даний по каждой дисциплине полностью соответствуют спецификациям Центра Тестирования МО РФ, на который возложено проведение ЕГЭ в нашей стране. Задания разрабатывались группами специалистов из Уральского государственного университета и Уральского государственного педагогического университета. Количество однотипных заданий, а также особенности контролирующей системы обеспечивают значительную вариативность, что позволяет ученику многократно проходить тестирование по одной и той же дисциплине. В-третьих, в систему заложена именно та схема оценивания результатов сдачи, которая принята в ЕГЭ. Таким образом достигается возможно более полная имитация действий ученика на экзамене.

В настоящее время осуществлена первая очередь проекта, в которой реализован тренаж по трем дисциплинам – русскому языку, математике и физике. В дальнейшем предполагается расширение круга дисциплин, а также повышение вариативности заданий. Система располагается на CD-носителе и может использоваться как на локальном компьютере, так и в локальной сети учебного класса. Помимо CD учащийся получает пособие, в котором разобраны примеры выполнения заданий типа С. По нашему мнению, подобный комплекс позволит заметно облегчить подготовку школьников к ЕГЭ и снять целый ряд связанных с ним стрессовых моментов.

РАЗРАБОТКА КРИТЕРИЕВ ОБЪЕКТИВНОЙ ОЦЕНКИ ЗНАНИЙ И УМЕНИЙ УЧАЩИХСЯ ПО ПРОГРАММИРОВАНИЮ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАТЕМАТИЧЕСКОЙ СТАТИСТИКИ

И. В. Рожина
Нижнетагильский государственный педагогический институт

В последние годы математические методы нашли широкое применение в психолого-педагогических исследованиях. Из них, пожалуй, наибольшее применение получили статистические методы обработки результатов измерений. Известно большое количество примеров плодотворного использования статистических методов для проведения весьма разнообразных и важных исследований. При этом сложность состоит в том, что не существуют какие-либо универсальные, пригодные для всех случаев параметры оценки эффективности методов обучения. В связи с этим возникает необходимость дальнейшей разработки методик применения математических методов в психолого-педагогических исследованиях.

Применение статистических методов предполагает, что выделены величины, характеризующие исследуемое явление. Каждая такая величина должна быть четко определена, необходимо установить ее характер и способ из-

мерения, при этом надо выяснить, является ли этот способ измерения точным или приближенным.

Все возможные величины можно разделить на первичные и вторичные. К *первичным* величинам относят те, измерение которых может быть произведено непосредственно с помощью единицы меры, являющейся значением этой же величины. Примерами *первичных* величин являются *время* и *расстояние*. *Вторичными* величинами являются те, измерение которых непосредственно или невозможно, или очень затруднительно, поэтому их измерение как-то сводят к измерению первичных величин. К ним относятся практически все величины, *связанные с педагогическим процессом*.

Так как обучение относится к целенаправленным процессам, а оценивание и прогнозирование хода обучения базируется на оценке его результатов в различные моменты времени, включая прошлые и будущие, то для решения данной задачи можно использовать методы и подходы, разрабатываемые и применяемые в квалиметрии и теории эффективности. Для этого необходимо ввести соответствующие показатели и критерии оценивания качества результатов и эффективности обучения. Формирование таких показателей и критериев производится на основе следующей системы понятий: свойство, характеристика свойства, количественные характеристики, качественные характеристики, оценивание свойства, оценка [3. С. 150–151].

Основным средством обучения в рассматриваемом нами курсе «Объектно-ориентированное программирование и визуальное проектирование» является задача, поэтому наиболее логично организовать определение достигнутого уровня обучения с использованием *задач*. Не секрет, что на практике оценка знаний и умений учащихся выставляется учителем зачастую субъективно и зависит от множества факторов, поэтому необходимо уйти от субъективизма и выработать критерии объективного оценивания.

Выделим две характеристики для оценки решения задачи: *качественная* и *количественная*.

