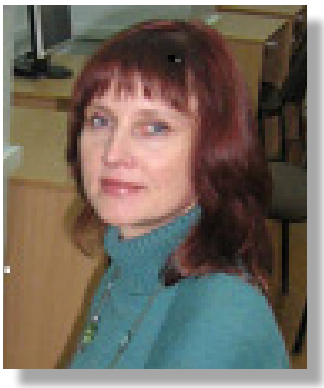
ФОРМИРОВАНИЕ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ УЧАЩИХСЯ ПРИ ОБУЧЕНИИ ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ



Фомичева Елена Иосифовна,
магистрант ВГУ имени П.М. Машерова

АЛГОРИТМИЧЕСКОЕ МЫШЛЕНИЕ – СОСТАВЛЯЮЩАЯ ИНТЕЛЛЕКТУАЛЬНОГО РАЗВИТИЯ

В статье раскрываются особенности формирования алгоритмического мышления учащихся. Рассмотрены основные принципы построения обучения, направленного на развитие алгоритмических способностей. Приведены примеры решения задач на объектном языке.

Введение. Важной задачей учреждения образования является формирование способностей учащегося, развитие его интеллекта и мышления. Существенной составляющей интеллектуального развития человека выступает алгоритмическое мышление. Большим потенциалом для формирования алгоритмических способностей учащихся среди различных дисциплин обладает программирование. «Решение задачи на компьютере невозможно без создания алгоритма. Умения решать задачи, разрабатывать стратегию ее решения, выдвигать и доказывать гипотезы опытным путем, прогнозировать результаты своей деятельности, анализировать и находить рациональные способы решения задачи путем оптимизации, детализации созданного алгоритма, представлять алгоритм в формализованном виде на языке исполнителя позволяют судить об уровне развития алгоритмических способностей школьников» [1].

Современное образование в области программирования сложно представить без объектно-ориентированного подхода. Так как в настоящее время языки программирования развиваются очень быстро, то эффективное обучение принципам объектно-ориентированного программирования требует исследования новых подходов к управлению процессами формирования алгоритмического мышления учащихся.

Приоритетной задачей обучения объектно-ориентированному программированию в средних специальных учебных заведениях является подготовка учащихся к продолжению образования в высшем учебном заведении и профессиональной деятельности, что потребует дальнейшего изучения программирования.

Основная часть. В процессе изучения различных тем дисциплин, связанных с программированием, «учащиеся должны уметь разрабатывать план решения задачи, выдвигать и доказывать гипотезы, прогнозировать результаты решения, анализировать и находить рациональные способы и т.д. Эти мыслительные умения характеризуют уровень развития алгоритмического мышления. Алгоритмическое мышление – познавательный процесс, характеризующийся наличием четкой, целесообразной последовательности совершаемых мыслительных процессов с присущей детализацией и оптимизацией укрупненных блоков, осознанным закреплением процесса получения конечного результата, представленного в формализованном виде на языке исполнителя с принятыми семантическими и синтаксическими правилами» [2].

В работах И.Н. Слинкиной [3] и Л.Г. Лучко [4] были определены три основных уровня развития алгоритмического мышления.

1. Операционный – учащийся понимает отдельные алгоритмические структуры и команды, но не может правильно использовать их, не владеет схемой их вложенности.

Репозиторий ВГУ

2. Системный – учащийся хорошо знает базовые алгоритмические конструкции и способы создания их сочетаний, умеет составлять программы для решения стандартных задач.

3. Методологический – учащийся грамотно использует уже имеющиеся мыслительные схемы для решения различных алгоритмических задач, может изменять их в зависимости от ситуации.

Необходимо при этом учитывать принципы построения обучения, которое направлено на развитие алгоритмического мышления:

– последовательность;

– системность;

– анализ;

– вариативность;

– полнота и всесторонность рассмотрения отдельных действий, входящих в систему.

Многие знают цитату Стива Джобса, разработчика первого персонального компьютера: «Каждый человек на планете должен учиться программированию на компьютере, потому что оно учит думать». Важно понять, что значит «думать, как программист», и как этому научиться.

