УДК 002:372.8

О. В. Якименко

ПРИМЕНЕНИЕ WEB-ВИЗУАЛИЗАТОРОВ В ШКОЛЬНОМ КУРСЕ ПРОГРАММИРОВАНИЯ

В статье рассматривается метод повышения качества усвоения материала, который основан на использовании при обучении программированию обучающих программ-тренажеров, а именно web-визуализаторов (программы-тренажеры, являющиеся web-приложениями и демонстрирующие процесс работы алгоритмов). Предлагается обучение основам программирования, в частности циклических конструкций, используя методику проблемного обучения.

Ключевые слова: *программирование, тренажер, web-визуализатор, проблемное обучение, методика обучения.*

Важной особенностью обучения программированию в школе является то, что обладание соответствующей компетенций не сводится к способности простого воспроизведения имеющихся знаний и не ограничивается умением применять шаблонные решения. Фактически любая реальная задача, решаемая программистом, требует нестандартного мышления и нестандартных действий.

Очевидно, что обучение программированию в школе начинается с рассмотрения элементарных конструкций. После изучения основ программирования на любом алгоритмическом языке следует перейти к решению типовых задач, которые являются основой алгоритмической культуры и служат опорным пунктом при дальнейшем изучении программирования. Подбор задач осуществляется на основе стандартной программы школьного курса [1]. Учащиеся должны овладеть первоначальными навыками программирования на языке высокого уровня, что включает в себя способность разрабатывать алгоритмы линейной структуры, использовать операторы ветвления, выбора, циклов, организации подпрограмм, в том числе рекурсивных. Ученик должен уметь использовать простые и составные типы данных: целочисленные, вещественные, символьные, массивы, записи. Наиболее важными в обучении программированию являются следующие алгоритмы: исполнение линейного алгоритма; условный оператор; циклы с предусловием; цикл с постусловием; цикл с параметром; суммирование элементов массива; поиск минимального и максимального элементов в массиве; простые сортировки массива; поиск подстроки; представление множества в ЭВМ; организация рекурсии; ввод, вывод в файл.

Для профильного курса информатики, ориентированного на углубленное изучение программирования, можно расширить предметное наполнение следующими темами [2]: деревья, обход дерева, переборные алгоритмы; матрицы, работа с числами и матрицами, строками, списками; инварианты, индуктивные доказательства; генерация псевдослучайных последовательностей.

В этом случае список рассматриваемых алгоритмов нужно расширить следующими алгоритмами: умножение матриц; улучшенные сортировки массива; внешние сортировки, сортировки файла; работа стека и очереди.

Данный список не является исчерпывающим.

Выбор данных задач обоснован целями и задачами обучения программированию в школе [1]. Однако следует помнить, что недопустимо связывать ученика какими-то определенными шаблонами, важно развивать способность действовать творчески.

При обучении программированию востребованными являются не только специальные знания и алгоритмическое мышление, развивающееся в процессе непосредственной алгоритмизации, но и сформированное логическое мышление.

Формирование логического мышления у учащихся, как правило, начинается в начальной школе при изучении математики и основ логики. Логические способности учеников могут быть развиты только в том случае, когда они активно участвуют в процессе усвоения новых знаний. Одним из наиболее эффективных методов развития самостоятельного логического мышления является проблемное обучение, так как именно оно наиболее близко к творческой деятельности ученого, которая характеризуется применением гипотезы, доказательства, эксперимента.

Приемы создания проблемных ситуаций выбираются в зависимости от конкретного содержания учебного материала. В одних случаях проблемная ситуация создается с явной опорой на имеющиеся знания учащихся. Опираясь на них, учащиеся делают вывод, который оказывается в противоречии с фактами. Это означает, что знания недостаточны, и нужна дополнительная информация для разрешения возникшего противоречия. Такой вариант проблемной ситуации всегда вызывает острый интерес у учащихся, отсюда и познавательная эффективность бывает высокой [3].

Ряд педагогов и психологов (В. Оконь В., И. Я. Лернер, М. И. Махмутов, Т. В. Кудрявцев [3]) предлагают

методику проблемного обучения для развития логического мышления. Проблемные методы основаны на создании проблемных ситуаций, активной познавательной деятельности учащихся, состоящих в поиске и решении сложных вопросов, требующих актуализации знаний, анализа, умений видеть за отдельными фактами явления и законы.

Учитель создает проблемную ситуацию, направляет учащихся на ее решение, организует поиск решения. Таким образом, ребенок в процессе поиска решения получает новые знания, он овладевает новыми способами действия. Проблемная ситуация специально создается учителем путем применения ряда особых методических приемов [3].

В зависимости от характера взаимодействия учителя и учащихся выделяют четыре уровня проблемного обучения [3]:

1. Уровень несамостоятельной активности – восприятие учениками объяснения учителя, усвоение образца умственного действия в условиях проблемной ситуации, выполнение учеником самостоятельных работ, упражнений воспроизводящего характера, устное воспроизведение.

При изучении программирования несамостоятельная активность проявляется при изучении готовых текстов программ. Так, например, изучение циклических конструкций можно начать с рассмотрения примера готового алгоритма решения задачи вычисления суммы натуральных чисел от 1 до N с помощью цикла while. Для этого можно подготовить раздаточный материал, содержащий программу, реализующую данный алгоритм, сопровождаемый комментариями:

```
program sum1;
var
        i, s, n: integer;
begin
        writeln(‘введите числа от 1 до n’);
        readln(n);
        i:=1; s:=0;
        while i<=n do begin
                s:=s+I;
                inc(i);
                end;
        writeln(‘s=’,s);
end.
```

2. Уровень полусамостоятельной активности характеризуется применением прежних знаний в новой ситуации и участие школьников в поиске способа решения поставленной учителем проблемы.

