В. В. Кашей,

Академия социального управления, г. Москва

ПРОБЛЕМЫ ИЗУЧЕНИЯ МЕТОДОВ СОВРЕМЕННОГО ПРОГРАММИРОВАНИЯ В КУРСЕ ИНФОРМАТИКИ В УСЛОВИЯХ РЕАЛИЗАЦИИ ФГОС ОБЩЕГО ОБРАЗОВАНИЯ

Аннотация

В статье рассматривается общий подход к изучению программирования с точки зрения школьного курса информатики и с позиций технологий современного программирования, его возможностей. На основании проведенного сравнения определяются достоинства и недостатки классического подхода к изучению программирования и подхода, определяющего основные понятия объектно-ориентированного, событийного, визуального программирования как базовые в начале изучения. Определяются принципы и предлагается методика обучения современному программированию.

Ключевые слова: информатика, межпредметные связи.

Введение в школе предмета «Информатика» первоначально подразумевалось как обучение во вводимом курсе искусству программирования. Популярным был лозунг «Программирование — вторая грамотность». Однако к моменту введения в школьный курс «науки о компьютерах» в качестве обязательного предмета в 1985 году стало ясно, что ограничиваться только обучением программированию нельзя. Тем не менее сегодня раздел «Алгоритмизация и программирование» занимает важнейшее место в курсе информатики.

Овладение умением программировать в рамках предмета «Информатика» в школе является базовой основой для профессиональной подготовки специалиста в самых различных сферах человеческой деятельности. В процессе овладения программированием у обучаемых должны вырабатываться навыки анализа ситуации, выделения существенных и несущественных факторов, логического и специфического алгоритмического мышления, взаимодействия с компьютером, аналитического и синтетического мышления, выявления причинно-следственных

связей для процессов, имеющих длинную цепочку связей и реализующих различные, в том числе случайные, звенья в этой цепочке, анализа полученных результатов и возможностей их изменения при изменении входных условий или передаточной функции объекта управления (программы).

Особо следует отметить выработку у обучающихся особого, алгоритмического типа мышления, которое помогает решать не только специфические задачи программирования, но и проблемы из других сфер человеческой деятельности.

Изучение программирования в школьном курсе информатики обычно проводится по классической схеме, при которой изучается выбранный учителем императивный язык программирования классического стиля (Basic, Pascal и т. п.). Предложенная Н. Виртом схема «структуры данных + алгоритмы = программы» вместе с изучением трех основных типов алгоритмов и вспомогательных алгоритмов успешно реализуется в школьном курсе информатики и позволяет в основном достигнуть заявленных целей [2]. Главной проблемой, о которой говорят практически

Контактная информация

Кашей Владимир Васильевич, канд. пед. наук, доцент 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:* ww54@mail.ru

V. V. Kaschei,

Academy of Public Administration, Moscow

PROBLEMS OF STUDYING THE METHODS OF MODERN SOFTWARE ENGINEERING IN THE INFORMATICS COURSE IN THE IMPLEMENTATION OF THE FSES OF GENERAL EDUCATION

Abstract

The article describes a general approach to learning programming languages from the point of view of a school course of informatics and from the standpoint of modern technologies of programming, its possibilities. On the basis of the comparison the strengths and weaknesses of the classical approach to the study of programming and approach that defines the basic concepts of object-oriented, event-driven, visual programming as a baseline at the beginning of the study are determined. The principles and the methodology of teaching modern programming are defined.

Keywords: modern programming, visual programming, object-oriented programming, environment of automated development.

все учителя, является недостаточное количество времени, отводимого на изучение программирования в базовом курсе информатики.

Но постепенно появляется и другая проблема, которая с каждым днем становится все более актуальной, — отставание программирования классического вида, изучаемого в школе, от современного [3].

Рассмотрим несколько наиболее важных отличий современного программирования от классического.

Современное программирование имеет *визуальный характер* — программное обеспечение может создаваться манипулированием графическими объектами вместо написания текста программы. Если в обычном языке программирования даже для «рисовки» обычного окна требовалось выполнить ряд вычислений для определения его координат и вызвать несколько процедур для начертания самого окна, то в среде визуального программирования достаточно выбрать соответствующий компонент с помощью мыши и «протянуть» его границы. Среда сама сгенерирует и запишет необходимый код.

Современное программирование является *объектно-ориентированным*, что позволяет относительно легко создавать сложные проекты, используя разработанные ранее элементы, их значения и свойства, не переписывая каждый раз заново фрагменты алгоритмов и описания структур данных за счет использования механизма наследования.

