Impact Factor ISRA (India) = **1.344**
Impact Factor ISI (Dubai, UAE) = **0.829**
based on International Citation Report (ICR)
Impact Factor GIF (Australia) = **0.356**

Impact Factor JIF = **1.500**
Impact Factor SIS (USA) = **0.438**
Impact Factor РИНЦ (Russia) = **0.179**

SOI: 1.1/TAS DOI: 10.15863/TAS
International Scientific Journal
Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2015 **Issue:** 03 **Volume:** 23

Published: 30.03.2015 **http://T-Science.org**

Almir Anfirovich Miniakhmetov
Headmaster's assistant,
Teacher of Computer Science and Information Technologies
Askino Secondary School №1,
Republic of Bashkortostan, Russia
minalmir@yandex.ru

SECTION 21. Pedagogy. Psychology. Innovations in the field of education.

THE ROLE OF THE EDUCATIONAL UNIT «BRANCHING» WHILE TRAINING RURAL STUDENTS FOR THE USE ON IT

Abstract: *The article deals with the algorithmic construction «Branching», conditional and unconditional transitions, operations on them on Pascal. The author gives the differences of given transitions and possible program changes in complex tasks of the USE. Block-schemes of «Branching» equations and some mistakes are shown on Pascal. New methodical recommendations and practical methods are suggested for training rural students for the USE on IT.*

Key words: *branching, programming, language, Pascal, the USE, IT, algorithm, rural school, informatics, technologies, block-schemes, conditional, unconditional, transition, operator, case, if-then, if-then-else.*

Language: *Russian*

Citation: Miniakhmetov AA (2015) THE ROLE OF THE EDUCATIONAL UNIT «BRANCHING» WHILE TRAINING RURAL STUDENTS FOR THE USE ON IT. ISJ Theoretical & Applied Science 03 (23): 63-68.

Soi: http://s-o-i.org/1.1/TAS*03(23)13 ***Doi***: crossref http://dx.doi.org/10.15863/TAS.2015.03.23.13

УДК 378.14

РОЛЬ УЧЕБНОГО БЛОКА «ВЕТВЛЕНИЕ» ПРИ ПОДГОТОВКЕ ВЫПУСКНИКОВ СЕЛЬСКИХ ОБЩЕОБРАЗОВАТЕЛЬНЫХ ШКОЛ К ЕГЭ ПО ИНФОРМАТИКЕ И ИКТ

Аннотация: *В статье рассмотрена алгоритмическая конструкция «Ветвление», условный и безусловный переходы, действия над ними на языке программирования Паскаль. Приведены отличия указанных переходов и возможные доработки программ в задачах высокого уровня сложности на примере демонстрационного варианта ЕГЭ по информатике. Показаны блок-схемы решения задач на ветвление и некоторые ошибки, допускаемые выпускниками при оформлении программ на Паскале. Учителям предложены новые методические рекомендации и практические приемы по подготовке обучающихся сельских общеобразовательных школ к ЕГЭ по информатике и ИКТ.*

Ключевые слова: *ветвление, программирование, язык, Паскаль, единый государственный экзамен, ИКТ, алгоритм, сельская школа, информатика, технологии, блок-схема, условный, безусловный, переход, оператор, выбор, если-то, если-то-иначе.*

Единый государственный экзамен (ЕГЭ) по любому школьному предмету представляет собой выпускной экзамен за курс среднего (полного) общего образования. Как известно, одно дело сдать экзамен в форме ЕГЭ в стенах родной школы, совершенно другое дело, когда ЕГЭ становится еще и вступительным экзаменом в учреждения высшего и среднего профессионального образования [1].

Как и любой экзамен, ЕГЭ по информатике и ИКТ – это довольно сложная форма контроля знаний, особенно для выпускников обычных сельских общеобразовательных школ.

Объясняется это тем, что в обычной сельской школе предмету информатика уделяется всего лишь один час в неделю (45 минут урока), и очень редко два часа в неделю (профильное направление). Как правило, этого недостаточно для качественной подготовки к такому серьезному экзамену, где зачастую, в периферийных и филиалах предмет информатика вообще

вводится лишь с 10 класса, и очень редко с 5 класса (школы районного центра).

