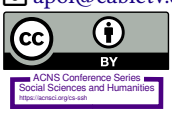
Keywords: educational process, instrumental and executive system of training and testing

Одним з напрямів підвищення ефективності комп'ютерної підтримки учбового процесу є надання викладачу-предметнику простої та зручної у користуванні інтерактивної інструментально-виконавчої системи з набором навчальних материалів та тестових завдань для перевірки знань. Наявність двох підсистем – підсистеми навчання та підсистеми тестування й обліку успішності дозволяє викладачеві вирішити проблему одночасового опитування при проведенні занять у комп'ютерному класі. Учень, в свою чергу, отримує можливість самостійного повторного тестування для виправлення не задовільняючих його оцінок з тих чи інших вже опрацьваних тем.

Педагогічний досвід свідчить, що таких систем було чимало; більшість з них були зорієнтовані на навчання програмуванню. Сьогодні майже всі створені у 60-і роки навчальні системи, крім PLATO (Programmed Logic for Automated Teaching Operations – програмована логіка для автоматизованих навчальних операцій [3]), не мають практичного значення. За своїми дидактичними можливостями вони мало відрізнялись від систем, що використовували найпростіші технічні засоби навчання і передбачали жорстку, практично виключаючу діалог детермінацію діяльності учнів. Проте саме перші розробки стимулювали інтерес до комп'ютерного навчання, а розвиток технічного і програмного забезпечення персональних комп'ютерів призвів до розширенння

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

apol@cabletv.dp.ua (A. P. Polishchuk)



ACNS Conference Series: Social Sciences and Humanities

можливостей їх використання у навчанні.

Не випадково в останні роки ми спостерігаємо стійкий інтерес до різного роду інструментальних оболонок типа Framework (на базі якої в Центрально-Міській гімназії м. Кривого Рога досить успішно вивчається курс комп'ютерного моделювання [7]), однак реалізація в них учбових задач відбувається не завжди адекватно, а найчастіше їх взагалі не можна втиснути у рамки цих систем. Інший їх різновид – авторські оболонки – як правило, призначені для якщо не разового використання, то для вузького кола користувачів. Більш того, навіть з такими системами познайомиться практично неможливо з тієї причини, що вони звичайно не виходять за межі учбового закладу, у якому розроблені.

Звичайною вимогою є те, що інструментальна частина системи повинна обслуговувати функції створення, знищення, коректування учбових матеріалів і тестових завдань, щоб кожен викладач мав змогу формувати виучуваний курс під свої методи і конкретні умови праці. У той же час поставка "порожніх" оболонок, навіть дуже гарних, малоефективна, оскільки процес їх наповнення вимагає витрат часу і вимагає замислюватись над тим, що саме ти хочеш зробити (наприклад, закладений нами у розроблену оболонку елементарний курс інформатики складається з 100 уроків й близько 1000 тестів загальним обсягом більше 1 Мб).

То ж як усунути чи пом'якшити зазначені вище недоліки? Насамперед, така система повинна бути гарно документованою і мати хоча б примітивну систему допомоги, бажано контекстно-залежну. Вона має бути відкритою і здатною до обміну даними з іншими програмами, не змушуючи користувача працювати з навчальним матеріалом лише в ній. І, звичайно ж, вона повинна мати інтуїтивно зрозумілі засоби керування, наприклад, в стандарті Commom User Access фірми IBM [6].

Розроблена нами система TEST в одному з своїх варіантів є повноцінною Windows-аплікацією і в цьому варіанті може функціонувати на IBM-сумісних PC-386 та вище під керуванням Windows for Workgroups 3.11+. Це означає, що вона може виконуватися одночасово з іншими програмами, не заважаючи їм, а користувач може переходити за необхідності у будь-яку з них. Так, скажімо, при вивченні курсу чисельних методів типовими програмами, що сумісно працювали на одній ЕОМ, були

- Диспетчер програм (progman) – оболонка, з якої відбувається запуск усіх інших программ
- Pri/Sec (ruslat) – русифікатор Windows for Workgroups
- Borland C++ 3.1 for Windows (bcw) – інтегроване середовище мови C++, у якому виконувалися завдання з курсу чисельних методів
- Інструментально-виконавча система навчання та тестування (test), яка містила теорію з методу, завдання на лабораторну роботу, методичну допомогу, а в особливо важких випадках і окремі фрагменти програмної реалізації виучуваного методу [5] (figure 1).

Система пропонує користувачу 2 основних режими роботи: "Розробка" та "Експлуатація" (навчання та тестування). Режим розробки призначений для викладача і захищений від випадкового та несанкціонованого використання кнопкою паролю, що