Учащийся, реализуя решение задачи с использованием технологии визуального проектирования, во-первых, разрабатывает макет (интерфейс) проекта, во-вторых, пишет обработчик событий (ряд процедур, реализующих последовательность выполнения тех или иных действий). При создании макета школьник выполняет некоторый *конечный набор* операций: выбор и размещение на форме нужного компонента (объекта), изменение его свойств (названия, цвета, размера, положения, шрифта и т. п.) и др. Следовательно, можно выделить типичные действия учащихся при создании интерфейса проекта и определить их временные характеристики. *Время* – это одна из первичных характеристик величины, которую можно измерить. Процедура – это последовательность операций, необходимых для выполнения некоторого события. Можно определить сложность для каждой процедуры и суммарную сложность задачи как сумму сложностей отдельных процедур. Определим *сложность* алгоритма в исполняемых компьютером командах: количество

арифметических операций, количество сравнений, вызовов стандартных процедур и функций, пересылки.

Остановимся подробнее на *технологии* определения и использования *временной* характеристики и *сложности* алгоритма решения задачи для оценки уровня знаний и умений учащихся.

Оценка временных затрат на выполнение типичных действий учащегося при составлении программ.

1. Нами выделены отдельные действия учащихся при создании макета проекта (табл. 1).

2. Опытным путем определены временные характеристики каждого отдельного действия. Полученная выборка была обработана исходя из закона логарифмически нормального распределения. Гипотеза о логарифмически нормальной модели распределения анализируемой эмпирической совокупности проверялась по λ-критерию Колмогорова. Полученные значения λ и вероятности P_λ допустимости расхождения оказались достаточно высоки, что позволило принять гипотезу. Исходя из логарифмически нормального распределения были оценены математическое ожидание, среднее значение и стандартное отклонение.

3. Вычислено общее время работы над одним проектом (табл. 2), что позволило рассчитать теоретическую временную характеристику каждого проекта – $T1_{теор}$ (время работы над проектом с установкой на скорость) и $T2_{теор}$ (время работы над проектом с установкой на безошибочность). Среднее время выполнения отдельного действия было получено при установке работы учащегося на скорость. Для вычисления данных, характеризующих временные затраты школьника при работе с установкой на безошибочность деятельности, мы использовали формулы пересчета [2. С. 116]:

$$\bar{\tau}_{iб} \approx 2{,}3\bar{\tau}_{iс};\sigma^2_{iб} \approx 2{,}3\sigma^2_{iс} + 0{,}5\bar{\tau}^2_{iс},$$

где $\bar{\tau}_{iб}, \sigma^2_{iб}$ – соответственно среднее значение и дисперсия времен выполнения *i*-го действия при установке на безошибочность; $\bar{\tau}_{iс}, \sigma^2_{iс}$ – то же, при установке на скорость работы. Кроме того, при определении времени решения задачи учащимся были учтены взаимные влияния действий: если однотипные действия выполняются в виде серии (например, последовательное нажатие кнопок), то время выполнения первого действия берется табличное, а время всех последующих действий уменьшается на 40 %. Учитывая вышесказанное, были рассчитаны $T1_{теор}$ и $T2_{теор}$.

Таблица 1

Теоретический расчет затрат времени на создание интерфейса проекта с использованием экспериментальных данных по затратам времени на выполнение действий

№	Действие	Среднее время, с	Первый проект		Второй проект		Третий проект		Четвертый проект	
			Колич.	Время, с	Колич.	Время, с	Колич.	Время, с	Колич.	Время, с
1	**Создание объекта**	3	8	15	13	24	15	27	21	38
2	*Название (заголовок)*	10	9	55	12	72	16	95	22	129
3	*Изменение цвета объекта*	4	1	4		0		0		0
4	*Задание стиля*	6	8	33		0		0		0
5	*Изменение размеров, перемещение*	8		0	14	68	5	26	10	50
6	*Изменение значения переключателя (вкл/выкл)*	5		0	1	5		0		0
7	*Изменение шрифта*	12	8	65	5	42	7	57		0
Итого:	*Время на скорость создания проекта ($T1_{теор}$)*			**172**		**212**		**206**		**217**
	Время на безошибочность создания проекта ($T2_{теор}$)			**395**		**487**		**473**		**498**