Известно, что ЕГЭ по информатике ориентирован в первую очередь на выпускников профильных и специализированных школ, которые собираются продолжить дальнейшее образование в сферах, связанных с информатикой и информационными технологиями. В то время, как уровень подготовки к экзамену по информатике обычных сельских выпускников довольно низкий, по сравнению с городскими.

В связи с этим, целью данной работы является обзор оптимальных методов и приемов подготовки обучающихся сельских общеобразовательных школ к выпускному экзамену в форме ЕГЭ по учебному блоку «Ветвление» на примере языка программирования Паскаль и рекомендованного демонстрационного варианта ЕГЭ-2015 по информатике.

Согласно Федеральному институту педагогических измерений [2], ЕГЭ принято проводить с использованием специально подобранных задач стандартизированного типа, контрольно-измерительными материалами (КИМ).

В базовом курсе информатики и ИКТ [3] обучающимся предлагается лишь общее ознакомление с алгоритмической структурой «Ветвление». Причем рассматривается язык программирования Visual Basic 2005, в то время как реальные задания ЕГЭ рассчитаны на пять языков программирования – Си, Паскаль, Бейсик, Алгоритмический язык, с этого года новый язык программирования Питон.

В профильном курсе информатики и ИКТ [4] на алгоритмическую структуру «Ветвление» также уделено мало внимания, всего лишь один параграф с общими понятиями, также для языка программирования Visual Basic 2005.

Современные авторы предлагают свои методы и приемы по подготовке обучающихся к ЕГЭ по информатике и ИКТ.

Автор работы [5] предлагает на базе школьного курса, актуальные задачи, что действительно позволяет выпускникам закрепить один из алгоритмов основного вида «Ветвление». Решения всех задач приводятся на языке Паскаль. Однако, перечень рассматриваемых задач ограничен количеством, что не позволяет в полной мере подготовить выпускника к экзамену.

Другой автор [6] в своей работе предлагает обобщенную методику решения некоторых задач на тему «Ветвление» с показательными примерами построения программы на Паскале. Данная методика рассчитана для студентов вуза и обучающихся специализированных школ, а не общеобразовательных, что затрудняет восприятие материала на школьном уровне. Хотя для подготовленного выпускника даже обычной школы, задачи вполне понятны и решаемы.

Каждый экзаменуемый на реальном ЕГЭ по информатике познает свой уровень усвоения предусмотренных учебной программой знаний, умений и навыков. В этом году, в заданиях №11, 19, 20, 21, 24, 25, 27 демонстрационного варианта ЕГЭ по информатике и ИКТ [7], проверялось умение выпускником реализовывать рекурсивный алгоритм или программу конкретного разработчика с фиксированным набором команд, умение анализировать готовую программу, разбивая на циклы и ветвление, находить и исправлять ошибки в алгоритмах.

Видимо, по одной из этих причин некоторые современные учебники информатики [8] уже целенаправленно с восьмого класса вводят основные понятия ветвления.

Преподавателю необходимо довести до обучающихся, что алгоритмизация и программирование начинается с понятия алгоритма, которое по определению [9], есть заранее заданное разработчику, точное предписание совершить определенную последовательность действий для получения решения задачи за конечное число шагов.

Многие обучающиеся не могут понять, что алгоритм становится более наглядным, если правильно применять элементы блок-схем. Согласно [10], блок-схема позволяет выделить в алгоритме основные алгоритмические структуры (линейная, ветвление, выбор и цикл).

Поэтому, учителю важно показать, что если разработчиком алгоритма является человек, то он может по предложенной блок-схеме очень легко проследить выполнение алгоритма или фрагмента программы даже сложной задачи.

Обучающиеся сами поймут, что все показанные элементы блок-схемы, соединенные стрелками, четко указывают на последовательные шаги выполнения алгоритма.

Например, уже в задании №11 [7], предлагается рекурсивный алгоритм вычисления суммы всех чисел, напечатанных на экране при выполнении вызова F(1) предложенного фрагмента программы (язык Паскаль).

```
procedure F(n : integer);
begin
    writeln(n);
    if (n < 5) then
    begin
    F(n + 1);
    F(n + 3);
    end;
end.
```

На первый взгляд задание простое, однако, выпускник должен четко знать, что под рекурсией понимается специальный прием, который сводит исходную задачу к одной или нескольким более простым задачам такого же типа.