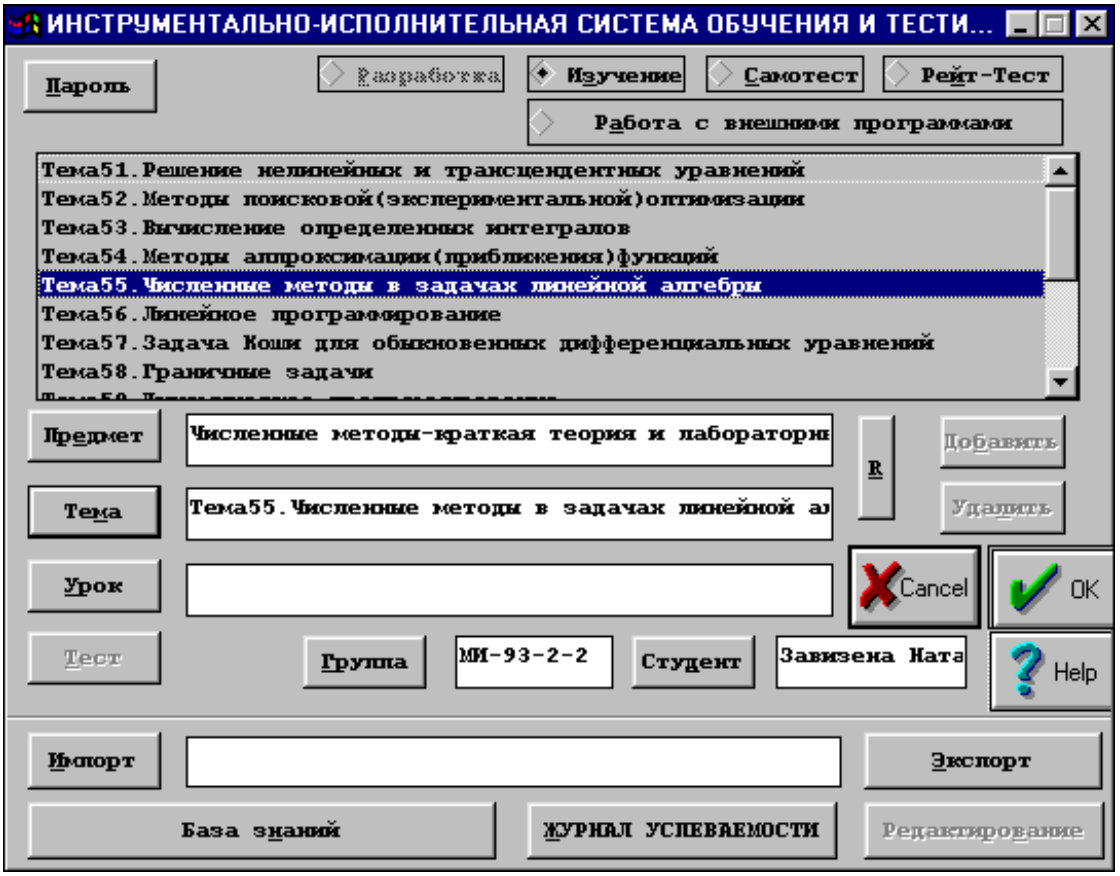


Figure 1: Інструментально-виконавча система навчання та тестування.

перевіряє наявність у дисководі ключового носія – дискети, а у разі її відсутності запитує пароль з клавіатури. Запуск системи здійснюється з DOS (з побіжним автоматичним запуском Windows) або безпосередньо з Windows. Windows система у користуванні попервах складна, тому запуск з Windows-програми з DOS була включена як можливість, що полегшує її експлуатацію. Як показав досвід, для успішного початку курсу лабораторних робіт з чисельних методів у Windows студентам, зовсім з нею не знайомим, досить однієї-двох пар. Після запуску встановлюється режим експлуатації. В системі відсутні багаторівневі меню вибору режимів та різних установок, такі звичні для складних систем – з метою спрощення всі органи керування зосереджені в головному діалоговому вікні у вигляді панелі керування з набором кнопок та текстових субвікон. Такого роду інженерній підхід – створення пульту або панелі керування – дозволяє постійно мати перед очима режим роботи, вибрану предметну область, групу та ім'я студента, що, як показав досвід користування, прискорює роботу, на відміну від меню-орієнтованих систем.

Для вибору виду робіт в режимі експлуатації у верхній частині панелі розташовано набір кнопок, що дозволяють обрати один з чотирьох видів діяльності, а у середній її

частині міститься велике вікно для відображення різних режимно-залежних списків (з можливістю вибору елемента списку) або журналу успішності для перегляду. Режим навчання реалізується після задання предметної області, вибір якої здійснюється за умовною трьохрівневою схемою ієрархії типа “Предмет – тема – урок”. На будь-якому рівні “натискання” кнопки “ОК” викликає перехід у вікно демонстрації навчального матеріалу одного або послідовно усіх уроків теми чи предмету (figure 2).

