

DOI: 10.32517/2221-1993-2023-22-5-33-38

Н. И. Заводчикова*Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия;
Средняя школа № 76, г. Ярославль, Россия***И. А. Быкова***Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия;
Средняя школа № 26, г. Ярославль, Россия*

ОРГАНИЗАЦИЯ РАЗЛИЧНЫХ ВИДОВ ЗНАКОВО-СИМВОЛИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ ПРИ ОБУЧЕНИИ ШКОЛЬНИКОВ ПРОГРАММИРОВАНИЮ

Аннотация

Отдельные действия учащихся при решении задач по теме «Программирование» можно отнести к знаково-символической деятельности. При обучении программированию учителю необходимо обратить особое внимание на совмещение различных планов: действия с конкретными данными, знаково-символическое описание этих данных и структуры алгоритма, написание кода программы. Выделяют следующие виды знаково-символической деятельности: моделирование, схематизация, кодирование, замещение. Целесообразно на первых этапах обучения включать в процесс подготовки задания, в которых данные виды деятельности являются самостоятельными единицами, а не одним из шагов решения задачи по программированию. Деятельность моделирования реализуется с помощью упражнений на заполнение трассировочных таблиц. Для организации схематизации используются задания на вербальное описание алгоритма и построение блок-схем. Выделению замещаемого и заместителя способствуют задачи на установление соответствия между строками кода и их назначением. В статье приведены примеры заданий для организации указанных видов знаково-символической деятельности.

Ключевые слова: обучение программированию, знаково-символическая деятельность, семиотика, ориентировочная основа действия, набор задач.

Контактная информация

Заводчикова Надежда Ивановна, канд. пед. наук, доцент,

доцент кафедры теории и методики обучения информатике, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108/1;
учитель информатики, средняя школа № 76, г. Ярославль, Россия; *адрес:* 150031, Россия, г. Ярославль, ул. Автозаводская, д. 69д;
e-mail: zaw-nadejda@yandex.ru

Быкова Ирина Альбертовна,

старший преподаватель кафедры теории и методики обучения информатике, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108/1;
учитель информатики, средняя школа № 26, г. Ярославль, Россия; *адрес:* 150045, Россия, г. Ярославль, ул. Блюхера, д. 74;
e-mail: i.bukova@yandex.ru

N. I. Zavodchikova

Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia;
School 76, Yaroslavl, Russia

I. A. Bikova

Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia;
School 26, Yaroslavl, Russia

ORGANIZATION OF VARIOUS TYPES OF SIGN-SYMBOLIC ACTIVITY WHEN TEACHING SCHOOLCHILDREN PROGRAMMING

Abstract

Individual actions of students when solving problems on the theme "Programming" can be referred to sign-symbolic activity. When teaching programming, the teacher should pay special attention to combining various plans: actions with specific data, sign-symbolic description of this data and the algorithm structure, writing program code. The following types of sign-symbolic activity are distinguished: modeling, schematization, coding, substitution. It is advisable to include tasks in the preparation process at the first stages of training, in which these types of activity are independent units, and not one of the steps to solve the programming problem. It is advisable to implement modeling activity using trace table completion tasks. To organize schematization, tasks are used to describe the algorithm verbally and make flow charts. Distinguishing of the substitute and substituent is facilitated by tasks for matchmaking between code lines and their purpose. The article provides examples of tasks that contribute to sign-symbolic activity organization of various types in programming training.

Keywords: programming training, sign-symbolic activity, semiotics, orientation basis of action, set of tasks.

1. Введение

В ряде российских и зарубежных исследований отмечается **целесообразность использования идей и принципов семиотического подхода для повышения эффективности обучения программированию** (см., например, [4, 5, 8, 12]). В основе семиотического подхода лежит принцип целенаправленного развития знаково-символических действий обучающихся. В работах [8, 12] отмечается, что низкий уровень развития знаково-символических действий ведет к тому, что ученик не соотносит знак (например, операторы, командные слова языка программирования) с обозначаемым им объектом действительности, что приводит к формализму в усвоении знаний. Д. М. Гребнева указывает на необходимость обратить особое внимание на совмещение различных планов: действия с конкретными данными, знаково-символическое описание этих данных и структуры алгоритма, написание кода программы [4, 5].

