УДК 004.94

DOI: 10.18384-2310-7251-2019-2-106-121

АЛГОРИТМ ЗАЩИТЫ ОТ НАРУШЕНИЙ ПРАВИЛ ВВОДА ИНФОРМАЦИИ С КОРРЕКЦИЕЙ КОНЕЧНОГО РЕЗУЛЬТАТА

Шабанова А. В., Калашников Е. В.

Московский государственный областной университет

141014, Московская область, г. Мытищи, ул. Веры Волошиной, д. 24 Российская Федерация

Аннотация. Ввод неправильной информации в диалоге «пользователь-машина» может быть преднамеренным или непреднамеренным. В любом случае необходимо «научить» систему, обслуживающую машину, понимать и распознавать ошибки с вводом информации и исправлять их. Таким образом, цель работы состоит в разработке алгоритма защиты от нарушений ввода информации и коррекции конечного результата. Для этого изучаются уже существующие и традиционные алгоритмы. Выявляются их достоинства и недостатки по отношению к поставленной задаче. В частности, аналогичная ситуация с выявлением ошибки ввода, распознаванием и её коррекцией часто наблюдается в наиболее примитивных ситуациях – в калькуляторах перевода из одной системы счисления в другую. Поэтому в представленной работе рассматриваются и разрабатываются пути исправления основных ошибок, которые встречаются в онлайн-калькуляторах при переводе из одной системы счисления в другую, на основе уже существующих алгоритмов.

Ключевые слова: алгоритм, алгоритмизация, кодирование, программирование, система счисления, программа, валидация.

ALGORITHM OF PROTECTION AGAINST VIOLATIONS OF THE RULES OF INFORMATION INPUT AND CORRECTION OF THE FINAL RESULT

A. Shabanova, E. Kalashnikov

Moscow Region State University

ul. Very Voloshinoy 24, 141014 Mytishchi, Moscow Region, Russian Federation

Abstract. Entering of incorrect information in the user-machine dialog may be intentional or unintentional. In any case, it is necessary to 'train' the system, which serves the machine, to understand and to recognize errors with the information input and to correct them. Thus, the aim of the work is to develop an algorithm for protection against violations of information input and correction of the final result. For this purpose, existing and traditional algorithms are studied. Their advantages and disadvantages in relation to the task are revealed. In particular, a similar situation with the detection of the input error, its recognition and correction is often observed in the most primitive situations – in calculators translating one number system (NS) to another. This primitiveness, first of all, indicates the fundamental nature of the problem to be solved. Therefore, in the presented work we use the already existing algorithms to discuss and

develop the ways to correct the major errors that are found in Online Calculators translating one number system (NS) to another.

Keywords: *algorithm*, algorithmization, coding, programming, number system, program, validation.

Введение

Проблема неправильного ввода информации остро стоит перед пользователями. Существует множество ресурсов, готовых предоставить необходимую информацию по запросу пользователя, например Google¹. Но возможна ситуация, когда перед системой встает проблема неверного ввода информации со стороны неопытного или неаккуратного пользователя. В таком случае необходимо «научить» систему понимать ошибки и исправлять их.

Одним из типичных примеров такой системы, способной понимать, распознавать и исправлять ошибки, является поисковая система, которая перерабатывает некорректно введенные данные и предоставляет пользователю вариант правильного написания запроса в виде сообщения «Возможно, вы имели в виду ...». На смартфонах таким примером может послужить автоматическая замена некорректно введенного слова. Не все сервисы могут предоставить (оказать) подобного рода «помощь».

Ситуация с выявлением ошибки ввода, распознаванием и её коррекцией наблюдается на более примитивном уровне – в калькуляторах перевода из одной системы счисления в другую, например, перевода из двоичной системы счисления в восьмеричную. В представленной работе рассматриваются *пути исправления основных ошибок*, которые встречаются в онлайн-калькуляторах перевода системы счисления, на основе уже существующих алгоритмов.

Цель: построение и применение алгоритма защиты от нарушений правил ввода информации с коррекцией конечного результата на примере перевода из двоичной системы счисления в восьмеричную.