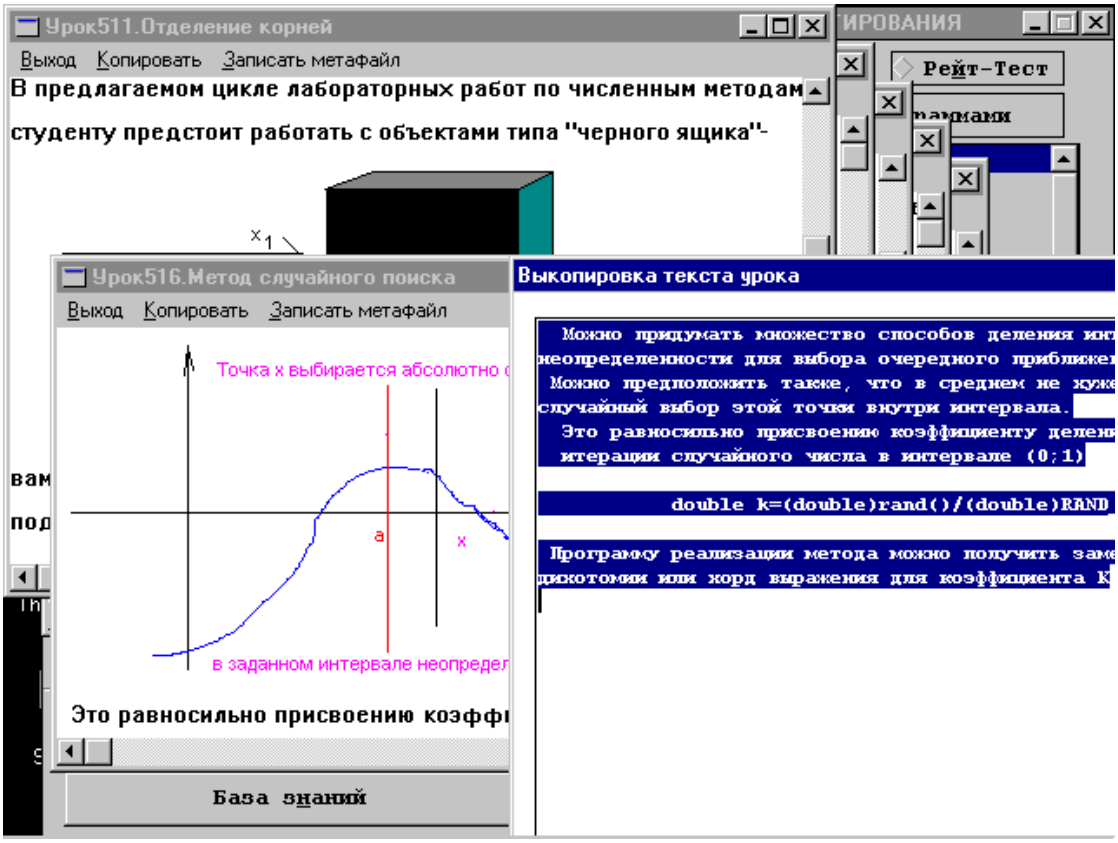


Figure 2: Відображення навчального матеріалу.

В режимі навчання доступні усі стандартні функції перегляду (редагування навчального материалу в цьому режимі заборонено), виділення та переносу блоків тексту через Clipboard – буфер обміну Windows, який є одним із засобів обміну даними між різними аплікаціями.

На жаль, оцінка, незважаючи на досвід Ш. О. Амонашвілі [1], досі залишається мірою знань, тому була здійснена спроба сумістити як тестування “для себе”, так й “для вчителя”. Режим “Самотестування” слугує допоміжним засобом навчання та анонімної перевірки знань без занесення оцінки до журналу успішності. В цьому режимі після вибору предметної області здійснюється перехід в діалогову панель тестування; набір текста відповіді здійснюється з клавіатури або копіюванням з буферу обміну (туди може бути занесена важка для запам’ятовування частина уроку або весь урок в режимі навчання,

так що навчальний матеріал при тестуванні, як кажуть, “під рукою”) (figure 3).

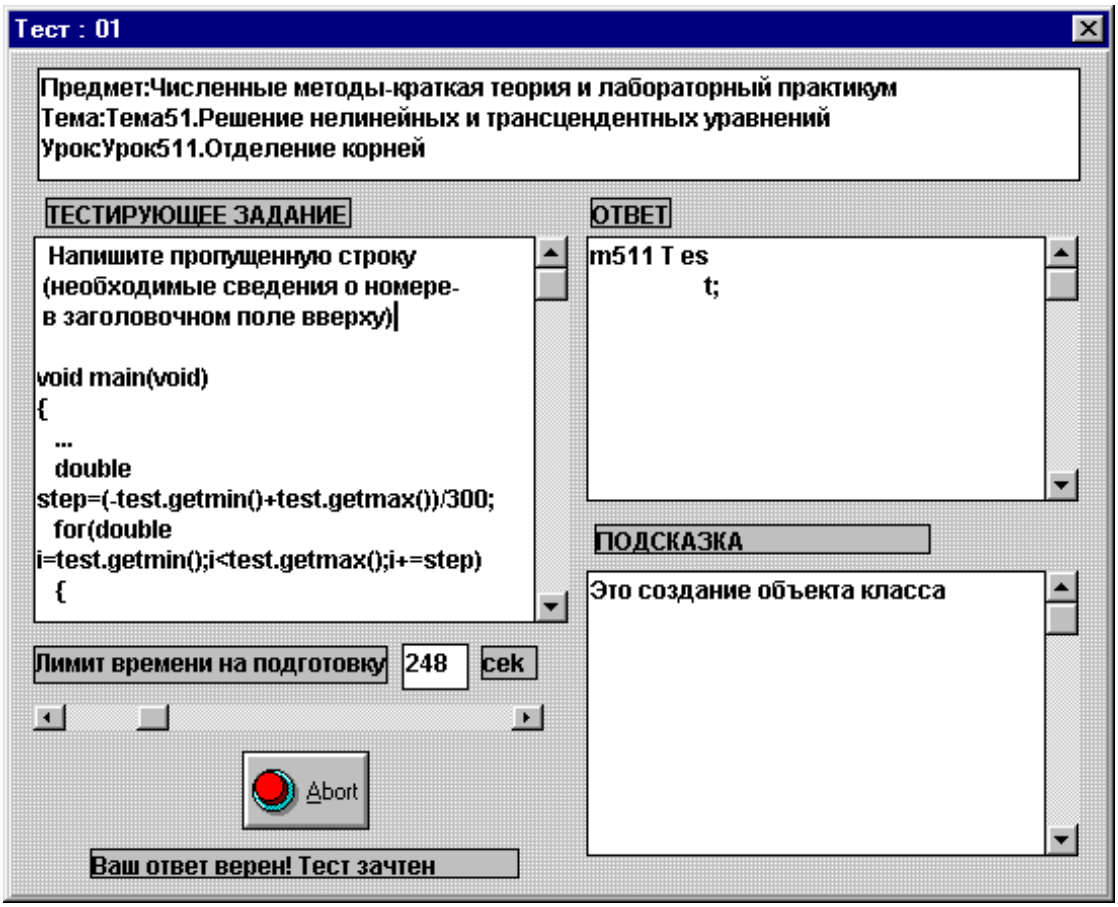


Figure 3: Режим тестування.