Н. Г. Салмина выделяет следующие **виды знаково-символической деятельности** [9]:

- моделирование;
- схематизация;
- кодирование;
- замещение.

Моделирование предполагает получение новой информации в процессе оперирования знаково-символическими средствами. В **схематизации** знаково-символические средства выполняют ориентировочную роль, заключающуюся в структурировании реальности, выявлении связей между явлениями. В **кодировании** знаково-символические средства выполняют коммуникативную функцию сообщения информации в форме, понятной приемнику. **Замещение** — самый простой уровень знаково-символической деятельности, когда функции (признаки, свойства) замещаемого объекта переносятся на знаково-символическое средство [9, с. 80].

Целесообразно на первых этапах обучения включать в процесс подготовки задания, в которых данные виды деятельности являются самостоятельными единицами, а не одним из шагов решения задачи по программированию.

Рассмотрим типы заданий, которые можно использовать для явного выделения знаковой ситуации и ее аспектов, на примере упражнений для создания ориентировочной основы деятельности при решении следующей **задачи**:

Дана строка символов. Найти максимальную длину подстроки, состоящей из символов А.

2. Моделирование

Семиотический подход к обучению выделяет следующие **этапы моделирования** [9, с. 93]:

- 1) предварительный анализ задачи;
- 2) перевод на знаково-символический язык;
- 3) работа с моделью;
- 4) соотнесение результатов с реальностью.

Последние два этапа предполагают отладку и тестирование программы; проблема выделения этих этапов как отдельного элемента усвоения при обучении про-

граммированию неоднократно рассматривался в методической литературе. В данной статье приведем примеры заданий для первых двух этапов, так как они тоже имеют свой специфический операциональный состав, который должен выступать как предмет усвоения учащимися.

При анализе условий задачи происходят абстрагирование от конкретных данных и выделение структур данных, необходимых для решения. *Организовать одновременное восприятие отдельных компонентов задачной ситуации целесообразно с помощью **трассировочной таблицы**, в которой отражены изменения значения величин для конкретных входных данных.* Трассировочная таблица позволяет, как правило, перейти от оперирования конкретными данными к буквенно-цифровым обозначениям этих данных.

Задание 1.

Для данной строки $s = \text{'BACAACAACA'}$ заполните пропуски в таблице (табл. 1) на каждом шаге; в последней строке опишите правила изменения величин, указанных в первой строке.

Таблица 1

Таблица к заданию 1

Символ, который рассматривается на данном шаге	Длина подстроки, состоящей из символов А, заканчивающейся на рассматриваемом символе	Максимальная длина подстроки, состоящей из символов А, на данном шаге
В	0	0
А	1	1
С	0	1
А		
А		
...		
С		
А		
Последовательно принимает значение очередного символа исходной строки		

Трассировочная таблица, являясь моделью изменения значений величин, участвующих в решении задачи, выполняет функцию внешней опоры и выступает оружием для получения алгоритма. Введение переменных можно оформить в форме таблицы с описанием имен, типов и назначения переменных, используемых в решении (см. табл. 2). Подобные таблицы предлагалось заполнять учащимся при изучении «КуМира» [1].

Задание 2.

Выделите переменные, необходимые для решения задачи. Укажите их имена, типы и назначение в программе.

Таблица 2

Пример выполнения задания 2

Величина	Имя	Тип	Назначение
Исходная строка	<i>s</i>	str	аргумент
Максимальная длина подстроки, состоящей из символов А	<i>maxleng</i>	int	результат
Символ, который рассматривается на данном шаге	<i>x</i>	str	промежуточная величина
Длина подстроки, состоящей из символов А, заканчивающейся на рассматриваемом символе	<i>leng</i>	int	промежуточная величина

3. Схематизация

Схематизация — средство ориентировки в задаче, использование которого позволяет осуществить замещение действий с конкретными данными на уровне сущности [9, с. 80]. Схему решения задачи по программированию, как правило, оформляют с помощью вербального описания алгоритма, псевдокода или блок-схемы.