Сравнительный анализ алгоритмов защиты

Пользователь может вводить некорректные данные по разным причинам. На практике встречается множество ситуаций, когда определённые поля в пользовательском интерфейсе могут содержать только те данные, которые строго соответствуют определённым шаблонам. Если не учитывать подобные ситуации, то в системе могут возникать сбои, связанные с некорректным вводом пользователя. Ошибки могут быть допущены случайно или намеренно. Поэтому система должна производить предварительную проверку корректности введенных данных [1].

В работе [2] приведён алгоритм, похожий на реакцию поисковой системы на ошибки ввода. Для исправления опечаток в словах используются несколько алгоритмов.

- Расстояние Левенштейна – это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необ-

¹ Google [Электронный ресурс]. URL: <https://www.google.com> (дата обращения: 20.12.2018).

ходимых для превращения одной строки в другую. Используется для исправления ошибок в слове. Указанный алгоритм рассматривается подробно в работе [3].

- *Metaphone* – алгоритм индексирования слова по фонетическому принципу, работает только с буквами английского алфавита.

- Алгоритм Оливера, который вычисляет схожесть двух строк.

Также существует алгоритм, при котором возникает состояние ошибки. Состояние ошибки – это экран или окно, которые указывают пользователю на ошибки. Но этот алгоритм является неэффективным, так как оповещение приходит только после написания запроса. В таком случае на помощь приходит валидация [4], которая моментально информирует пользователя о корректности данного ответа сразу после того, как пользователь ввёл запрос. В статье [5] авторы подробно рассмотрели алгоритм валидации на примере трансформирования логических формул предикатов первого порядка. Существует ещё один алгоритм, не позволяющий вводить неверные символы в строку запроса. Он рассматривается в статье [4]. Алгоритм заключается в том, что накладывается ограничение ввода. Указанный метод подходит больше всего для реализации поставленной задачи. В таком случае, пользователь не сможет ввести цифры, не входящие в алфавит системы счисления.

В [6] указаны рекомендации для того, чтобы пользователь не совершил ошибку и валидация [7] не понадобилась.

Условно можно выделить две основные проблемы, с которыми встречаются калькуляторы перевода из одной системы счисления в другую: ошибки ввода и некорректного ответа

Ошибки ввода

Первая проблема. Основная проблема онлайн-калькуляторов перевода из одной системы счисления в другую заключается в том, что при неверном введении информации система не распознаёт ошибку и выдаёт неточный ответ. Например, сайт *Calculatori.ru*² пропускает неверные символы, но при нажатии кнопки «ПЕРЕВЕСТИ» (рис. 1) выдаёт рекомендации для ввода неопытному пользователю (рис. 2).

Введите число

Его система счисления	Перевести в
☒ Двоичная	☐ Двоичную
☐ Тройчная	☐ Тройчную
☐ Восьмеричная	☒ Восьмеричную
☐ Десятичная	☐ Десятичную
☐ Шестнадцатичная	☐ Шестнадцатичную
☐ Двоично-десятичная	☐ Двоично-десятичную
☐ Другая	☐ Другую

Рис. 1. Калькулятор № 1.

² ОНЛАЙН КАЛЬКУЛЯТОРЫ [Электронный ресурс]. URL: <http://calculatori.ru> (дата обращения: 20.12.2018).

Рис. 2. Реакция Калькулятора №1 на неверный ввод.

Сайт planetcalc.ru³ (рис. 3) также не реагирует на введение некорректной информации. При нажатии кнопки «РАССЧИТАТЬ» система не выдаёт ошибки. Вместо переведённого числа указан символ «?».

Рис. 3. Реакция Калькулятора №2 на неверный ввод.

Выдача некорректного ответа

Второй проблемой онлайн-калькуляторов является выдача некорректного ответа. Например, при вводе 000000001 в двоичной системе счисления, в ответе получается 001 в восьмеричной, что является некорректным результатом. Также, при вводе 000000, ответ выдаёт 00 в восьмеричной системе счисления⁴.

