Т. Б. Захарова,
Академия социального управления, г. Москва,

П. И. Ефимов,
средняя общеобразовательная школа № 20 имени Н. З. Бирюкова, г. Орехово-Зуево, Московская область

КОНТРОЛЬ ВХОДНЫХ ДАННЫХ КАК СОСТАВНАЯ ЧАСТЬ ФОРМИРОВАНИЯ КУЛЬТУРЫ ПРОГРАММИРОВАНИЯ ПРИ ИЗУЧЕНИИ РАЗДЕЛА «АЛГОРИТМИЗАЦИЯ»

Аннотация
В статье рассматриваются возможности повышения культуры программирования учащихся при изучении раздела «Алгоритмизация» курса информатики основной школы. Отмечаются проблемы, возникающие при традиционном изучении данного раздела. На примере задания из демонстрационного варианта ОГЭ 2017 года демонстрируется новый методический подход к решению подобных задач.
Ключевые слова: школьный курс информатики, алгоритмизация, программирование, методика обучения программированию.

Во многих исследованиях по методике обучения информатике их авторы анализируют, какую роль в школьном курсе информатики играет изучение программирования (этот вопрос неоднократно рассматривался и в работах одного из авторов данной статьи, см., например [2–4]). Очевидно, что в современных условиях информационного общества программирование — это важнейшая составляющая *профессиональной* деятельности специалистов многих областей. В то же время отдельные аспекты программирования становятся *общекультурными* составляющими жизни каждого человека.

Постоянно увеличивающееся количество интеллектуальных устройств требует от людей знания основ человеко-машинного интерфейса для коммуникации с ними. Человек волей-неволей вовлекается в процесс программирования данных устройств: он указывает последовательность приготовления пищи в мультиварке, устанавливает телевизор на запись передачи в определенное время, задает расписание будильника, прокладывает маршрут в навигаторе... Всё это наглядные примеры программирования поведения интеллектуальных устройств. Программирования, которое становится основой общения человека и машины в повседневной жизни.

Если же обратиться к профессиональной деятельности различных специалистов (не программистов!), то здесь мы видим использование программирования еще чаще и на более глубоком уровне. Так, применение скриптов в CAD-системах для автоматического построения чертежей, эскизов, трассировки схем, моделирования, а также в мультимедийных системах при расчете трехмерных сцен или при создании сценариев поведения объектов требует знания современных скриптовых языков программирования, таких как Lisp, Python и др. Разработка программ

Контактная информация

Захарова Татьяна Борисовна, доктор пед. наук, профессор, профессор кафедры информационно-коммуникационных технологий Академии социального управления, г. Москва; *адрес:* 129281, г. Москва, Староватутинский проезд, д. 8; *телефон:* (495) 472-32-08, доб. 149; *e-mail:* t_zakh@mail.ru

Ефимов Павел Игоревич, учитель информатики средней общеобразовательной школы № 20 имени Н. З. Бирюкова, г. Орехово-Зуево, Московская область; *адрес:* 142608, Московская область, г. Орехово-Зуево, ул. Бирюкова, д. 25; *телефон:* (496) 423-74-11; *e-mail:* paul1976@inbox.ru

T. B. Zakharova,
Academy of Public Administration, Moscow,
P. I. Efimov,
School 20, Orekhovo-Zuyevo, Moscow Region

CONTROL OF INPUT DATA AS A PART OF FORMING THE CULTURE OF PROGRAMMING IN THE STUDY OF THE THEME "ALGORITHMIZATION"

Abstract
The article considers the possibilities of increasing the students' culture of programming in the study of the theme "Algorithmization" of the informatics course in elementary school. The problems encountered in the traditional study of the theme are described. On the example of the task from the demonstration version of the exam 2017, a new methodical approach to solving such tasks is demonstrated.
Keywords: school informatics course, algorithmization, programming, teaching programming.

для станков с ЧПУ и иной микроконтроллерной техники может потребовать знания низкоуровневых языков программирования. В научной деятельности, особенно в области обработки больших данных и статистики, часто требуется знание таких языков программирования, как R и Mathematica.

Эти примеры с очевидностью указывают на то, что умение алгоритмизировать задачи и знание основ программирования становятся не только ключевыми условиями человеко-машинного общения, но и в целом успешности человека в современном мире.