Очень часто многие считают, что программирование – это только составление программы и получение результата, однако это и 1) анализ задачи; 2) выбор средств и методов решения; 3) формирование модели решения; 4) составление алгоритма в виде блок-схемы; 5) составление программы; 6) отладка программы; 7) анализ логических и синтаксических ошибок; 8) тестирование программы; 9) анализ полученных результатов.

Каждый указанный этап может вызвать трудности и создать проблемную ситуацию. Преодоление этих трудностей, грамотное написание кода, нахождение ошибок и их исправление, рациональное использование ресурсов развивают алгоритмическое мышление учащихся.

Следования, ветвления, циклы и функции – это элементы структурного программирования. Его возможностей хватает для написания небольших, простых программ. Более серьезные проекты часто реализуются с помощью средств объектно-ориентированного программирования.

Важно понимать суть объектно-ориентированного подхода в программировании. По названию языка понятно, что весь процесс организации программирования связан с объектами. Мир, в котором мы живем, представляет собой множество объектов, обладающих определенными свойствами, взаимодействующих между собой, изменяющихся и трансформирующихся. Эта привычная для взгляда человека картина мира была перенесена в программирование. Объектно-ориентированный подход заключается в моделировании объектов реального мира, интерпретируемых как совокупность свойств и поведения. Примерами свойств для людей могут быть цвет волос, рост или место работы; для автомобилей – мощность двигателя, количество посадочных мест. Таким образом, свойства объектов равносильны данным в программах. Поведение – это некоторое действие объекта в ответ на внешнее воздействие. Поведение сходно с функциями, которые вызываются, чтобы совершить какое-либо действие. Основная идея объектно-ориентированного подхода к программированию – это объединение данных и функций, оперирующих этими данными, в одно целое, которое и называется объектом.

Такой подход к программированию потребовал более высокого уровня абстракции при обработке и сохранении данных, конструировании программ и распределении программистами между собой задач. Данный подход дает возможность более легкой и продуктивной разработки серьезных проектов. Если команда программистов занимается разработкой игры, то эту программу можно представить как систему, состоящую из различных героев и среды их обитания, включающей множество задач и предметов. Каждый участник, оружие, растение, строение – это цифровой объект, в котором содержатся его свойства и методы, с помощью которых он может влиять на свои свойства и свойства других объектов. Программист из команды разрабатывает определенную группу объектов. Разработчикам достаточно договориться между собой только о том, как объекты будут взаимодействовать, то есть об их интерфейсах.

Для управления процессом формирования алгоритмического мышления учащихся при обучении объектно-ориентированному программированию важно научить их рассматривать каждую задачу через призму следующих элементов:

1. Понимание. Необходимо четкое понимание сути задачи. Очень часто «сложные» задачи кажутся такими потому, что учащийся неправильно их понимает. Если задача понятная, то ее можно просто объяснить другому человеку обычным языком. Многие учащиеся не могут это сделать и сразу замечают пробелы в логическом и абстрактном мышлении, которых до этого не осознавали. «Если вы не можете объяснить что-либо простыми словами, вы это не понимаете» (Ричард Фейнма).

2. План. Важно научиться составлять план решения задачи и разбивать ее на несколько конкретных этапов.

3. Анализ. Сложную и объемную задачу лучше разбить на несколько более простых подзадач. Начинать необходимо с самой простой: с той, ответ на которую уже известен или очевиден, а также не зависит от решения остальных подзадач. После того как все подзадачи решены, ответы собираются вместе, и это решение исходной проблемы.

В случае возникновения проблемной ситуации необходимо проанализировать следующие этапы:

Репозиторий ВГУ

1. Отладка: нужно пошагово пройти по решению в поиске ошибок.

2. «Дебаггинг – это искусство отличать, что вы на самом деле сказали сделать программе, от того, что вы думали, что сказали ей сделать» (Эндрю Сингер).

3. Переоценка: необходимо вернуться к предыдущему этапу и посмотреть на проблему с другой стороны.

Например, требуется составить программу решения следующей задачи на объектном языке. Сформируйте класс «Круг»: координаты центра, радиус, конструктор. Сформируйте класс «Точка на плоскости»: координаты, конструктор. Проверьте, принадлежит ли точка кругу. Используйте дружественную функцию.