Введение некорректной информации

Существует множество алгоритмов защиты от введения некорректной информации.

Первым из традиционных примеров такого алгоритма является поисковая система Google. Если введённая информация становится непонятной для системы (рис. 4), то она предлагает возможные варианты пользователю (рис. 5) [3; 8].

³ Онлайн калькуляторы [Электронный ресурс]. URL: <https://planetcalc.ru/> (дата обращения: 20.12.2018).

⁴ См.: Перевод из CC 2 в CC 8 [Электронный ресурс] // Киберфорум : [сайт]. URL: cyberforum.ru/pascalabc/thread913916.html (дата обращения: 20.12.2018).

Рис. 4. Реакция поисковой системы на ввод неверных символов.

Рис. 5. Реакция поисковой системы на некорректный ввод.

Автозамена / T9 или форматированный ввод решается аналогично приведённому примеру (рис. 5) с реагированием браузера на некорректный ввод.

Выдача на экран ошибки

Второй алгоритм – это выдача на экран ошибки (рис. 6). Подобная ситуация встречается при неверном введении символов во время регистрации на сайтах. Например, сайт портала государственных услуг РФ⁵ имеет определённые требования для безопасности при создании пароля. Система не только не даёт возможности сохранить пароль, если он не соответствует выдвинутым критериям, но и выдаёт дополнительное окошко с невыполненными условиями.

Рис. 6. Реакция системы на несоответствие требованиям.

⁵ Портал государственных услуг Российской Федерации. URL: <https://www.gosuslugi.ru/> (дата обращения: 20.12.2018).

Игнорирование ненужных символов

Третьим способом защиты от некорректного ввода информации является игнорирование ненужных символов (рис. 7). Например, в поле для ввода даты рождения система не даёт вводить другие символы, кроме цифр⁶.

A screenshot of a web form. At the top, there is a label 'Дата рождения:' (Date of birth:). Below it is a text input field containing the date '13.01.1992'. The input field has a light gray border and a subtle shadow.

Рис. 7. Реакция системы на неверный ввод.

Таким образом, можно сделать вывод, что существует три основных способа проверки ввода информации [9]:

- предотвращение некорректного пользовательского ввода, то есть до ввода;
- пользователь может вводить всё, что хочет, но осуществляется проверка ввода на корректность; если проверка прошла успешно, то выполняется конвертация данных в переменную;
- пользователь может вводить всё, что хочет, но при операции извлечения данных оператором параллельно создаются исправления возможных ошибок.

Перевод из двоичной в восьмеричную систему счисления

Как отмечалось во введении (и во втором пункте), разработка алгоритмов защиты от ошибок затрагивает сами принципы построения таких алгоритмов и, по-видимому, носит фундаментальный характер. Поэтому рассмотрим построение алгоритма на примере перевода из одной системы счисления в другую. Для этого напомним простейшие операции перевода из двоичной в восьмеричную.

Система счисления [10] – знаковая система, в которой приняты определённые правила записи чисел. Знаки, с помощью которых записываются числа, называются цифрами. Их совокупность называется алфавитом, а их количество – основанием системы счисления [11].

Двоичная система счисления имеет основание – 2, состоит из двух символов, следовательно, её алфавит: 0,1 (рис. 8).

A diagram titled 'Двоичная СС' (Binary SN). It contains two boxes. The first box says 'алфавит: 0, 1' (alphabet: 0, 1). The second box says 'Форма записи: 010111_2' (writing form: 010111_2). The diagram is enclosed in a rectangular frame with a drop shadow.

Рис. 8. Двоичная система счисления.

⁶ См., например, сайт: Здравоохранение: электронная регистрация // Портал государственных услуг Российской Федерации. Московская область. URL: <https://uslugi.mosreg.ru/zdrav/> (дата обращения: 20.12.2018).

Восьмеричная система счисления имеет основание – 8, следовательно, состоит из восьми символов: 0 – 7 (рис. 9).

Рис. 9. Восьмеричная система счисления.