В основной школе при изучении в курсе информатики раздела «Алгоритмизация» учащиеся получают широкий спектр знаний и навыков в области программирования:

- узнают понятия «формальный исполнитель», «система команд исполнителя», «алгоритм», «формальное исполнение алгоритма» и др.;
- знакомятся с исполнителями Черепаха, Кузнечик, Водолей, Чертежник и др.;
- получают первые навыки составления программ на алгоритмическом языке, а также на одном из высокоуровневых языков программирования.

Данный раздел, без сомнения, следует считать одним из основных в курсе информатики.

В то же время, на наш взгляд, в школе уделяется недостаточно внимания вопросам *культуры программирования*. При выполнении заданий обычно предполагается, что данные, поступающие на вход программы, — корректные, т. е. записаны в правильном формате и принадлежат допустимому диапазону значений. Такая ситуация наблюдается и в учебных задачах, и в задачах олимпиад по информатике, и в экзаменационных заданиях. Понятно, что в образовательном процессе мы в основном сосредоточены на формировании у обучающихся умения составить правильный алгоритм для решения задачи, а проверка допустимости входных данных — это отдельный алгоритм, который по сложности может быть не менее, а даже более сложным, чем решаемая учебная задача. Поэтому мы специально формулируем учебную ситуацию как некоторую идеальную модель, которая позволяет обучающемуся абстрагироваться от деталей реализации полного комплекса мер обработки информации. Однако при этом у учащегося создается ложное впечатление, что в программировании только такие ситуации и бывают и вопросы контроля входных данных имеют несущественное значение. Такой подход к программированию приводит к возникновению целого ряда ошибок при работе учебных программ.

Анализ дидактических аспектов формирования представлений обучающихся о контроле входных данных при изучении раздела «Алгоритмизация» курса информатики основной школы показывает, что целенаправленное изучение школьниками вопросов контроля входных данных способствует достижению новых образовательных результатов, предусмотренных требованиями ФГОС основного общего образования. В частности, при реализации содержания общего образования, соответствующего современному уровню развития науки, изучение этих вопросов помогает развитию мотивационных, операциональных (инструментальных) и когнитивных ресурсов личности.

Разработанный авторами методический подход к организации образовательного процесса по освоению школьниками основ программирования (в том числе вопросов контроля входных данных) делает акцент на общеобразовательном потенциале этого раздела курса информатики. Для реализации методики создан необходимый учебный материал (содержание обучения, учебные задачи), разработаны методические рекомендации.

Рассмотрим особенности методики на примере выполнения задания 20.2 демонстрационного варианта ОГЭ по информатике 2017 года.

Задание сформулировано следующим образом:

Напишите программу, которая в последовательности натуральных чисел определяет минимальное число, оканчивающееся на 4. Программа получает на вход количество чисел в последовательности, а затем сами числа. В последовательности всегда имеется число, оканчивающееся на 4. Количество чисел не превышает 1000. Введенные числа не превышают 30 000. Программа должна вывести одно число — минимальное число, оканчивающееся на 4.

Возможные варианты решения данной задачи представлены на рисунках 1–3.

```
Program Demo;
var
  N, a, i, m : integer;
begin
  m := 30004;
  readln(N);
  for i := 1 to N do
    begin
      readln(a);
      if (a mod 10 = 4) and (a < m) then
        m := a;
    end;
  writeln(m);
end.
```

Рис. 1. Решение задачи на языке Pascal

```
#include <stdio.h>

int main(void)
{
  int N, a, m, i;
  m = 30004;
  scanf("%d", &N);
  for (i=0; i<N; i++) {
    scanf("%d", &a);
    if ((a%10==4) && (a<m)) {
      m = a;
    };
  };
  printf("%d\n", m);
}
```

Рис. 2. Решение задачи на языке C

```
m = 30004
N = int(input())
for i in range(0,N):
  a = int(input())
  if (a%10==4) and (a<m):
    m = a
print(m)
```

Рис. 3. Решение задачи на языке Python 3

Очевидно, что эти варианты являются образцами потенциально опасного кода, приводящего к аварийному останову программ, что нарушает одно из свойств алгоритма — результативность. То есть программа завершается без достижения желаемого результата.

Здесь можно выделить ошибки двух типов:

- во-первых, в программах отсутствует контроль входных данных на принадлежность допустимому диапазону;
- во-вторых, в программах отсутствует проверка входных данных на корректность типов данных.

Подобные ошибки, если они возникают в реальных (неучебных) программах, могут стать источником ряда уязвимостей, которые могут позволить злоумышленникам произвести вторжение в систему или исказить результаты работы программы.