Как отмечает Н. Г. Салмина, выявление сущности и связей всякого рода значительно легче осуществляется при оперировании пространственно-графическими моделями [9, с. 99]. Псевдокод и вербальное описание алгоритма на первых этапах изучения программирования выглядят для ученика линейно, а блок-схема позволяет избавиться от линейности и сформировать правильный образ семантики алгоритма.

При решении указанной выше задачи можно использовать следующие задания, предполагающие выполнение действия схематизации.

Задание 3.

Заполните пропуски в описании алгоритма решения задачи:

Последовательно перебираем ... строки *s*:

- если текущий символ равен А, то ... счетчик *leng* на ..., в противном случае ...
- если ли текущее значение счетчика *leng* ... *maxleng*, заменяем значение ... на ...

Выводим значение ...

Задание 4.

Отметьте блок-схемы, соответствующие алгоритму решения задачи (см. рисунок на с. 36).

4. Кодирование

При кодировании происходит отделение формы от содержания, т. е. происходит выделение замещаемого и заместителя.

Например, в предложенной задаче:

- план замещаемой реальности: «*x* последовательно принимает значения символов строки *s*»;
- план ее выражения с помощью замещающих средств: «for *x* in *s*».

Целесообразно для совмещения планов замещаемого и замещающего использовать задания 5–8.

Задание 5.

Отметьте, какие команды позволят осуществить перебор всех символов строки *s*:

- for *x* in *s*:
- if *x* in *s*:
- for *i* in range(len(*s*)):
- for *s* in *x*:

Задание 6.

Отметьте, какая команда описывает правило, по которому происходит изменение значения длины текущей подстроки из символов А (переменной *leng*):

- if *a* in *s*:
- if *x*=='A':
 leng+=1
else:
 leng=0
- if *x*=='A':
 leng+=1
- if *x*!='A':
 leng=0

Задание 7.

Отметьте, какие команды описывает правило, по которому происходит изменение значения длины максимальной подстроки из символов А на данном этапе (переменной *maxleng*):

- maxleng*=max(*maxleng*, *leng*)
- if *maxleng*<*leng*:
 maxleng=*leng*
else:
 maxleng=0
- if *maxleng*<*leng*:
 maxleng=*leng*
- if *leng*<*maxleng*:
 leng=*maxleng*

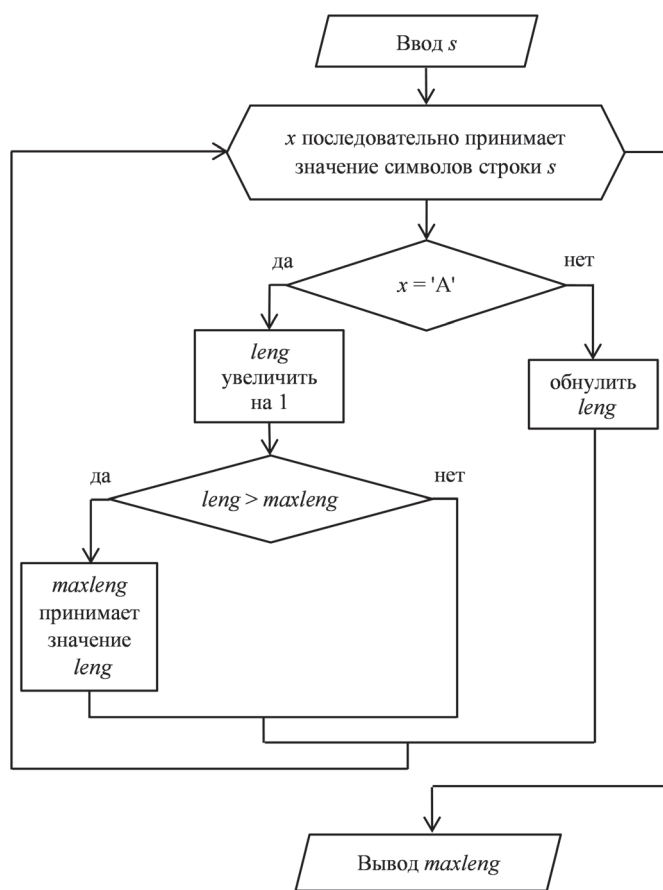
Задание 8.