Существует несколько алгоритмов перевода из двоичной системы счисления в восьмеричную. Следует определить наиболее удобный из них для реализации поставленной цели. Наиболее распространённым является перевод из двоичной системы счисления в десятичную, а из десятичной в восьмеричную [12].

При переводе числа из системы счисления с основанием 2 в десятичную систему необходимо представить это число в виде суммы произведений степеней основания двоичной системы счисления на соответствующие цифры числа и выполнить арифметические вычисления. Формула перевода имеет вид [11; 13; 14]:

$$a_{n-1}a_{n-2} \dots a_1a_0 = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \dots + a_02^0, \quad (1)$$

где q – основание системы счисления, a – цифры двоичного числа, n – количество целых разрядов числа.

Например, если число 10001 перевести из двоичной системы счисления в десятичную, используя представленную формулу, получится 19:

$$10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0. \quad (2)$$

Далее необходимо перевести полученное число из десятичной системы счисления в восьмеричную. Для этого необходимо воспользоваться методом последовательного деления десятичного числа на 8, а затем выписыванием последнего частного и остатков в обратном порядке (рис. 10) [4; 10; 13].

Рис. 10. Перевод из 10-ой системы счисления в 8-ую.

Второй алгоритм напрямую преобразует двоичное число в восьмеричное.

Для реализации перевода из двоичной системы счисления в восьмеричную воспользуемся традиционной таблицей систем счисления, см. табл. 1 [10; 11].

- разделить число на триады от младших разрядов к старшим (справа налево);
- заменить каждую триаду двоичных чисел соответствующей восьмеричной цифрой по таблице 1;
- недостающие до триады позиции заполнить незначащими нулями.

Таблица 1. Системы счисления

Основание системы счисления			
10	2	8	16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

Каждый разряд двоичного числа содержит 1 бит информации. Для записи восьмеричных чисел используются восемь цифр, то есть в каждом разряде числа возможны 8 вариантов записи. Решаем показательное уравнение: $8 = 2^i$. Так как $8 = 2^3$, то $i = 3$ бита [11], то есть каждый разряд восьмеричного числа содержит 3 бита информации.

Таким образом, для перевода целого двоичного числа в восьмеричное, его нужно разбить на группы по три цифры, справа налево, а затем преобразовать каждую группу в восьмеричную цифру. Если в последней, левой, группе окажется меньше трёх цифр, то необходимо её дополнить слева нулями.

Переведём таким способом двоичное число 101001_2 в восьмеричное:

$$101_2 \Rightarrow 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5 \quad (3)$$

$$001_2 \Rightarrow 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1 \quad (4)$$

Ответ: 51_8

Решение

Преобразование двоичной системы счисления в восьмеричную, удобную для использования в компьютере.

Проведённый анализ алгоритмов перевода из 2-ой системы счисления в 8-ую позволяет разработать обобщённый алгоритм этого перевода (рис. 11):

1) имеется двоичное число, которое требуется перевести в восьмеричную систему;

2) данное число необходимо разделить по три ячейки (триады) справа налево, так как $2^3 = 8$. При этом нужно принять во внимание то, что если в крайней триаде не хватает ячеек, то они дополняются нулями;

3) каждую триаду следует представить в виде чисел: 4, 2, 1;

Рис. 11. Представление ячеек.

4) следующий шаг зависит от выбранного метода.

Первый метод заключается в подстановке данных в формулу, как указано на рис. 12.

Рис. 12. Первый вариант перевода.

Второй метод позволяет быстро вывести ответ перевода, избегая расписывания формулы (рис. 13).

Алгоритм заключается в рассмотрении отдельных ячеек каждой триады. Если ячейка заполнена 1, то значение истинно, если 0, то ложно или пустое. Истинные значения суммируются, как показано на рисунке 13.

Третий метод является наиболее простым, но неудобным для восприятия. Заключается он в применении таблицы систем счисления в виде таблицы 1.

При составлении калькулятора систем счисления, будем использовать первый вариант, рис. 12, так как он является наиболее понятным для компьютера.

Рис. 13. Второй метод перевода.

Построение алгоритма защиты от неопытного пользователя на примере перевода из 2-ой в 8-ую систему счисления.