4. Определена первичная временная характеристика – время работы над проектом ($T_{эксп}$). Полученная выборка также была обработана исходя из закона логарифмически нормального распределения. Вычислено математическое ожидание (M_o) и среднеквадратичное отклонение (σ). Исходя из затраченного учащимися времени были выставлены баллы (от 1 до 11) за каждую задачу (табл. 2). Для этого полученный временной диапазон был разбит на 11 равных интервалов, исходя из полученных M_o и σ, каждому из которых приписывался соответствующий балл.

Таблица 2

Таблица распределения баллов и оценок учащихся за проекты

Ученик:	1	2	3	4	5	6	7	8	9	10	11	12	13
Первый проект	284	213	216	156	181	240	204	208	172	293	112	361	638
баллы:	5	7	7	9	9	7	8	8	9	5	11	3	1
Второй проект	487	228	235	230	231	660	361	309	206	406	200	515	563
баллы:	1	10	9	10	10	1	5	7	11	4	11	1	1
Третий проект	342	330		304	368	780	334	280	331	325	215	219	293
баллы:	6	7		7	5	1	6	8	6	6	10	10	8
Четвертый проект	400	275	270	264	295	480	222	560	340	328	205	219	423
баллы:	4	8	8	8	7	1	10	1	6	6	10	10	3
итого:	16	32	24	34	31	10	29	24	32	21	42	24	13
оценка:	**уд**	**отл**	**хор**	**отл**	**отл**	**уд**	**отл**	**хор**	**отл**	**хор**	**отл**	**хор**	**уд**

5. Выставлены оценки учащимся исходя из суммы набранных баллов (табл. 2, рис. 1). Для определения интервалов оценивания мы исходили из того, что величины подчиняются логарифмически нормальному распределению. Нами выделены следующие интервалы: оценка «хорошо» ставится при попадании величины в диапазон $M_o \pm 3\sigma$, «отлично» – при превышении баллов $M_o + 3\sigma$, «удовлетворительно» – количество баллов меньше $M_o - 3\sigma$. Диаграмма распределения оценок учащихся по теме представлена на рис. 1. Нужно отметить, что определение интервалов должно зависеть от M_o и σ, а диапазоны выставления оценки могут быть выбраны другие.

Сравнение $T1_{теор}$ и $T2_{теор}$ (табл. 1) с $T_{эксп}$ (табл. 2) обнаруживает сходства и некоторые различия, объясняющиеся тем, что при вычислении $T_{теор}$ достаточно сложно учесть такие факторы, как время, затрачиваемое на вспомогательные действия (перенос руки, перемещение взгляда и т. п.), скрытое время решения логических задач и др. Несмотря на это полученные значения $T_{теор}$ можно использовать для выставления баллов учащимся за создание проекта без определения $T_{эксп}$.

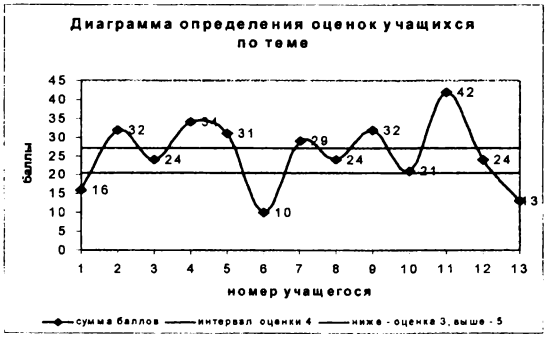


Рис. 1. Определение оценок учащихся

Определение сложности алгоритма. Оценка *сложности* производится на основе анализа текста алгоритма. Для определенности будем считать, что алгоритм записан на универсальном языке программирования типа Паскаль (например, Object Pascal) и содержит явно записанные арифметические операции, операции сравнения и пересылки скалярных данных, управляющие конструкции циклов и выбора. Если в программе встречаются вызовы процедур, то вызов не рассматривается как одна операция: он вносит вклад в общую сумму на основе подсчета количества операций в вызываемой процедуре плюс собственно оператор вызова (с единичной сложностью) [1. С. 115–116].