Установите соответствие между строками кода программы для решения задачи и их назначением (табл. 3).

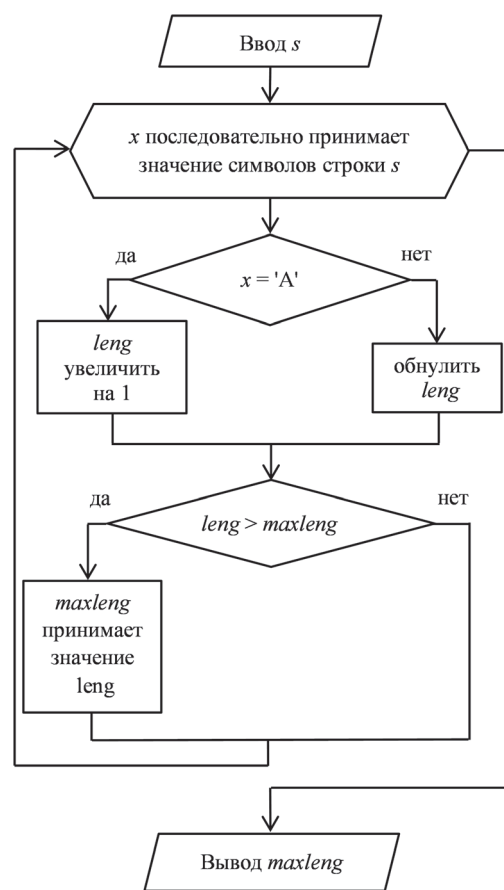
Таблица 3

Таблица к заданию 8

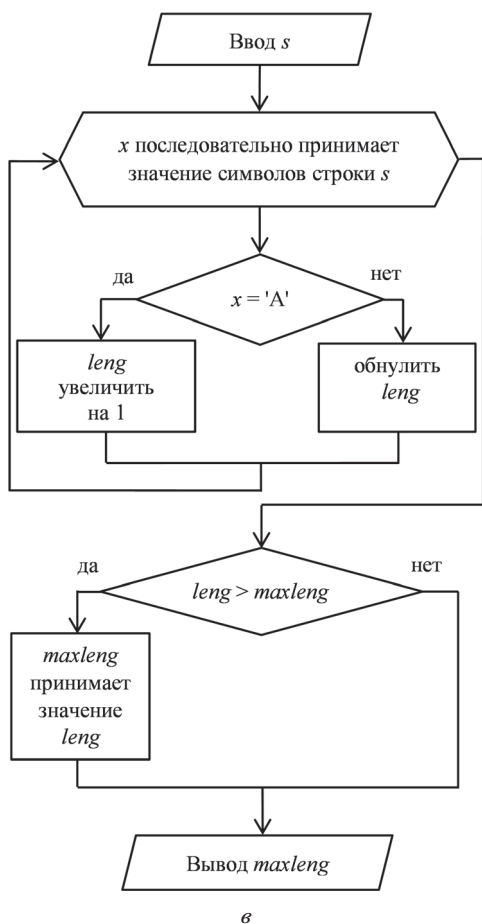
Назначение	Строки кода программы
Инициализация переменных	for <i>x</i> in <i>s</i> :
Организация последовательного перебора символов строки <i>s</i>	print (<i>maxleng</i>)
Изменение значения счетчика <i>leng</i> по правилу: если текущий символ равен А, то увеличиваем счетчик на единицу, в противном случае обнуляем его	<i>maxleng</i> = max(<i>maxleng</i> , <i>leng</i>)
Изменение значения максимальной длины подстроки, состоящей из символов А, по правилу: если текущее значение счетчика <i>leng</i> больше <i>maxleng</i> , заменяем значение <i>maxleng</i> на <i>leng</i>	if <i>x</i> =='A': <i>leng</i> +=1 else: <i>leng</i> =0
Вывод результата	<i>leng</i> =0 <i>maxleng</i> =0