Для исправления основных недостатков калькуляторов перевода системы счисления разработан алгоритм, представленный на блок-схеме (рис. 14).

Защита от неопытного пользователя на примере перевода из 2-ой в 8-ую систему отсчёта.

После анализа статей по решению подобной проблемы на разных языках программирования [1; 2; 9; 14] был сделан вывод, что лучше предотвратить ошибки в начале ввода информации: ограничить допустимые символы (алфавит двоичной системы счисления).

Рис. 14. Блок-схема перевода из 2-ой в 8-ую системы счисления.

Изначально пользователь вводит цифры двоичного числа (переменную C). Программа во время ввода проверяет, соответствует ли символ алфавиту системы счисления. Если соответствует, то продолжает перевод, если нет, то не вводит символ, а возвращает в ввод. В PascalABC можно легко реализовать указанный алгоритм. Для этого необходимо ввести переменную символа и определить алфавит двоичной системы счисления (рис. 15).

```
if c in ['0','1'] then  
begin  
  write(c);  
  s2:=s2+c;  
end;
```

Рис. 15. Алгоритм проверки символов.

Для исправления второй ошибки калькулятора с коррекцией конечного результата необходимо произвести преобразование переменной из строкового типа в числовой, в нашем случае, – целочисленный. Таким образом, получается переменная типа «строка», в которой записаны цифры “0000000” или “0000132”. Чтобы избавиться от ненужных (лишних) символов, используется процедура перевода из строкового типа в числовой. При выполнении этой операции все “0”, будут убраны. Для этого необходимо воспользоваться функцией `val` для перевода из одного типа данных в другой (см. рис. 16).

```
val(s8,k,b);  
write('Число в 8 системе счисления: ', k);  
readln  
End. |
```

Рис. 16. Коррекция конечного результата.

Готовая программа указана на рис. 17. При введении 00000000000 программа выдает один ноль, что является корректным результатом (рис. 18). Также при введении 00000001 происходит коррекция конечного результата (рис. 19).

```

uses crt;
const trd:array[0..7] of string[3]=('000','001','010','011','100','101','110','111');
var s2, s8, m, v: string;
    i, j, z, k, b: integer;
    c: char;

Begin
clrscr;
TextColor(green);

s2:='';
writeln('Введите число в 2 системе счисления, и нажмите Enter:');
repeat
c:=readkey;

if c in ['0','1'] then
begin
write(c);
s2:=s2+c;
end;

if c=#13 then writeln;
until c=#13;
while length(s2) mod 3 <> 0 do s2:='0'+s2;
s8:='';
writeln('Делим введённое число на триады');

z:=0;
j:=0;
while s2 <> '' do

begin
z:=z+1;
j:=j+1;
for i:= 0 to 7 do
if copy(s2,1,3)= trd[i] then s8:=s8+chr(i+48);

m:= copy(s2,1,3);
v:= copy(s8,z,j);
writeln(copy(s2,1,3), '=',m[1], '**4', '+',m[2], '**2', '+',m[3], '**1', '=', v);
delete(s2,1,3);
end;

val(s8,k,b);
write('Число в 8 системе счисления: ', k);
readln
End.

```

Рис. 17. Полный текст программы.

■ Перевод из 2 в 8.exe

Введите число в 2 системе счисления, и нажмите Enter:
000000000000
Делим введённое число на триады
000=0*4+0*2+0*1=0
000=0*4+0*2+0*1=0
000=0*4+0*2+0*1=0
000=0*4+0*2+0*1=0
Число в 8 системе счисления: 0

Рис. 18. Коррекция конечного результата при введении 000000000000.

Рис. 19. Коррекция конечного результата при введении 00000001.

Таким образом, разработан алгоритм защиты от нарушений правил ввода информации с коррекцией конечного результата на примере калькулятора перевода из двоичной системы счисления в восьмеричную.