Такие ошибки создают условия для возникновения угрозы безопасности информации УБИ.100 «Угроза обхода некорректно настроенных механизмов аутентификации» (по классификации банка угроз безопасности информации, опубликованном на сайте ФСТЭК России), если они возникают в алгоритмах процедуры прохождения аутентификации [1].

Ошибки второго типа являются наиболее распространенными и потенциально наиболее опасными. При этом если ошибки первого типа еще могут рассматриваться на уроках информатики, то ошибки второго типа не рассматриваются совсем, хотя они приводят к возникновению угроз безопасности информации УБИ.118 «Угроза перехвата привилегированного процесса» [1].

Реакция программ на ввод некорректных данных представлена на рисунках 4–6.

На рисунке 4 программе на вход подано некорректное значение «w4» (программа ожидает числовые данные, а ей дали символьную строку). Программа не смогла обработать ситуацию и аварийно (т. е. неконтролируемо со стороны программиста) завершила свое выполнение, тем самым нарушив основные свойства алгоритма — результативность и определенность.

На рисунке 5 на вход программы был дан некорректный второй параметр из трех ожидаемых числовых значений («w4» вместо «14»). Выполнение

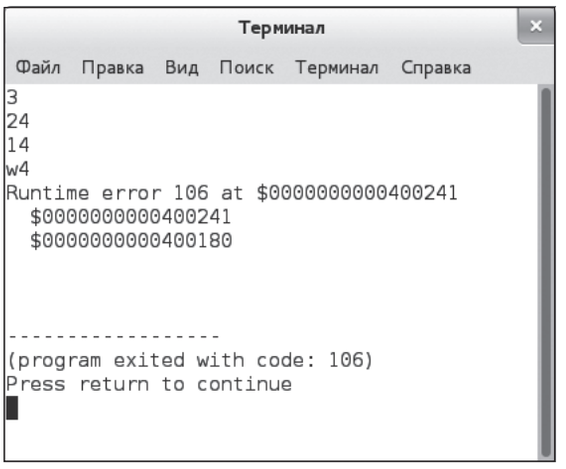

Рис. 4. Реакция на ввод некорректных данных программы на языке Pascal

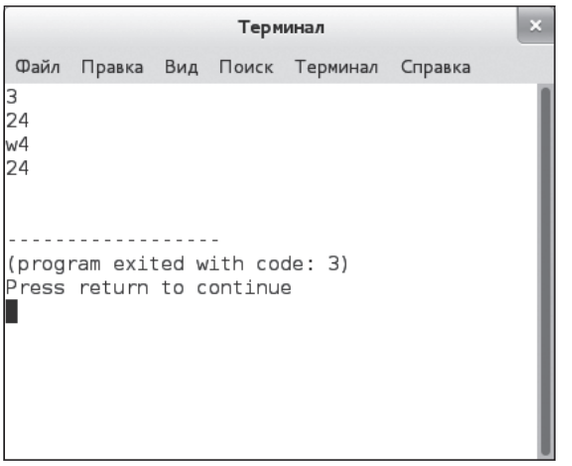

Рис. 5. Реакция на ввод некорректных данных программы на языке C

цикла аварийно завершилось, и программа выдала некорректный результат, не дожидаясь ввода всех данных. Данная ситуация усугубляется тем, что программа вроде бы продолжает работу и выводит некий результат. Это осложняет диагностику ошибки, а в случае использования в пакетном режиме — и сам факт обнаружения ошибки.

На рисунке 6 на вход программы подан некорректный параметр («w4» вместо «14»). Программа аварийно завершила свою работу аналогично программе на Pascal.

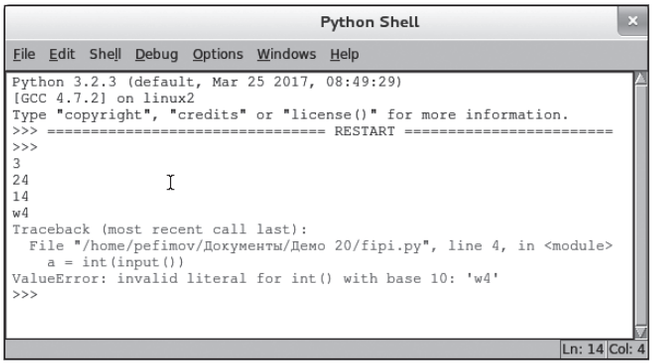

Рис. 6. Реакция на ввод некорректных данных программы на языке Python 3

Итак, в трех рассмотренных случаях программа преждевременно прерывала свою работу, причем в двух случаях (программы на Pascal и Python 3) создавалась потенциальная опасность перехвата управления программой, а в одном случае (программа на C) создавалась ситуация трудно отслеживаемого искажения результатов.