Поэтому, выпускника, необходимо научить правильно и грамотно определять рекурсию, что предполагает знание условия остановки рекурсии.

Как видно, при реализации условия *(n < 5)*, выполняется два рекурсивных вызова *F(n + 1)* и *F(n + 3)*. Самый простой прием решения задачи состоит в построении двоичного графа на ветвление, в узлах которой запишутся значения параметров при вызове функции.

Конечно, многие выпускники обходятся и без дерева, тогда необходимо уже учитывать, что при каждом последующем вызове с условием *(n < 5)*, происходит два рекурсивных вызова и сумма чисел, полученных при вызове функции *F(n)*, вычисляется методом «обратного хода».

В любом рекомендованном учебнике информатики и ИКТ, ветвление начинается с понятия условного оператора. Поэтому, учителю важно до изучения данной темы, обратить особое внимание на определения разветвляющегося алгоритма, условия и конструкцию условного оператора в виде блок-схемы.

Следует просто и доступно довести до обучающихся, что разветвляющимся называется такой алгоритм, в котором в зависимости от истинности или ложности некоторого условия выбирается один из нескольких возможных путей продолжения алгоритма или программы.

Многие школьники уже с ранних курсов математики знают, что под условием понимается отношение между двумя величинами (=, <>, >, <, >=, <=).

Известно, что для демонстрации процесса ветвления применяется обозначение элемента блок-схемы в виде геометрической фигуры ромба. Необходимо акцентировать внимание обучающихся на том, что ромб служит для обозначения условий в алгоритмических структурах «Ветвление» и «Выбор», который имеет строго один вход сверху и строго два выхода по бокам.

В информатике принято считать, что выход налево от ромба реализуется, если проверяемое условие истинно, и направо при ложном условии. Таким образом, важным является умение обучающегося графически представлять процесс ветвления для решения задачи.

Далее учителю важно разъяснить тот факт, что в случае истинности проверяемого условия реализуется строго одна последовательность команд *(Действия 1)*, в случае ложности, соответственно, совершенно другая последовательность команд *(Действия 2)*.

Выпускники часто забывают, что в зависимости от результата проверки условия («Да» или «Нет») осуществляется выбор одного из альтернативных путей работы алгоритма или фрагмента программы.

Здесь учитель должен на конкретных примерах показать обучающимся, что каждый из путей ветвления ведет к общему выходу. Поэтому, работа алгоритма будет продолжаться независимо от того, какой из путей ветвления будет выбран.

Как показывает опыт работы по подготовке обучающихся к ЕГЭ, в основном реальные задания экзамена содержат только две конструкции структуры «Ветвление» из четырех возможных. Это полная условная конструкция *Если-То-Иначе (If-Then-Else)* и неполная условная конструкция *Если-То (If-Then)*.

Можно сказать, что условный оператор является средством ветвления вычислительного процесса, представляющий собой полную условную конструкцию [11].

Обучающиеся должны понять, что условный оператор позволяет проверить некоторое заданное условие, и в зависимости от результата проверки выполнить то или иное действие.

Преподавателю необходимо полностью раскрыть структуру условного оператора, например, имеющего следующий вид на языке программирования Паскаль *If <условие> Then <оператор1> Else <оператор2>*. В данной структуре *If*, *Then*, *Else* являются зарезервированными словами, *<условие>* – это произвольное выражение логического типа, *<оператор1>* и *<оператор2>* – любые операторы языка Паскаль.

Многие выпускники неправильно понимают механизм работы условного оператора. Здесь учителю важно объяснить, что вначале вычисляется условное выражение *<условие>*.

Если результатом проверки условия является *True (истина)*, то выполняется *<оператор1>*, а действие *<оператор2>* пропускается.

Если же, наоборот, результат проверки условия есть *False (ложь)*, то *<оператор1>* не реализуется и пропускается, а выполняется *<оператор2>*.

Данный механизм четко прослеживается в задании №24 демонстрационного варианта ЕГЭ [7]. Требуется найти все ошибки в предложенной программе и исправить их, выписав строки с ошибками.