Выводы

Основной проблемой калькуляторов перевода из одной системы счисления в другую является то, что система допускает ошибки неверного ввода информации со стороны пользователя и отсутствие коррекции конечного результата. В результате анализа основных алгоритмов реакции разных систем на некорректный ввод был разработан алгоритм защиты от нарушений правил ввода информации с коррекцией конечного результата на примере калькулятора перевода из двоичной системы счисления в восьмеричную.

Статья поступила в редакцию 17.01.2019 г.

ЛИТЕРАТУРА

1. Chubатов R. Validation and Error Handling in AngularJS Applications [Электронный ресурс] // SteelKiwi : [сайт]. URL: <https://steelkiwi.com/blog/validation-error-handling-angularjs-applications/> (дата обращения: 20.12.2018).
2. Применение алгоритма нечетного поиска в PHP [Электронный ресурс] // habr : [сайт]. URL: <https://habr.com/post/115394/> (дата обращения: 20.12.2018).
3. Чувилин К. В. Эффективный алгоритм сравнения документов в формате LATEX // Компьютерные исследования и моделирование. 2015. Т. 7. № 2. С. 329–345.
4. Hariprasad P. Minimize Errors in Mobile App Forms Using Interaction Design Patterns [Электронный ресурс] // Medium : [сайт]. URL: <https://medium.com/ux-for-india/minimize-errors-in-mobile-app-forms-using-interaction-design-patterns-3e88211f633d> (дата обращения: 20.12.2018).
5. Popov N., Jebelean T. A Complete Method for Algorithm Validation [Электронный ресурс]. URL: http://www3.risc.jku.at/publications/download/risc_3915/PopJeb-AUTOMATHEO.pdf (дата обращения: 20.12.2018).
6. Laubheimer P. Preventing User Errors: Avoiding Unconscious Slips [Электронный ресурс] // Nielsen Norman Group : [сайт]. URL: <https://www.nngroup.com/articles/slips/> (дата обращения: 20.12.2018).
7. Заикин М. Ю., Долгополов Е. С., Обухова О. Л., Соловьев И. В. Технология предотвращения дублирования библиографических описаний в базе данных научных пу-

- бликаций БИАС ИПИ РАН // Системы и средства информатики. 2015. Т. 25. № 1. С. 168–185.
8. Сегалович И. В. Как работают поисковые системы // COLTA.RU. URL: <https://www.colta.ru/articles/specials/4070> (дата обращения: 20.12.2018).
 9. Ворон Ю. В. Урок №72. Обработка некорректного пользовательского ввода [Электронный ресурс] // Ravesli: программирование для начинающих : [сайт]. URL <https://ravesli.com/urok-72-obrabotka-nekorrektnogo-vvoda-cherez-std-cin/> (дата обращения: 20.12.2018).
 10. Kushwaha K. Number System and base conversions [Электронный ресурс] // GeeksforGeeks : [сайт]. URL: <https://www.geeksforgeeks.org/number-system-and-base-conversions/> (дата обращения: 20.12.2018).
 11. Бурдинский И. Н. Системы счисления и арифметика ЭВМ: учеб. пособие. Хабаровск: Издательство Тихоокеанского государственного университета, 2008. 79 с.
 12. Complete Description of Well-known Number Systems using Single Table / Latif S., Qayyum J., Lal M., Khan F. // International Journal of Electrical & Computer Sciences. 2011. Vol. 11. No. 3. P. 23–29.
 13. Parhami B. Number Representation and Computer Arithmetic // Encyclopedia of Information Systems. USA: Academic Press, 2001. P. 317–333.
 14. Хабибулин И. Ш. Программирование на языке высокого уровня. C/C++ СПб.: Санкт-Петербург, 2006. 512 с.