Прежде всего, необходимо понять задачу и составить план ее решения:

1. Программа будет содержать два класса и одну дружественную функцию, которая является глобальной и будет определена вне обоих классов.

2. Класс «Точка на плоскости» включает свойства: координаты (x1, y1) точки.

3. Класс «Круг» содержит свойства: координаты центра круга (x0, y0) и радиус r.

4. Дружественная функция определяет принадлежность заданной точки кругу. Используя уравнение окружности, можно составить условие определения принадлежности: ((x1 – x0)(x1 – x0) + + (y1 – y0)*(y1 – y0)< = r*r).*

5. Определенное выше условие в программе записывается с использованием объектов класса.

В указанной ниже программе установлены классы с общей дружественной функцией, в главной функции объявлены и инициализированы объекты этих классов и выполнен вызов дружественной функции для определения принадлежности заданной точки кругу.

Выполнена отладка программы и получен верный результат.

```
#include “stdafx.h”
#include <iostream>
#include <iomanip>
using namespace std;
class krug;
class point
{float x,y;
public:
point(float,float);
friend void prov(point,krug);};
point::point(float x1,float y1)
{x=x1; y=y1;}
class krug
{float x0,y0,r;
public:
krug(float,float,float);
friend void prov(point,krug);};
krug::krug(float x2,float y2,float r2)
{x0=x2;y0=y2;r=r2;}
void prov(point p,krug k)
{if ((p.x-k.x0)*(p.x-k.x0)+(p.y-k.y0)*(p.y-k.y0)<=k.r*k.r) cout<<“Точка принадлежит кругу yes”<<endl;else cout<<“Точка не принадлежит кругу not”<<endl;}
int _tmain(int argc, _TCHAR* argv[])
{ point P(4.0,5.5);
krug K(0.0,0.0,3.0); cout<<“Результат:”<<endl; prov(P,K);return 0;}
```

4. Практика.

Чтобы овладеть указанными выше навыками, нужно решить много задач. Только со временем и постоянной практикой придет понимание того, как легко находить подход к той или иной проблеме.

Заключение. Основные принципы построения обучения, направленного на развитие алгоритмического мышления, сводятся к следующим: систематичность работы, ориентированной на развитие алгоритмического мышления; системность, полнота и всесторонность рассмотрения отдельных действий, входящих в структуру алгоритмического мышления; возможность соотнесения полученных результатов с эталоном [3].

Благодаря такой работе у учащегося снимается психологический барьер перед поиском решения задачи. Понимая, что программа может быть составлена с использованием различных подходов, он смелее берется за разработку программ на объектно-ориентированном языке. Постепенно учащийся приобретает некоторый опыт, что позволяет ему развивать алгоритмическое мышление, систематизируются знания, используется накопившийся опыт для решения аналогичного класса задач, расширяется общеобразовательный кругозор.

Таким образом, развитие алгоритмического мышления представляет собой процесс, проходящий в несколько этапов – начиная с начальной школы и заканчивая процессом обучения в вузе.

ЛИТЕРАТУРА

1. Семакин, И.Г. Преподавание базового курса информатики в средней школе: метод. пособие [текст] / И.Г. Семакин. – 2-е изд., испр. и доп. – М.: БИНОМ. Лаборатория знаний, 2009. – 228 с.
2. Газейкина, А.И. Стили мышления и обучение программированию студентов педагогического вуза [Электронный ресурс] / А.И. Газейкина. – Режим доступа: http://ito.edu.ru/2013/Moscow/I/1/I-1-6371.html. – Дата доступа: 07.10.2019.
3. Слинкина, И.Н. Использование компьютерной техники в процессе развития алгоритмического мышления у младших школьников [текст] / И.Н. Слинкина. – Екатеринбург: УрГПУ, 2010. – 22 с.
4. Лучко, Л.Г. Решение задач школьного курса информатики [текст]: учеб.-метод. пособие / Л.Г. Лучко. – Омск: ОмГПУ, 2001. – 80 с.

Репозиторий ВГУ