Система оцінювання тестового завдання м’яка та демократична – заощаджений час на підготовку до відповіді призводить до підвищення оцінки з заданим коефіцієнтом, а прострочений - до незаліку теста; за підказку ж доводиться розраховуватися деяким зниженням оцінки, яка виставляється після проходження обумовленої кількості тестів для даного уроку (не менше трьох).

Аналіз тексту відповіді – найбільш важка і відповідальна частина системи. Якщо б вона була закритою і обмежувалась однією областю знань, то проблем би не виникло. У літературі згадується, наприклад, система SOPHIE (sophisticated instructional environment). Це імітаційна програма, призначена для навчання пошуку несправностей у електронних ланцюгах [2]. В межах даної дуже вузької області ця програма дозволяє аналізувати відповідь, вказуючи не лише на помилку, а й на послідовність дій, що спричинили її. Сьогодні не існує ефективних методів оцінки смислового змісту відповіді за довільною тематикою, а тому доводиться покладатися на формальний аналіз тексту (збіг з еталоном). Існує ряд класифікацій тестів, але всі вони, від примітивних вибіркових до конструкцій

з послідовності слів та фраз, набираються з клавіатури і первинним матеріалом для аналізу завжди є текст.

Очевидно, що випадкова помилка в одному символі при наборі не повинна призводити до незаліку тесту. Скажімо, возьмемо який-небудь першоквітневий тест на тему "Саме...":

Питання : Напишіть прізвище російської радянської актриси, що складається з більш як 30 букв.

Еталонна відповідь : Архневолоточерепопіндрюковська

Найзаядліші балетомани навряд чи з першого разу зможуть написати його правильно (не кажучи вже про те, щоб вимовити). Крім того, в окремих випадках доводиться допускати можливість перестановки слів у фразі, тому в розглядуваній системі використовується багаторівневий аналіз. Якщо після видалення пропусків та переведень рядків немає повного співпадання з еталоном, для тестів із зведеним флагом складності виконується додатковий змістовний аналіз. Певна річ, у тестах з числовими відповідями, відповідями з визначенням номеру вірного (хибного) варіанту і таке інше зазначених проблем не виникає.

Режим рейтинг-тестування з занесенням оцінок до журналу реалізується після вибору групи та прізвища у списку, які за необхідності поповнює викладач. Різниця між цими режимами несуттєва, тому що студенту, який жодного разу не тестувався, за всі уроки у журналі стоїть оцінка “2”, і його мета – не набирати гарні оцінки, щоб, протестувавшись на “відмінно” з одного уроку, одержати вищій бал з усього предмету (як це зроблено у деяких оболонках), а виправляти ці двійки. Такий підхід може здатися не дуже гуманим, однак альтернативи йому нема – тільки так можна забезпечити проходження учнем тестів якщо не за всіма, то хоча б з більшої частини уроків. Завдяки такому підходу перед ним не існує психологічного бар’єру, пов’язанного з можливістю отримання поганої оцінки – вона в нього вже стоїть.

Режим “Робота з імпортованими програмами” призначений для об’єднання “під одним дахом” усіх необхідних для вивчення курсу допоміжних інструментальних засобів. Це можуть бути навчальні програми з окремих розділів, анімаційні програми та навчаючі ігри, транслятори з мов програмування і взагалі будь-які програмні засоби.

Режим “Розробка” стає доступним після натиснення кнопки “Пароль” при наявності в дисководі ключової дискети або при її відсутності після набору пароля. Відмінність режиму розробки від режиму експлуатації полягає у доступності функцій коректування та редагування списків предметів, тем, уроків, тестов, груп та їх складу, імпортованих програм, текстів уроків і тестових завдань. Для формування уроків можуть використовуватися вже готові текстові файли в кодировці ANSI чи OEM (після перекодування утилітою fconvert або через стандартний Windows-редактор Write [Word-Pad] і буфер обміну). Ця можливість виявляється не тільки корисною, а й незамінною у тих випадках, коли курс лекцій вже набрано раніше. В оболонці також передбачена можливість експортування навчального матеріалу у файли в ANSI чи OEM кодуванні (тільки текст) чи у форматі Windows Metafile (текст + графіка). Отримання твердої копії з файла залишається на розсуд користувача.

Система є об’єктно-орієнтованою; доступ до об’єктів класів “Урок” та “Тест” на диску прямий завдяки невеликим за обсягом асоціативним словникам, які одразу

завантажуються до оперативної пам'яті і безперервно коректуються у процесі роботи. Для прискорення дискових операцій запис кожного разу іде у кінець відповідного файлу, так що у процесі роботи на диску може виявитися декілька копій одного й того уроку чи тесту різної свіжості. Щоб запобігти такого марнотратства дискової пам'яті (а іноді це сотні килобайт), наприкінці сеансу роботи з системою користувачу пропонується пересортувати дані з метою вилучення "мертвих" даних, на які немає посилань у словниках. Це опціональна операція, оскільки при великих об'ємах даних вона віднімає деякий час.

Зараз обговорювана оболонка наповнена уроками та тестами з чотирьох предметів: "Чисельні методи", "Елементарний (ввідний) курс інформатики", "Основи наукових досліджень", "Програмування в Turbo Vision". Апробація системи проводилась у чотирьох групах студентів 3-го курса (поток МІ-93) й, окрім усунення ряду помилок і недоробок, показала такі результати.