REFERENCES

1. Chubатов R. Validation and Error Handling in AngularJS Applications. In: *SteelKiwi*. Available at: <https://steelkiwi.com/blog/validation-error-handling-angularjs-applications/> (accessed: 20.12.2018).
2. [Application of fuzzy search algorithms in PHP]. In: *habr*. Available at: <https://habr.com/post/115394/> (accessed: 20.12.2018).
3. Chuvilin K. V. [An efficient algorithm for comparing LATEX documents]. In: *Komp'yuternye issledovaniya i modelirovanie* [Computer Research and Modeling], 2015, vol. 7, no. 2, pp. 329–345.
4. Hariprasad P. Minimize errors in mobile app forms using interaction design patterns. In: *Medium*. Available at: <https://medium.com/ux-for-india/minimize-errors-in-mobile-app-forms-using-interaction-design-patterns-3e88211f633d> (accessed: 20.12.2018).
5. Popov N., Jebelean T. A Complete method for algorithm validation. Available at: http://www3.risc.jku.at/publications/download/risc_3915/PopJeb-AUTOMATHEO.pdf (accessed: 20.12.2018).
6. Laubheimer P. Preventing user errors: avoiding unconscious slips. In: *Nielsen Norman Group*. Available at: <https://www.nngroup.com/articles/slips/> (accessed: 20.12.2018).
7. Zaikin M. Yu., Dolgoplov E. S., Obukhova O. L., Solov'ev I. V. [Technology for prevention of duplication of bibliographic descriptions in the scientific database BIAS IPI RAS]. In: *Sistemy i sredstva informatiki* [Systems and Means of Informatics], 2015, vol. 25, no. 1, pp. 168–185.
8. Segalovich I. V. [How search engines work]. In: *COLTA.RU*. Available at: <https://www.colta.ru/articles/specials/4070> (accessed: 20.12.2018).
9. Voron Yu. V. [Lesson number 72. Processing invalid user input]. In: *Ravesli: programming for beginners*. Available at: <https://ravesli.com/urok-72-obrabotka-nekorrektnogo-vvoda-cherez-std-cin/> (accessed: 20.12.2018).

10. Kushwaha K. Number system and base conversions. In: *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/number-system-and-base-conversions/> (accessed: 20.12.2018).
11. Burdinskii I. N. Sistemy schisleniya i arifmetika EVM [Number system and arithmetic of computers]. Khabarovsk, Pacific National University Publ., 2008. 79 p.
12. Latif S., Qayyum J., Lal M., Khan F. Complete description of well-known number systems using single table. In: *International Journal of Electrical & Computer Sciences*, 2011, vol. 11, no. 3, pp. 23–29.
13. Parhami B. Number representation and computer arithmetic. In: *Encyclopedia of Information Systems*. USA, Academic Press Publ., 2001. pp. 317–333.
14. Khabibulin I. Sh. *Programmirovaniye na yazyke vysokogo urovnya C/C++* [Programming in high level language C/C++]. St. Petersburg, Sankt-Peterburg Publ., 2006. 512 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Шабанова Анжела Владимировна – студент физико-математического факультета Московского государственного областного университета;
e-mail: anzhela.shabanova.97@mail.ru;

Калашников Евгений Владимирович – доктор физико-математических наук, профессор кафедры вычислительной математики и методики преподавания информатики Московского государственного областного университета;
e-mail: ekevkalashnikov1@gmail.com

INFORMATION ABOUT THE AUTHORS

Anjela V. Shabanova – student of Faculty of Physics and Mathematics, Moscow Region State University;
e-mail: anzhela.shabanova.97@mail.ru;

Evgenii V. Kalashnikov – Doctor in physical and mathematical sciences, Professor at the Department of Computational Mathematics and Informatics Teaching Methods, Moscow Region State University;
e-mail: ekevkalashnikov1@gmail.com

ПРАВИЛЬНАЯ ССЫЛКА НА СТАТЬЮ

Шабанова А. В., Калашников Е. В. Алгоритм защиты от нарушений правил ввода информации с коррекцией конечного результата // Вестник Московского государственного областного университета. Серия: Физика-математика. 2019. № 2. С. 106–121.
DOI: 10.18384-2310-7251-2019-2-106-121

FOR CITATION

Shabanova A. V., Kalashnikov E. V. Algorithm of protection against violations of the rules of information input and correction of the final result In: *Bulletin of Moscow Region State University. Series: Physics and Mathematics*, 2019, no. 2, pp. 106–121.
DOI: 10.18384-2310-7251-2019-2-106-121