При этом небольшого изменения кода достаточно, чтобы программы стали более надежными и управляемыми. Примеры безопасных программ на языках Pascal, C и Python 3 приведены на рисунках 7–9.

Как можно видеть на рисунках 7–9, самой сложной модификации подверглась программа на языке C, а самой простой — программа на языке Python 3. Косвенно это может являться критерием выбора языка программирования для изучения в основной школе. Но вопрос о дидактических

```
Program Demo;
uses sysutils;

var
  S : string;
  N, a, i, m : integer;
begin
  m := 30004;
  readln(S);
  N := StrToIntDef(S, -1);
  if N>0 then
    begin
      for i := 1 to N do
        begin
          readln(S);
          a := StrToIntDef(S,-1);
          if a > 0 then
            begin
              if (a mod 10 = 4) and (a < m) then
                m := a;
            end
          else
            begin
              writeln('Ошибка! Введен неверный
                параметр в строке ', i);
              m := -1;
            end;
        end;
      if m > 0 then
        writeln(m)
      else
        writeln('Программа завершилась
        некорректно');
    end
  else
    writeln('Ошибка! Введен неверный параметр:
      количество чисел');
end.
```

Рис. 7. Пример безопасной программы на языке Pascal

принципах такого выбора не входит в тему данной статьи. Мы придерживаемся мнения, что для изучения программирования в основной школе наиболее целесообразно выбирать язык Pascal.

Разумеется, рассмотренная в статье задача ОГЭ относится к высокому уровню сложности для изучения в основной школе, а ее модификация с учетом контроля входных данных еще больше усложняет задачу. Можно утверждать, что изучение конкретных приемов безопасного программирования целесообразно разбирать в рамках внеурочных занятий или на консультациях. Но познакомить с потенциальными опасностями некорректного программирования, на наш взгляд, необходимо *всех* учащихся, так как это является не только аспетом повышения культуры программирования, но и может стать общекультурной составляющей подготовки молодого поколения к жизни в современных условиях.

Список использованных источников

1. Банк данных угроз безопасности информации // Федеральная служба по техническому и экспортному контролю. Государственный научно-исследовательский испытательный институт проблем технической защиты информации. http://bdu.fstec.ru/threat

2. *Захарова Т. Б., Захаров А. С.* Информатика как обязательный учебный предмет в системе общего образования // Наука и школа. 2015. № 5.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define L 10

int isalldigits(char*, int);

int main(void)
{
  int N, a, m, i;
  char S[L];
  m = 30004;
  fgets(S, L, stdin);
  if (isalldigits(S, L)) {
    N = atoi(S);
    for (i=0; i<N; i++) {
      fgets(S, L, stdin);
      if (isalldigits(S, L)){
        a = atoi(S);
        if ((a%10==4) && (a<m)) {
          m = a;
        };
      } else {
        printf("Указан неверный параметр
          в %d строке\n",i);
      };
    };
    printf("%d\n", m);
  } else {
    printf("Указано неверное количество
      строк\n");
  };
}

int isalldigits(char* S, int l)
{
  int i, r;
  r = 1;
  for(i=0; i<l && S[i]!='\0'&& S[i]!=0xA; i++){
    if (isdigit(S[i])==0){
      r = 0;
    };
  };
  return r;
}
```

Рис. 8. Пример безопасной программы на языке C

```
m = 30004
N = input()
if N.isdecimal():
  N = int(N)
  for i in range(0, N):
    a = input()
    if a.isdecimal():
      a = int(a)
      if (a%10==4) and (a<m):
        m = a
      else:
        print("Ошибка в строке ", i+1, "\n")
  print(m)
else:
  print("Ошибка в количестве строк")
```

Рис. 9. Пример безопасной программы на языке Python 3

3. *Кузнецов А. А., Захарова Т. Б.* Школьная информатика: вчера, сегодня, завтра // Информатика и образование. 2014. № 10.

4. *Кузнецов А. А., Захарова Т. Б., Захаров А. С.* Общая методика обучения информатике. М.: Прометей, 2016.