Засвоєння органів керування досягається за 10-20 хвилин за умови хоча б мінімального досвіду спілкування з Windows; у разі його відсутності зайняття розпочинаються з підручника до операційного середовища Windows. При серйозному відношенні викладача до оцінок у системному журналі успішности останній грає ключову роль в активному використанні студентами навчальних матеріалів і відвідувань додаткових занять у комп'ютерному класі. Це особливо ефективно для студентів, що мають проблеми із засвоєнням курсу. Спостерігаються спроби багаторазового тестування, конспектування еталонних відповідей при незаліках тестів з метою майбутнього використання, активний пошук відповідей в текстах уроків; студент несамохіть втягується у процес "самонатаскування" з погано засвоєних розділів (у вільний від занять час). Інший ефект – підвищення продуктивності, наприклад, при навчанні програмуванню, виникає за рахунок копіювання фрагментів програм з наведених в уроках методичних матеріалів до інтегрованого середовища розробки, що дає можливість підвищити складність завдань з програмування шляхом "збирання" програм з рекомендованих фрагментів. Крім того, інструментально-виконавча система передбачає

1. Індивідуальний темп навчання, при якому студент може повернутися до будь-якої методичної рекомендації та прикладу, ще раз пройти погано засвоєний матеріал [4].
2. Диференційоване тестування, що дозволяє встановити рівень вимог студента щодо оцінки.
3. Незалежність від змісту відповіді при тестуванні та матеріалу при вивченні дає змогу наповнювати систему довільним матеріалом (як приклад можна навести курс "Основи наукових досліджень" та батарею тестів з педагогіки).

References

[1] Amonashvili, S., 1989. Non-directive teaching and the humanization of the educational process. *Prospects*, 19(4), pp.581–590. Available from: https://doi.org/10.1007/BF02206752.

[2] Brown, J.S., Burton, R.R. and Bell, A.G., 1975. SOPHIE: A step toward creating a reactive

learning environment. *International Journal of Man-Machine Studies*, 7(5), pp.675–696. Available from: https://doi.org/10.1016/S0020-7373(75)80026-5.

[3] Control Data Corporation, 1981. PLATO User's Guide. Available from: http://www.bitsavers.org/pdf/cdc/plato/97405900C_PLATO_Users_Guide_Apr81.pdf.

[4] Gergei, T. and Mashbits, E.I., 1986. Psychological and pedagogical problems of effective computer use in the educational process. *Soviet Education*, 28(10-11), pp.213–229. Available from: https://doi.org/10.2753/RES1060-9393281011213.

[5] Polishchuk, A.P. and Semerikov, S.A., 2022. Concept of the course "Numerical methods in object methodology". *ACNS Conference Series: Social Sciences and Humanities*, 1, p.01003. Available from: https://doi.org/10.55056/cs-ssh/1/01003.

[6] Taylor, K., 1990. IBM Systems Application Architecture: Common User Access from first principles. *Computing & Control Engineering Journal*, 1, pp.123–127(4). Available from: https://digital-library.theiet.org/content/journals/10.1049/cce_19900033.

[7] Teplytskyi, I.O., 2022. Physics models in the course "The basics of computer simulation". *ACNS Conference Series: Social Sciences and Humanities*, 1, p.01002. Available from: https://doi.org/10.55056/cs-ssh/1/01002.

Development of algorithms that simulate the solution of a problem by a person

Iurii V. Filatov[1,2]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*
[2]*Saksahanskyi Natural Science Lyceum of the Kryvyi Rih City Council in Dnipropetrovsk oblast, 32A Meleshkina Str., Kryvyi Rih, 50071, Ukraine*

Abstract. Some algorithms, which are often based on the use of elements of higher mathematics, possessing high speed and compact coding in algorithmic languages, are poorly mastered by most students. It can be assumed that this is due to the difficulty of presenting the principles of their work in the form of human actions in ordinary situations. Thus, a certain contradiction arises between the way of solving the problem that a person resorts to without using a computer and the way we force our computer to solve this problem. Comparison of the process of explaining algorithms speaks in favor of algorithms imitating human thinking. The discussion of the advantages of the algorithms themselves is beyond the scope of this article and undoubtedly deserves a separate study. If artificial intelligence is created, then its creator or creators will certainly be ranked among the outstanding geniuses in the history of civilization, no matter what algorithms it uses. However, so far there is no one to solve problems for us and create algorithms, so we will use all available means and try to teach this to children.

Keywords: algorithms, solution of the problem

«Движенья нет» – сказал мудрец брадатый.
Другой смолчал и стал пред ним ходить,
Сильнее бы не смог он возразить.
Хвалили все ответ замысловатый.

А. С. Пушкин [1]

Некоторые алгоритмы, в основе которых зачастую лежит использование элементов высшей математики, обладая высоким быстродействием, и компактной кодировкой на алгоритмических языках, плохо усваиваются большинством учащихся. Можно предположить, что это связано с трудностью представления принципов их работы в виде действий человека в обычных ситуациях. Таким образом, возникает некоторое противоречие между тем способом решения задачи, к которому прибегает человек без использования компьютера и тем, которым мы заставляем решать эту задачу наш компьютер.

Рассмотрим возникновение такого противоречия на конкретном примере. Для этого

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine
imfilatov@mail.ru (I. V. Filatov)

ACNS Conference Series: Social Sciences and Humanities

сравним алгоритмы “пузырьковой” сортировки и быстрой сортировки Хоара (Hoare) [2]. Если мы будем сравнивать быстродействие этих алгоритмов, то заметим, что результаты прогонов на массивах разного размера, несортированных вовсе и почти упорядоченных, а также при различной настройке компилятора различны. Тем не менее, алгоритм Хоара работает в большинстве случаев значительно быстрее (исключение составит большой массив, в котором требуется поменять местами несколько пар стоящих рядом элементов). Посмотрим теперь, как обстоит дело с объяснением механизма работы этих алгоритмов.