С целью анализа алгоритм (программу) удобно представлять управляющим графом. *Управляющий граф* строится следующим образом.

Каждому оператору присваивания или вызову процедуры ставится в соответствие вершина графа (точка). Два последовательно записанных оператора (последовательно исполняемых) изображаются вершинами, соединенными стрелкой, указывающей порядок исполнения.

Последовательность из нескольких операторов присваивания или вызовов процедур изображается цепью и называется линейным участком (рис. 2 *а*).

Условный оператор изображается вершиной, соответствующей вычислению условия, и двумя выходящими дугами с приписанными им вариантами then, else (рис. 2 *б*). Если после then записан составной оператор (линейный участок), то в управляющем графе ему соответствует цепь. Условный оператор задает разветвление в управляющем графе, заканчивающееся пустой вершиной (без операций), помеченной точкой с запятой. Если в операторе часть else отсутствует, то нижняя ветвь включает пустую вершину. Подобный фрагмент в управляющем графе порождает оператор выбора case, но только разветвление может быть на большее число ветвей и выходящие дуги помечаются соответствующими константами (рис. 2 *в*). Операторы цикла изображаются в управляющем графе замкнутой последовательностью вершин – циклом (рис. 2 *г–д*).

Кроме этого, в управляющем графе вводится одна начальная вершина и одна или несколько заключительных вершин. Каждой вершине графа припишем число – количество *операций*, которые необходимо выполнить для исполнения *оператора* программы, связанного с этой *вершиной*. Это число назовем *весом вершины*. Вес пустой, начальной и заключительных вершин равен нулю. Если из вершины выходит две или более дуги, то каждой из них припишем условие, при выполнении которого вычислительный процесс пойдет в этом направлении. Эти условия будем называть *пометками дуг*. Таким образом, управляющий граф программы представляет собой *направленный взвешенный граф*.

Оценка сложности по управляющему графу. **Рассмотрим возможные варианты оценки сложности по управляющему графу:**

1. Управляющий граф представляет собой *линейный* участок. *Сложность* равна *сумме весов вершин*, принадлежащих линейному участку, и является константой, если на участке нет вершин – вызовов процедур. Если такие вершины есть, то их вес равен функциям сложности соответствующих процедур.

2. Управляющий граф содержит *разветвления*, но не содержит циклов. Это означает, что вычислительный процесс в зависимости от исходных данных может направиться по одной из конечного числа ветвей, начинающихся в начальной вершине и заканчивающихся в одной из конечных вершин. Расчет сложности зависит от того, рассматриваем мы худший (максимальное количество операций обработки), лучший (минимальное количество операций) или средний случай.

Для *худшего* случая нужно вычислить *вес* каждой *ветви* как *сумму весов* входящих в нее вершин, после чего следует найти *максимальный* из весов ветвей. Это и будет *сложностью* процедуры.