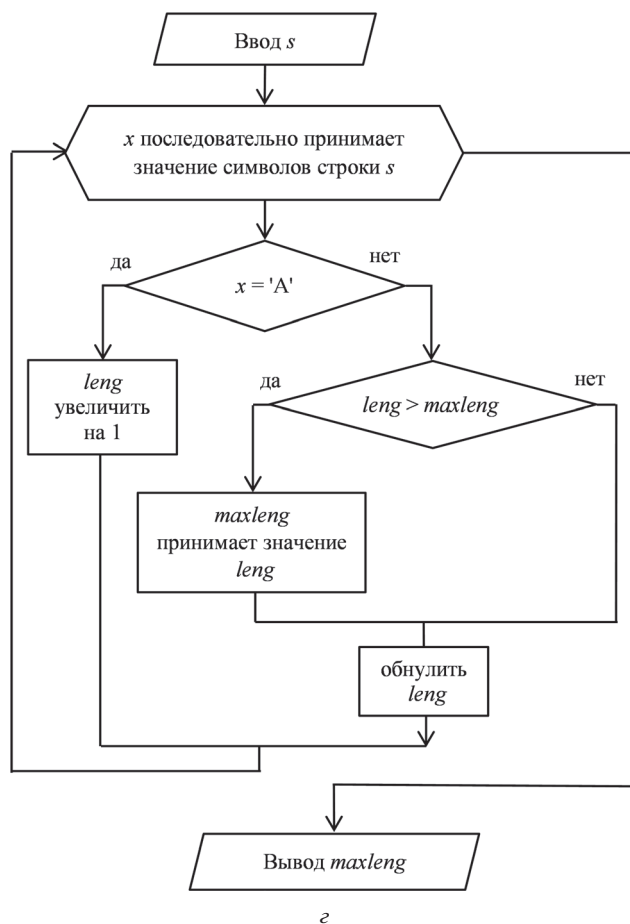
a



б



в



г

Рисунок к заданию 4

5. Уровни сформированности знаково-символических действий

Уровни сформированности знаково-символических действий можно оценить по следующим параметрам [9]:

- рефлексивность;
- обратимость;
- инвариантность;
- интенция.

Рефлексивность предполагает осознание двух планов: обозначаемого и обозначающего, знание алфавита и синтаксиса используемого языка. *Обратимость* — возможность перехода от одного плана к другому и обратно, от использования одного языка к использованию другого. *Инвариантность* — сохранение содержания при изменении его формы. *Интенция* — сознательное и произвольное построение и применение знаково-символических средств [9, с. 102].

Задания 9 и 10 можно использовать для формирования и контроля сформированности обратимости знаково-символической операции при написании программы для решения приведенной выше задачи.

Задание 9.

Заполните пропуски в трассировочной таблице (табл. 4).

Задание 10.

Что будет выведено на экран при выполнении приведенной ниже программы?

```
s='QQUQQQUUQ'
leng=0
maxleng=0
for x in s:
    if x=='Q':
        leng+=1
    else:
        leng=0
    maxleng=max(maxleng,leng)
    print(x,leng,maxleng)
```

В задании 11 в явном виде выделен инвариант схемы действия, изменение условия задачи способствует переносу инварианта в новые условия.

Задание 11.

Заполните пропуски в программе для решения задачи: «Найти максимальную длину подстроки, состоящей из одинаковых символов».

```
leng=...
maxleng=...
for i in ...:
    if s[i]==s[i+1]:
        leng=...
    else:
        leng=...
        maxleng=...
print(...)
```

Задание 12 проверяет сохранение инварианта в измененных условиях.

Задание 12.

Расположите блоки кода для решения задачи: «Найти максимальную длину подстроки, состоящей из символов А, В или С» в правильном порядке и с правильными отступами:

```
s=input()
if x in 'ABC':
    k+=1
print(m)
m=0
for x in s:
    k=0
    m=max(m,k)
else:
    k=0
```

Уровень инвариантности знаково-символических действий можно оценить, изменяя либо условие задачи, либо инструмент ее решения.