Для объяснения работы “пузырьковой” сортировки будем ходить вдоль длинного ряда карточек с числами слева направо (можно просто записать числа на доске, но в этом случае придется отвлекаться на стирание и переписывание чисел). Напомним детям, что все видели, как учитель физкультуры выстраивает новый класс по росту, проходя вдоль шеренги учеников, которые встали в произвольном порядке, и, меняя местами двух стоящих рядом юных спортсменов, если более высокий стоит ближе к левому флангу. (Здесь уместно обратить внимание учащихся на то, что человек и компьютер сравнивают за один раз только две величины, т.е. кажущаяся возможность сравнить сразу больше двух величин, на самом деле сводится к некоторому количеству попарных сравнений)

Итак, последуем примеру учителя физкультуры, при этом будем держать в руке флажок, поднимая его в начале каждого прохождения от начала к концу шеренги чисел. При этом в поле нашего зрения одновременно находятся только две лежащие рядом карточки, и если числа на них нарушают избранный порядок (возрастание или убывание чисел в ряду должно соответствовать поставленной задаче), то мы меняем карточки местами и опускаем флажок (объясняем, что если он не был поднят, то эта процедура проделывается вхолостую, дабы избежать лишней проверки состояния флажка). В конце каждого такого похода проверяем поднят ли флажок. Если да, то, стало быть, карточки местами не менялись и следовательно сортировка окончена.

Понятно, что для столь наглядной демонстрации принципа работы быстрой сортировки Хоара потребуется минимум три человека, которые будут перемещаться вдоль ряда чисел и перекликаться, выясняя чьё число больше, но кроме этого нужен хранитель стека (или массива заменяющего стек при не рекурсивной реализации алгоритма), и, желательно, менеджер, который будет регламентировать очерёдность действий этих исполнителей. (Можно предложить построение по росту методом Хоара для КВНа.) При этом наибольшую трудность вызывает имитация стека, т.к. объём “стека” нормального человека, как правило, невелик. Попробуйте проверить глубину стека, предлагая выворачивать наизнанку числа; пока дело касается примеров “17”—“71” ошибок почти не бывает, но попробуйте на “3.244.179” получить ответ “9.714.423”.

Итак, сравнение процесса объяснения алгоритмов говорит в пользу алгоритмов имитирующих мышление человека. Обсуждение же достоинств самих алгоритмов выходит за пределы темы данной статьи и, несомненно, заслуживает отдельного исследования. Если искусственный интеллект будет создан, то его создатель или создатели наверняка будут причислены к выдающимся гениям в истории цивилизации, независимо от того какие алгоритмы он будет использовать.

Однако, пока некому решать за нас задачи и создавать алгоритмы, поэтому будем использовать все имеющиеся средства и постараемся научить этому детей.

Прежде чем перейти к решениям конкретных задач, для которых мне удалось построить алгоритмы имитирующие решение задачи человеком, приведу схему коллективной работы над задачами. Применение этой схемы позволило развить в учениках способность совмещать поиск оригинального решения с модернизацией известных алгоритмов.

1. Постановка задачи, уточнение требований, разбор предельных и вырожденных случаев.
2. Конференция на тему «Как бы я решил эту задачу, не имея компьютера и не зная алгоритмических языков». Каждый докладчик описывает последовательность своих действий, а слушатели оппонируют, предлагая "неудобные" для данного алгоритма случаи.
3. Обсуждение проблем кодирования на Turbo Pascal, доклады о встреченных ранее недостатках некоторых конструкций.
4. Самостоятельная модернизация и кодирование алгоритма, принятого за базовый в процессе обсуждения докладов.
5. Сравнение работоспособности и эффективности полученных программ, доклады победителей тестирования.
6. Необходимая индивидуальная доработка программ.

Следует отметить, что подобный подход даёт наилучшие результаты во внеклассной деятельности, при работе с небольшими группами учащихся, но ведь задача массового воспитания программистов и не ставится.

Задача о направлении обхода.

Постановка задачи:

Произвольный N-угольник задан координатами (x, y) своих вершин, пары координат перечислены в той последовательности, в которой соединяются вершины, последняя вершина соединяется с первой. Требуется определить соединялись вершины обходом по часовой стрелке или против.

Необходимое уточнение:

Первая из перечисленных точек не обязательно лежит “с краю”, кроме того, поскольку N-угольник произвольный, то точка $K + 1$, а равно и $K - 1$ расположены произвольно относительно точки с номером K.

Задача имеет красивое и компактное решение на языке векторной алгебры.

Разработка алгоритма решения:

Представим себе, что N-угольник представляет собой систему пронумерованных колышков, соединённых бечевой, а мы находимся примерно посредине этого лабиринта, возле колышка с номером K. Краёв системы не видно, зато мы хорошо видим, что севернее (юго-западнее и т.д.) находится колышек с номером K, а южнее (северо-восточнее и т.д.) с номером $K + 1$ (см. рис. 1).

Понятно, что такая ситуация не прояснит направления обхода, даже если мы точно определим направление на соседние колышки. Сравнение рисунков 1 и 2 убеждает нас в этом.

Так, находясь у третьего колышка мы обнаружим, что направление на второй и четвёртый одинаково, но на первом рисунке обход осуществлялся по часовой стрелке, а на втором против.