Данному уровню характерно изучение нового материала на основе подсказок учителя. В качестве таких подсказок можно использовать шаблоны программ. Так, например, при изучении цикла repeat учащимся можно дать шаблон программы вычисления суммы натуральных чисел от 1 до N с помощью цикла repeat:

```
program sum2;
var
        i, s, n: integer;
begin
        writeln(‘введите числа от 1 до n’);
        readln(n);
        i:=1; s:=0;
        repeat
                ……
        until
                …….
        writeln(‘s=’,s);
end.
```

У них должно получиться:

```
repeat
        s:=s+i;
        inc(i)
until i<=n;
```

3. Уровень самостоятельной активности – выполнение работ репродуктивно-поискового типа, когда ученик сам решает по тексту учебника, применяет прежние знания в новой ситуации, конструирует, решает задачи среднего уровня сложности, доказывает гипотезы с незначительной помощью учителя и т. д.

Для этого уровня необходимо поставить перед учеником задачу, которую должен решить самостоятельно, без помощи учителя. Например, решение задачи вычисления суммы натуральных чисел от 1 до N с помощью цикла for:

```
program sum3;
var
        i, s, n: integer;
begin
        writeln(‘введите числа от 1 до n’);
        readln(n);
        for i:=1 to n do
        s:=s+i;
        writeln(‘s=’,s);
end.
```

В качестве самостоятельного задания – задача на вычисление факториала (произведения натуральных чисел от 1 до N).

4. Уровень творческой активности – выполнение самостоятельных работ, требующих творческого воображения, логического анализа и догад-

ки, открытия нового способа решения учебной проблемы, самостоятельного доказательства; самостоятельные выводы и обобщения, изобретения, написание художественных сочинений.

На этом уровне нужно поставить задачу, при решении которой ученик будет не просто самостоятельно работать, но и подходить к этому творчески. Стимулировать творческую активность можно с использованием элемента метода проектов. Ученику следует предложить самостоятельно формализовать и решить практико-ориентированную задачу с использованием циклических конструкций. Например, можно поручить ученику написать программу, которая бы позволила подсчитать среднее количество тетрадей в портфелях учеников:

```
program notebook;
var i, k, s:integer;
        p:real;
begin
        writeln(‘введите количество учеников’);
        readln(n);
        s:=0;
for i:=1 to n do begin
        writeln(‘введите количество тетрадей у , i,
‘-го ученика ’);
        read(k);
        s:=s+k;
end;
p:=s/k;
writeln(‘среднее количество тетрадей’, p);
end.
```

Ученик, решая задачу, самостоятельно формирует математическую модель. Первым шагом будет произведен подсчет общего количество тетрадей за счет суммирования количества тетрадей, сообщаемого каждым учеником. По окончании суммирования подсчитанное значение необходимо разделить на количество учеников.

Аналогичные приемы можно использовать на протяжении обучения всему курсу.

Существенной трудностью реализации данного подхода является необходимость индивидуальной работы с учениками. Действительно, эффективность методики для каждого ученика будет достигнута, если он сам будет заполнять пропуски и сам выполнять задания. К сожалению, учителю трудно уделить необходимое внимание каждому ученику. Одним из решений данной проблемы может стать использование визуализаторов (программ-тренажеров), демонстрирующих внутреннюю логику алгоритма и управляющих деятельностью учеников. Важным моментом является возможность размещения тренажеров в сети Интернет. Данная технология позволяет обеспечить общедоступность создаваемых средств (технологически это приводит к необходимости реализации тренажеров в виде web-приложений). Программы-тренажеры, являющиеся web-приложениями и демонстрирующие процесс работы алгоритмов, можно назвать web-визуализаторами [4]. Данные продукты могут широко применяться как в традиционной классно-урочной системе, так и в самостоятельной работе.

Список литературы

1. Информационная система «Единое окно доступа к образовательным ресурсам»: база данных содержит примерные программы основного общего образования по информатике и ИТ. URL: http://window.edu.ru/window_catalog/pdf2txt?p_id=14196
2. Информационная система «Единое окно доступа к образовательным ресурсам»: база данных содержит примерную программу профильного уровня. URL: http://window.edu.ru/window_catalog/pdf2txt?p_id=14196
3. Вербицкий А. Психолого-педагогические основы образования взрослых: контекстный подход. URL: http://www.znanie.org/jornal/n2_01/psih_podhod.html
4. Якименко О. В., Стась А. Н. Применение обучающих программ-тренажеров в обучении программированию // Вестник ТГПУ. 2009. Выпуск 1 (79). Сер. «Педагогика: теория и практика». С. 54–56.

Якименко О. В., аспирант.
Томский государственный педагогический университет.
Ул. Киевская, 60, г. Томск, Томская область, Россия, 634041.
E-mail: yakimenkoOV@tspu.edu.ru

Материал поступил в редакцию 26.04.2010.

O. V. Yakimenko

USE OF WEB-VISUALIZERS IN COURSE OF STUDIES PROGRAMMING

The article concerns the method based on the use of computer tutors in classroom instruction, namely web-visualizers (which are web applications demonstrating the way algorithms work). We suggest programming principles, specifically cyclic design using the method of problematical searching.

Key words: *programming, web-visualizers, computer tutors, methods of teaching, problematical searching.*

Tomsk State Pedagogical University.
Kiyevskaya Street, 60, Tomsk, Tomsk region, Russia, 634041.
E-mail: yakimenkoOV@tspu.edu.ru