Приведем лишь правильный фрагмент программы, показывающий обучающимся механизм проверки количества нечетных чисел в исходной последовательности и максимальное нечетное число ветвлением.

```
if (x mod 2 <> 0) then
begin
    count := count + 1;
    if (x > maximum) then maximum := x;
end;
if (count > 0) then
begin
    writeln(count);
    writeln(maximum);
end
else
writeln('No');
```

Анализируя фрагмент программы, обучающийся в результате ручной прокрутки поймет, что переменная *maximum* получит значение переменной *x*, если только это значение не превышает *maximum*, иначе *maximum* станет равным изначально заданному значению еще задолго до проверки условия.

Конечно, часть конструкции *Else* <*onepamop2*> условного оператора может быть и опущена, но правильно и грамотно.

Тогда данная структура будет считаться неполной условной конструкцией.

В этом случае, надо довести до обучающихся, что при значении *True (истина)* условного выражения выполняется <*onepamop1*>, а <*onepamop2*> пропускается.

Простым примером, демонстрирующим возможность реализации ветвления без части конструкции *Else* <*onepamop2*>, является стандартная задача нахождения максимального (минимального) из двух заданных чисел.

В случае полной условной конструкции фрагмент программы на Паскале будет выглядеть следующим образом:

```
if (a > b) then max := a else max := b;
```

Если переписать данный фрагмент для случая неполной условной конструкции, то строка программы примет следующий вид:

```
if (a > b) then max := a;
if (a < b) then max := b;
```

Однако, есть и сложность другого характера, который заключаются в том, что выпускники не могут полностью разобраться в случае «вложенных» условных операторов.

Из недавней работы по анализу типов данных [12] следует, что любой из операторов <*onepamop1*> и <*onepamop2*> может быть любого типа, в том числе и условным.

В то же время не каждый из «вложенных» условных операторов может содержать в себе часть *Else* <*onepamop2*>, что может привести к неправильной трактовке и пониманию выпускником предложенных для проверки условий.

Учителю необходимо показать, что в сокращенной форме условного оператора ключевое слово *Else* полностью отсутствует. Это означает, что любая встретившаяся часть *Else* будет соответствовать ближайшей к ней «сверху» части *Then* условного оператора.

Причем, если в качестве оператора должна выполняться серия операторов, то они должны объединяться в операторные скобки (так называемые «внутренние кавычки») *Begin-End*.

Демонстрацию работы ветвления без *Else* <*onepamop2*> можно наглядно увидеть в задании №25 демонстрационного варианта ЕГЭ того же источника [7]. Здесь требуется привести фрагмент программы, позволяющий найти и вывести количество пар элементов массива, сумма которых нечетна и положительна.

Данный механизм реализуется в короткой версии условного оператора, который приведем в виде фрагмента программы на Паскале.

```
k := 0;
for i := 1 to (n – 1) do
if ((a[i] + a[i + 1]) mod 2 <> 0)
and (a[i] + a[i + 1] > 0) then k := k + 1;
writeln(k);
```

Как видим, данный тип задания рассчитан на операции с элементами массива. Поэтому, при подготовке обучающихся к экзамену, учителю необходимо научить организовывать проверку соответствия элементов заданного массива некоторому условию, например, нахождению минимального четного элемента в массиве, нахождения количества и суммы всех четных элементов в массиве и т.д.

Особую сложность у экзаменуемых вызывает задание №27 указанного демонстрационного варианта ЕГЭ источника [7]. Задания подобного типа рассчитаны на реализацию сложного алгоритма с обязательным использованием современных систем программирования.

То есть выпускник на экзамене должен уметь создавать собственную программу в пределах 50 строк для решения задачи повышенной сложности, и очень редко средней сложности.

По условию задачи, требуется найти в заданной серии показаний некоторого прибора минимальное произведение двух показаний, между моментами передачи которых прошло определенное время (тип строго *Integer*).

Здесь важно учителю при подготовке разбить структуру программы на отдельные

части, уделяя немаловажное значение ветвлению. Приведем лишь фрагмент программы, эффективный по времени и памяти, содержащий элементы ветвления для указанного задания на языке программирования Паскаль.

if (a[i mod s] < min) then min := a[i mod s];
*if (a * min < ms) then ms := a * min;*

При анализе указанного фрагмента и всей структуры программы, необходимо обучающимся показать важность организации правильного поиска максимального (минимального) значения. То есть нужно грамотно показать работу с массивом максимумов (минимумов) из указанной границы элементов.