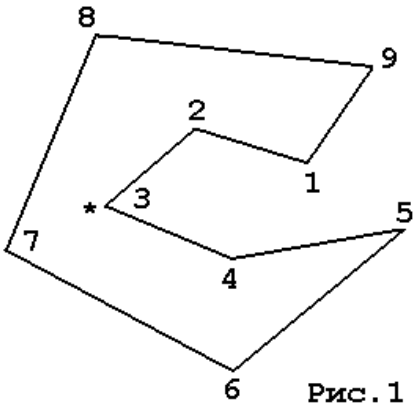


Рис.1

Попробуем теперь идти от стороны 2-3 к стороне 3-4. Отыскивая отличия, заметим, что при движении внутри замкнутой фигуры угол увеличивается, если обход совершался по часовой стрелке (рис. 1) и уменьшается в противном случае (рис. 2). При этом угол будем измерять от оси X до направления из вершины K на вершины $K-1$ и $K+1$ (как в тригонометрии, если считать вершину K началом координат).

Для того же чтобы не сомневаться, что мы внутри, выберем например верхнюю левую вершину, тогда оба угла будут наверняка больше 180° и меньше 360° (один из них может быть равен 360°). Заметим, что можно выбирать правую нижнюю (в этом случае углы будут лежать в пределах от 0° до 180°) и любую другую самую крайнюю вершину.

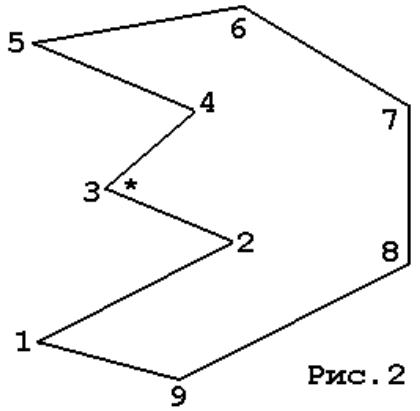


Рис.2

Таким образом, однозначно определив углы и выяснив какой больше, получаем ответ на поставленный вопрос.

Попробуем теперь преобразовать наши рассуждения в алгоритм подходящий для реализации на каком-либо языке программирования:

1. Найдём верхнюю левую вершину, для этого достаточно найти вершину с максимальной ординатой, а если таких окажется несколько, то среди них — ту, у которой абсцисса минимальна. Запомним номер этой вершины (допустим K).
2. Определим углы на $K-1$ и $K+1$ вершины. Не забудем, что последняя вершина соединяется с первой и, следовательно, если $K=N$, то следующая за K вершина — первая, а если $K=1$, то предыдущая — N. Оба угла больше 180°, кроме того надо считать угол равным 270°, если ордината $K-1$ или $K+1$ вершины равна ординате K вершины, а если соответственно равны абсциссы, то угол равен 360°, а не 0° или 180° (как было замечено выше). Поскольку нас интересует относительная величина этих углов, то перевод радиан в градусы, при написании программы, несущественен.
3. Сравним углы. Если вершины соединялись (т.е. перечислялись в условии) обходом по часовой стрелке то угол наклона стороны соединяющей вершины $K-1$ и K меньше, чем угол наклона стороны соединяющей вершины K и $K+1$ (см. рис. 3).

4. Таким образом, вычислив и сравнив углы, мы получаем ответ на поставленный вопрос.

Такой подход к разработке алгоритма и реализованные здесь идеи позволяют не только решить поставленную задачу, но и дают учащимся возможность самостоятельно строить алгоритмы для целого класса задач связанных с геометрией на плоскости, например классическую задачу о точке внутри многоугольника.

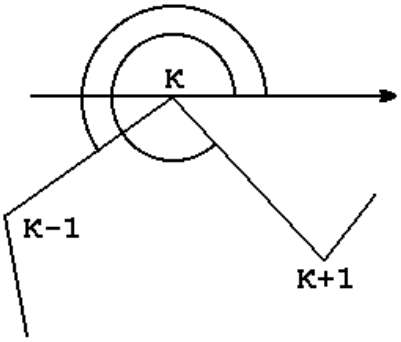


Рис. 3

Задача о точке, лежащей внутри N-угольника, достаточно известна, имеет несколько вариантов формулировки и, соответственно, несколько алгоритмов решения. Остановимся на решении задачи о нахождении точки лежащей внутри произвольного N-угольника (частные алгоритмы работающие только на выпуклых многоугольниках заслуживают отдельного разговора).

Задача о нахождении точки, лежащей внутри произвольного N-угольника.

Постановка задачи:

Произвольный N-угольник задан координатами (x, y) своих вершин, пары координат перечислены в той последовательности, в которой соединяются вершины, последняя вершина соединяется с первой. Требуется найти координаты точки, которая лежит внутри этого N-угольника.

Необходимое уточнение:

Необходимо указать минимальное расстояние d от искомой точки до ближайших сторон N-угольника, так как точность компьютерных вычислений ограничена разрядной сеткой или количеством шагов специального алгоритма, то точка математически лежащая на стороне N-угольника, программно может быть определена, как лежащая внутри (или вне) этого N-угольника. (Попробуйте, например, сравнить значение выражений $\sin 45°$ и $0.5\sqrt{2}$ на своём компьютере. Скажем, в среде Turbo Pascal 7.0 условие SIN(PI/4)=0.5*SQRT(2) выполняется, если правильно настроить компилятор, при {N+}-True, {N-}-False.)

Принимая во внимание, что при бесконечном уменьшении просвета между сторонами, попасть внутрь невозможно, разумно потребовать, чтобы расстояние между несмежными сторонами превышало $2d$.

Разработка алгоритма решения:

Итак, уяснив требования к решению, приступим к разработке алгоритма. Допустим, что нам дан многоугольник (см. рис. 4).