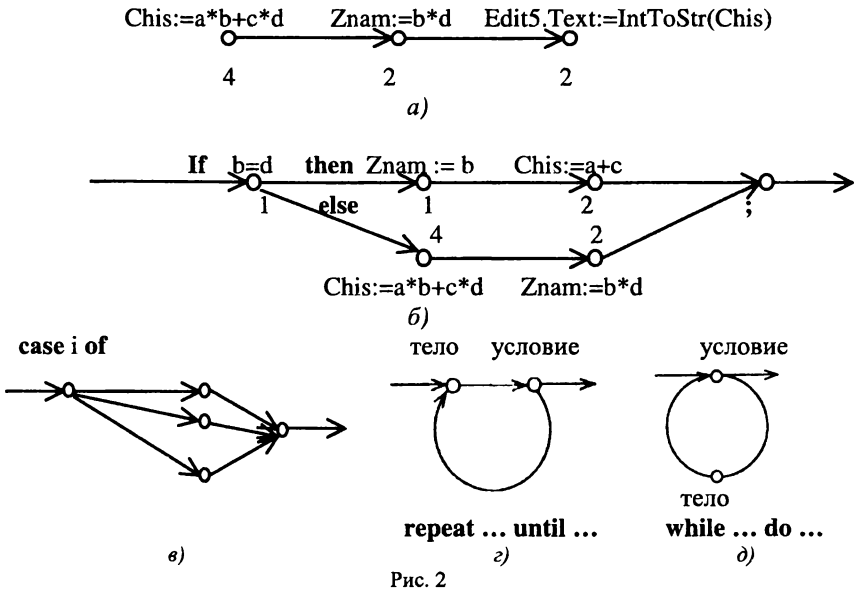


Рис. 2

Пример 1. Управляющий граф содержит 11 вершин с весами {1, 1, 1, 1, 1, 1, 2, 2, 4, 2, 2}. Вершина 5 (вес равен 1) соответствует вычислению условия в операторе *if*; вершины 6, 7 входят в часть *then*; вершины 8, 9 входят в часть *else*. Таким образом, имеется две ветви с весами 1+1+1+1+1+1+2+2+2 = 12 и 1+1+1+1+1+2+4+2+2 = 15; следовательно, сложность равна 15.

3. Управляющий граф содержит цикл, порожденный оператором *for*. Если выражения, определяющие начальное и конечное значения параметра цикла – константы, то нетрудно подсчитать, сколько раз будет выполняться тело цикла, и умножить на этот коэффициент вес тела цикла. Если выражения зависят от исходных данных, то можно оценить значение разности этих выражений в худшем или среднем случаях.

4. Управляющий граф содержит цикл, порожденный операторами *while* или *repeat*. Циклы *while* и *repeat* могут вызвать больше затруднений при анализе, чем оператор *for*, так как количество исполнений тела цикла зависит от истинности или ложности условия, а переменные, входящие в состав условия, изменяются в теле цикла, причем оценить величину и направленность этого изменения достаточно сложно [1. С. 117–121].

Для определения *сложности алгоритма* будем рассматривать худший случай, т. е. рассчитывать *максимальное* количество *необходимых операций* для получения результата. В программе анализируются только существенные

операции, т. е. которые влияют на исход решения задачи. Среда визуального проектирования Delphi позволяет выполнять большое количество операций, не являющихся необходимыми для решения конкретной задачи (например, менять цвет, шрифт, размеры объектов и т. п.). *Сложность задачи* определим как *сумму сложностей* входящих в нее подзадач (отдельных процедур).

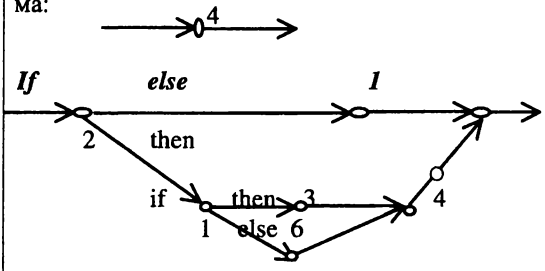
Предложенный подход позволяет

- классифицировать предлагаемые ученикам задачи на основе оценки сложности;
- дифференцировать поход к обучению программированию.