Иногда в подобных задачах ветвление помогает четко определить второе по величине (второе максимальное или второе минимальное) значение в данном массиве за однократный просмотр массива.

Несомненно, высокому результату экзамена по информатике способствует знакомство со структурой представленной работы, представление о возможных типах и форматах заданий реального ЕГЭ на ту или иную тему.

Поэтому учителю важно подготовить своих подопечных к экзамену таким образом, чтобы при составлении программы или его фрагмента, обязательно учитывали указания для экспертной комиссии по проверке и оцениванию работы.

Для этого, уже в среднем звене, обучающихся необходимо подготавливать к предстоящим экзаменам, постепенно добавляя новые знания и устраняя пробелы.

Необходимо обучать записывать простые условия в виде простых равенств или неравенств. Очень хорошо подходят для этого разного уровня задачи на программирование по теме «Ветвление».

Например, задачи разного уровня сложности [13] следующего содержания.

Задача 1. Запишите условный оператор, в котором значение переменной вычисляется по формуле: *(a + b)*, если *a* – нечетное число и *(a * b)*, если *a* – четное число.

Задача 2. Вывести на экран большее и меньшее из двух чисел *a* и *b*, введенных с клавиатуры.

Постепенно переходя к сложным условиям, учителю необходимо довести до школьников, что все сложные условия составляются и записываются из простых условий, с помощью логических операций [14]. Логические операции могут быть *And*, *Or*, *Xor*, *Not*. Особенно следует уделить внимание детей на решение неравенств.

Приведем примеры задач такого уровня, опираясь на источник [13].

Задача 3. Написать программу, проверяющую, принадлежит ли число, введенное с клавиатуры, интервалу *(0, 5)*.

Задача 4. Найти наибольшее (наименьшее) из трех данных чисел, введенных с клавиатуры.

Задача 5. Вывести на экран номер четверти, которой принадлежит точка с координатами *(x, y)*, при условии, что *x* и *y* отличны от нуля.

Задача 6. Составьте программу нахождения произведения двух наибольших (наименьших) из трех введенных с клавиатуры чисел.

Как видим, уровень заданий увеличивается от простого к сложному. Для проверки достижения высокого уровня подготовки, учителю необходимо постепенно вводить задачи повышенной сложности, чтобы переходя в старшее звено, обучающийся был готов начать окончательную подготовку к ЕГЭ по информатике.

Приведем некоторые задачи более сложные, чем были предложены выше, требующие наиболее полный и развернутый ответ.

Задача 7. Составьте программу, которая определяла бы вид треугольника, если данные отрезки позволяют его построить.

Задача 8. Составьте программу вычисления сложного выражения: *{max(x + y + z, xyz) + 3; min(x^2 + y^2, y^2 + z^2] – 4}*, если *x*, *y*, *z* введены с клавиатуры.

Задача 9. Составьте программу, которая из трех введенных с клавиатуры чисел возводит в квадрат положительные, а отрицательные оставляет без изменения.

В старшем звене (10-11 классы) обучающиеся выбирают тот или иной профиль обучения. Значит и требования при подготовке к ЕГЭ по информатике со стороны учителя также плавно возрастают в течение последующих двух лет обучения.

Приведем лишь фрагмент программы, посильный для решения старшекласснику, который запрашивает произвольное десятичное целое число в интервале *0…15* и преобразует его в шестнадцатеричный вид.

```
if (n >= 0) and (n <= 15) then
begin
if (n < 10) then rez := chr(ord('0') + n) else
rez := chr(ord('A') + n - 10);
end
else writeln('Ошибка');
```

Как видно, фрагмент программы четко показывает, что в шестнадцатеричной системе счисления используется ровно 16 цифр в каждом разряде, где цифры *0…9* обозначают первые десять возможных значений разряда, а буквы *A…F* оставшиеся шесть значений разряда.