Мы хотим найти точку, которая заведомо лежит внутри него. Отметим, что существует алгоритм для определения, лежит ли данная точка внутри данного многоугольника, путём подсчёта количества пересечений луча, проведённого из этой точки со сторонами многоугольника. Этот алгоритм нетрудно использовать для решения данной задачи если ставить точку наугад до тех пор пока не попадём внутрь. Однако здесь возникают

определённые трудности. Во первых, как учитывать пересечения с вершинами (на рис. 5 точка лежит внутри фигуры А — нечётное количество пересечений и вне фигуры Б — чётное, а на рис. 6 количество пересечений чётное для обеих фигур; потребуется дополнительно определять лежат ли рёбра, выходящие из вершины, по одну сторону от луча или нет).

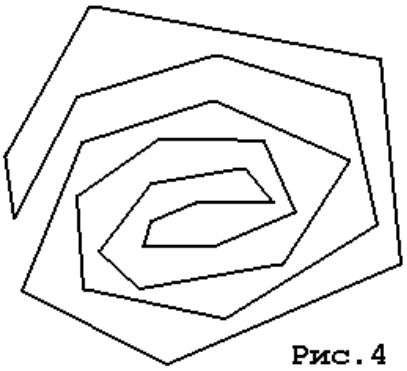


Рис. 4

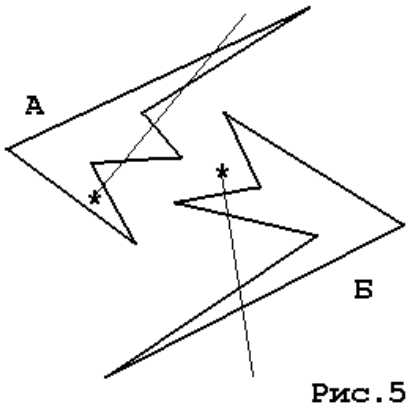


Рис. 5

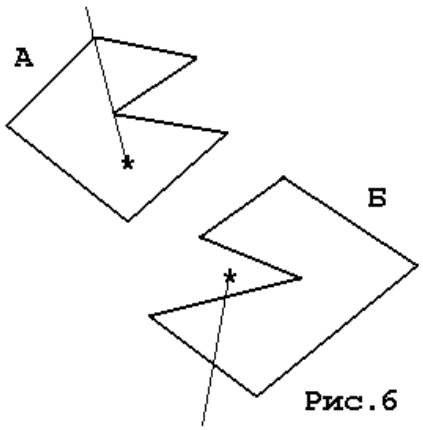


Рис. 6

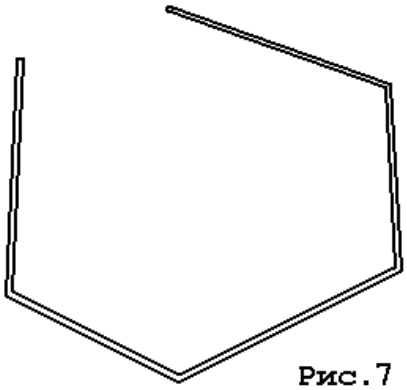


Рис. 7

Во-вторых, как долго нам придётся “стрелять”, пока мы попадём внутрь фигуры подобной той, что приведена на рис. 7.

Попробуем придумать свой алгоритм, используя метод из предыдущей задачи.

Итак, имеется достаточно запутанный случай (см. рис. 4). Трудно предположить, что человек поставит точку наугад в центре сооружения, а затем начнёт определять попал ли он внутрь, особенно если учесть, что при этом возможно мы не видим всего многоугольника. Скорее всего, человек постарается поставить точку где-нибудь поближе к краю (см. аналогичные соображения в предыдущей задаче). Дальнейшие действия очевидны:

1. По аналогии с предыдущей задачей найдем углы наклона сторон соединяющих верхнюю левую вершину с предыдущей (угол A) и последующей (B).

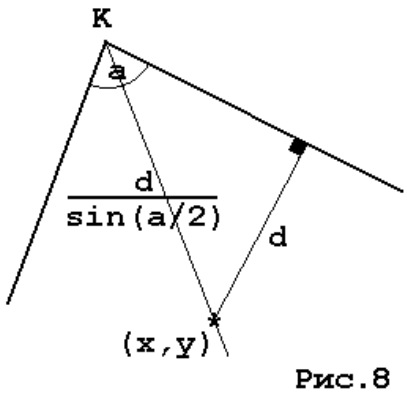


Рис.8

2. Таким образом угол наклона отрезка соединяющего искомую точку с K вершиной лежит между углами найденными в п. 1, для простоты можно считать его средним арифметическим этих углов ($C = (A + B)/2$).
3. Остаётся с помощью несложных тригонометрических расчётов найти расстояние от вершины, которое обеспечит удалённость искомой точки от сторон прилежащих к K вершине и определить (x, y) координаты этой точки (см. рис. 8). При этом α = больший угол − меньший, и расстояние от K до искомой точки $r = d/\sin(\alpha/2)$. Тогда координаты: $x = x_K + r\cos C$ и $y = y_K + r\sin C$.

References

[1] Pushkin, A.S., 1947. *Dvizhenie [Motion].* Izd-vo AN SSSR, p.432. Polnoe sobranie sochinenii, 1837—1937. Tom 2. Stikhotvoreniia, 1817—1825. Litceiskie stikhotvoreniia v pozdneishikh redaktciiakh. Kn. 1. Available from: http://feb-web.ru/feb/pushkin/texts/push17/vol02/y21-432-.htm.

[2] Sedgewick, R., 1978. Implementing quicksort programs. *Commun. ACM*, 21(10), p.847–857. Available from: https://doi.org/10.1145/359619.359631.