При освоении алгоритма решения приведенной выше задачи можно *изменить условие* — предложить учащимся написать программы, в которых требуется найти максимальную длину подстроки, состоящей только из символов А и В; из одинаковых символов; из попарно различных символов; и т. п. Подобные задачи можно найти в [6].

Изменение инструмента можно осуществить, предложив школьникам решить с помощью средств табличного процессора задачу, аналогичную приведенной выше,

Таблица 4

Трассировочная таблица к заданию 9

Код программы	Значения переменных и условий							
s='DCAADAAAC' leng=0 maxleng=0 for x in s:	Итерация цикла							
if x=='A':	нет	нет	да					
leng+=1			1					
else:								
leng=0	0	0						
maxleng=max(maxleng,leng)	0	0	1					

а именно задачу: «Дана последовательность символов латинского алфавита, расположенная в первом столбце электронной таблицы. Найдите максимальную длину подпоследовательности, состоящей из символов А».

6. Заключение

Л. Л. Босова отмечает важность использования в учебном процессе не разрозненных задач, а специальным образом сформированных последовательностей задач [2]. Последовательное решение учащимися представленных выше заданий позволит соединить содержание и форму действия, предупредить формализм в усвоении знаний. Реализация описанных упражнений в цифровом формате, с автоматической проверкой позволит сократить временные затраты учителя на проверку работ учащихся, а школьникам предоставить информацию о правильности выполнения отдельных заданий непосредственно в процессе их решения [3, 7]. Для создания подобного контента можно использовать возможности различных образовательных платформ (см., например, [10, 11]).

Семиотический аспект является лишь одной из сторон обучения программированию. Когда школьник только начинает изучать программирование, соединение содержания и формы действия вызывает затруднения. Использование заданий, в которых знаковая ситуация и различные ее аспекты выделены в явном виде, существенно облегчает переход на следующий этап обучения программированию.

Список источников

1. Авербух А. В., Гусин В. Б., Зайдельман Я. Н., Лебедев Г. В. Изучение основ информатики и вычислительной техники: пособие для учителя. М.: Просвещение, 1992. 302 с.

2. Босова Л. Л. Программирование как инструмент формирования вычислительного мышления обучающихся // Информатика в школе. 2020. № 10. С. 4–10. EDN: GURIPH. DOI: 10.32517/2221-1993-2020-19-10-4-10

3. Быкова И. А., Заводчикова Н. И. Укрупненные упражнения как средство подготовки школьников к основному государственному экзамену по информатике // Информатика в школе. 2021. № 8. С. 12–17. EDN: WJSYCN. DOI: 10.32517/2221-1993-2021-20-8-12-17

4. Гребнева Д. М. Обзор методических подходов к обучению программированию в школе // Научное обозрение. Педагогические науки. 2016. № 3. С. 13–27. EDN: WLYGER.

5. Гребнева Д. М., Егорова Л. Е. Обучение школьников программированию на основе семиотического подхода. Нижний Тагил, 2018. 112 с. EDN: YOCWNN.

6. ЕГЭ по информатике (2024) // Преподавание, наука и жизнь: сайт Константина Полякова. <https://kpolyakov.spb.ru/school/ege.htm>

7. Заводчикова Н. И., Быкова И. А. Управление познавательной деятельностью учащихся при изучении программирования на уроках информатики // Математика и проблемы образования. Материалы 41-го Международного научного семинара преподавателей математики и информатики университетов и педагогических вузов. Киров, 2022. С. 213–215. EDN: OQIDQI.

8. Курганова Н. А. Развитие знаково-символической деятельности учащихся в процессе обучения информатике на основе семиотического подхода: автореф. дис... канд. пед. наук. Омск, 2006. 22 с. EDN: NKAICB.

9. Салмина Н. Г. Знак и символ в обучении. М.: Изд-во Московского университета, 1988. 288 с.

10. LearningApps.org — создание мультимедийных интерактивных упражнений. <https://learningapps.org/>

11. Stepik — образовательная платформа и конструктор онлайн-курсов. <https://stepik.org/>

12. Waite J., Sentance S. Teaching programming in schools: A review of approaches and strategies // Teach, learn, and make with the Raspberry Pi Foundation. <https://www.raspberrypi.org>