Вполне возможно, что подобные задачи и не встретятся на реальном ЕГЭ, но база выпускника даже обычной сельской школы будет высокая, что позволит легче вести подготовку на основе предложенных заданий Федерального института педагогических измерений.

Важно, чтобы экзаменуемый четко представлял себе, что демонстрационный вариант ЕГЭ только показывает, какой вид задания может быть на этом месте в реальной работе. Это будет означать, что в реальном пакете КИМ будет совершенно другое задание, не идентичное демонстрационному (нулевому) варианту.

На основе опубликованных статей в данной области исследования, разрабатывается учебно-методический комплекс с новыми рекомендациями и практическими приемами по подготовке обучающихся средних общеобразовательных школ к выпускному экзамену в форме ЕГЭ по информатике.

Данное пособие будет также содержать задачи с решениями конкретной тематики, соответствующие реальным заданиям ЕГЭ.

References:

1. Leshhiner VR, Krylov SS, Jakushkin AP (2015) Optimal'nyj bank zadanij dlja podgotovki k EGJe 2015. Informatika. – Moscow: Intellekt-Centr, 2015. – pp.3-6.
2. (2015) Specifikacija kontrol'no izmeritel'nyh materialov dlja provedenija v 2015 godu edinogo gosu-darstvennogo jekzamena po informatike i IKT. Available: http://new.fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory/INF_SPEC_2015.pdf (Accessed 01.03.2015).
3. Ugrinovich ND (2005) Informatika i IKT. Bazovyj kurs. Uchebnik dlja 9 klassa. – Moscow: Binom. Laboratorija znanij, 2005. – pp.88-92.
4. Ugrinovich ND (2002) Informatika i informacionnye tehnologii. Uchebnik dlja 10-11 klassov. – Moscow: Laboratorija bazovyh znanij, 2002. – pp.151-153.
5. Pavlova IM (2012) Zanimatel'nye zadachi po teme «Bazovye algoritmicheskie struktury». / Informatika v shkole. – Moscow: «Obrazovanie i informatika», 2012. – №9. – pp.56-60.
6. Polevshhikov IS (2014) Metodika provedenija laboratornoj raboty po discipline «Informatika» na temu «Operatory vetvlenija» dlja studentov bakalavriata. Molodoj uchenyj. – Chita, 2014. – №4. – pp.107-110.
7. (2015) Demonstracionnyj variant EGJe 2015. Informatika i IKT, 11 klass. Available: http://new.fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory/INF_DEMO_2015.pdf (Accessed 01.03.2015).
8. Bosova LL, Bosova AJ (2013) Informatika. Uchebnik dlja 8 klassa. – Moscow: BINOM. Laboratorija znanij, 2013. – pp.76-81.
9. Evich LN, Kulabuhova SJ (2014) Informatika i IKT. Podgotovka k EGJe. Uchebno-metodicheskoe posobie. – Rostov-na-Donu: Legion, 2014. – pp.40-45.
10. Ugrinovich ND (2010) Informatika i IKT. Profil'nyj uroven'. Uchebnik dlja 10 klassa. – M.: Binom. Laboratorija znanij, 2010. – pp.190-194.
11. Faronov VV (2001) Turbo Paskal' 7.0. Nachal'nyj kurs. Uchebnoe posobie. – Moscow: Nolidzh, 2001. – pp.35-37.
12. Miniahmetov AA (2014) Rol' tipov dannyh pri podgotovke vypusknikov sel'skih shkol k EGJe po informatike i IKT. Nauchno-metodicheskij zhurnal «Sociosfera». – №4. – Razdel «Pedagogika». – Penza, 2014. – pp.96-99.
13. Chernov AA (2004) Konspekty urokov informatiki v 9-11 klassah. Praktikum po programmirovaniju. V pomoshh' prepodavatelju. – Volgograd: Uchitel', 2004. – pp.45-48.
14. Miniahmetov AA (2014) Rol' uchebnoj temy «Preobrazovanija tipov i dejstvija nad nimi» pri podgotovke vypusknikov sel'skih obshheobrazovatel'nyh shkol k EGJe po informatike. Ezhekvartal'nyj nauchnyj zhurnal «Azimut nauchnyh issledovanij: pedagogika i psihologija». – №4 (9). – Tol'jatti, 2014. – pp.84-87.