Разберем данный подход на примере, казалось бы, совершенно простой задачи: «Найти сумму двух обыкновенных дробей, заданных числителями и знаменателями, оценив при этом правильность ввода данных». При такой формулировке задачи в ней кроется множество подводных камней, открывается простор для развития и оценки мыслительных операций. Для оценки сложности нами выделены девять подзадач (это далеко не предел). По управляющему графу для каждого алгоритма мы получили множество из девяти элементов {14, 15, 16, 17, 18, 19, 20, 21, 27+2·V} (табл. 3). Параметр V (количество выполнений цикла While) получился при использовании оператора цикла с предусловием для определения НОД(M,N) и зависит от значений M и N (от модуля разности). Нами выделены три класса сложности задач. В первый класс попали алгоритмы со сложностью равной {14, 15, 16} – это линейные алгоритмы или алгоритмы с одним оператором ветвления. Во второй – {17, 18, 19, 20, 21} – разветвляющиеся с количеством ветвлений больших одного. В третий класс попал алгоритм с использованием цикла (сложность 27+2·V).

Таблица 3

Управляющий граф процедуры сложности 17

Procedure TForm1.Button1Click(Sender: TObject);
Begin
val(edit2.text,z1,cod1);
val(edit4.text,z2,cod2);
val(edit1.text,c1,cod3);
val(edit3.text,c2,cod4);
If (z1<>0) **and** (z2<>0) **then begin**
If z1=z2 **then begin** z3:=z1;
c3:=c1+c2; **end**
else begin z3:=z1*z2;
c3:=c1*z2+c2*z1; **end**;
edit5.Text:=inttostr(c3);
edit6.Text:=inttostr(z3);**end**
else edit5.Text:='Ошибка!!!'
end;
	Сложность = 17. Управляющий граф алгоритма:
	 

В предложенном подходе каждая задача оценивается двумя параметрами: количеством баллов за создание интерфейса проекта и оценкой сложности алгоритма. Разработанные критерии позволили нам более объективно подойти к оценке знаний и умений учащихся.

Литература

1. Миков А. И. Информатика. Введение в компьютерные науки: учеб. пособие для университетов по направлению «Прикладная математика и информатика» / Пер. ун-т. Пермь, 1998.

2. Справочник по инженерной психологии / Под ред. Б. Ф. Ломова. М.: Машиностроение, 1982.

3. Юсупов Р. М., Заболотский В. П. Научно-методологические основы информатизации. СПб.: Наука, 2000.

КОМПЬЮТЕРНАЯ СИСТЕМА ДЛЯ ОЦЕНКИ ТВОРЧЕСКИХ СПОСОБНОСТЕЙ НА ОСНОВЕ ТЕСТА ТОРРЕНСА

О. Г. Берестнева, И. С. Кострикина
Томский политехнический университет
Томский университет систем управления и радиоэлектроники

Актуальность задачи оценки творческих способностей школьников, абитуриентов и студентов не вызывает сомнений [1–3]. Авторами разработана компьютерная система для оценки творческих способностей учащихся на основе теста Торренса.

Основываясь на понимании функционирования интеллекта как интегрального процесса и учитывая влияние различных форм понятийного мышления на некоторые характеристики сенсорно-перцептивной сферы [1; 4], для компьютерной реализации нами был использован модифицированный вариант методики Торренса, предложенный М. А. Холодной [4]. В отличии от стандартного варианта методики [5] модифицированный вариант предоставляет более широкие возможности для оценки творческих способностей и особенностей познавательной деятельности учащихся. Например, имеется возможность исследовать особенности «эффекта когнитивной интегрированности» [6] и «продуктивность познавательной деятельности в условиях работы с перцептивными стимулами» [7].

Согласно описанию М. А. Холодной, процедура проведения обследования аналогична стандартному варианту методики Торренса в плане предъявления инструкции, сохранены графические стимулы. Многократное предъявление стимула до полного отказа испытуемого находить новые варианты завершения и называния стимула побуждало обследуемого постоянно преодолевать ригидность мышления и выдвигать разнообразные идеи. Модифицированный вариант объединяет диагностику интеллектуальных особенностей в вербальной и наглядно-образной сфере. Как и многие другие проективные