В отличие от классических императивных языков программирования, современные языки программирования четвертого поколения (4GL — 4th Generation Languages) используют *событийный подход*: программа, в отличие от императивных языков, выполняется не последовательно от первого оператора до последнего, а отдельными фрагментами программного кода (подпрограммами и функциями с унифицированным автоматическим «интеллектуальным» интерфейсом), начиная с некоторого инициирующего события (прерывания), и заканчивается либо новым внешним прерыванием, изменяющим ход выполнения программы, либо создавая свое событие — прерывание для другого фрагмента. Программа в 4GL обычно не является однозначной, заранее предопределенной цепью последовательных фрагментов, как в императивных языках. Она представляет собой набор программных звеньев, последовательность выполнения которых определяется событиями. Такой подход обеспечивает работу современного программного обеспечения, работающего в режиме взаимодействия с другими программами, устройствами и человеком.

Еще одной характерной чертой современного программирования является *мультизадачность*. Это свойство допускает возможность одновременного выполнения сразу нескольких программ, что требует реализации механизма синхронизации программ и использования данных.

Современные системы разработки программного обеспечения *широко интегрированы* и позволяют разрабатывать приложения для работы с различными средствами, такими как, например, СУБД, базы знаний, Интернет и др.

Среды разработки (IDE — Integrated Development Environment) императивных языков, таких, напри-

мер, как Borland Pascal, Pascal ABC, совмещали в себе редактор кода, средства выполнения и отладки программ. Среды современных средств разработки программного обеспечения имеют значительно более мощные и разнообразные возможности, высокую степень автоматизации, используют элементы искусственного интеллекта.

Чтобы **уменьшить разрыв между изучаемым в школе и современным программированием**, следует начинать изучение объектно-ориентированного программирования, методов визуального программирования, современных сред разработки программного обеспечения уже в школьном курсе информатики. Однако малое время, отведенное на изучение информатики, сложность изложения и усвоения данного материала, необходимость подготовки к ЕГЭ, задания которого ориентированы на классические императивные языки, препятствуют такому изучению [1].

Возникает **противоречие**, выражающееся в необходимости изучения методов и сред современного программирования при нехватке времени и ориентации выходных испытаний (в данном случае ЕГЭ) на классические императивные языки, а также при высокой стоимости профессиональных сред разработки программного обеспечения. Например, цена среды разработки Delphi может составлять от 250 (стартовый комплект с минимальными возможностями) до 35 000 (полный профессиональный комплект) долларов.

Вопрос цены решается достаточно просто: можно воспользоваться аналогичными продуктами свободно распространяемого программного обеспечения (например, аналог для Delphi — это Lazarus) или договориться с фирмой и получить ограниченную по возможностям версию для обучения. Первый вариант представляется более привлекательным.

Ограничения, обусловленные необходимостью подготовки к решению заданий ЕГЭ, более существенны. Единственный выход в настоящий момент — перенести изучение ООП на более ранние сроки, т. е. в курс информатики основной школы. Сразу же возникают два возражения. Во-первых, времени для изучения информатики там и так мало. Во-вторых, ООП требует введения довольно сложных понятий (объект, наследование, полиморфизм и т. п.), базирующихся на информации, получаемой при изучении классических языков программирования.

Время можно найти, немного перестроив программу за счет использования средств ООП при изучении определенных тем, например, объединив разработку алгоритмов и их реализацию средствами ООП. Можно использовать и время, отводимое для внеурочной деятельности. К примеру, во внеурочное время обучающимися разрабатываются относительно простые исполнители алгоритмов для использования их на уроке в соответствующих классах.

Введение основных понятий ООП можно сделать аксиоматическим способом. Если в классическом подходе мы изучаем записи, подпрограммы, модули и только после этого переходим к определению понятия «объект», то в случае раннего изучения ООП мы вводим понятие объекта как данности, базового

элемента, имеющего определенные свойства и методы работы с ним, не вникая в происхождение и реализацию понятия «объект». Сразу же можно просто и доходчиво объяснить понятие «инкапсуляция».

В результате такого подхода мы создаем новый базис знаний в сфере программирования у обучающихся, отвечающий современным требованиям.

При использовании профессиональной среды выявляется проблема сложности ее интерфейса для начинающих. Эта проблема решается на основе принципа минимизации получаемой обучающимися информации. Учащийся получает на данном этапе минимум информации, необходимый ему для выполнения задания или проекта. При первом создании нового проекта он узнает, как создать новый проект и сохранить его в отведенной папке, что такое объект, форма и компонент среды, как создавать форму и компонент на ней, как задавать свойства объекта, как запустить проект на выполнение и как остановить исполнение проекта.

С каждым новым проектом обучающийся получает относительно небольшую порцию новых знаний и навыков. Причем необходимость их получения строго замотивирована: для получения результата необходимо знать это и уметь выполнять такие действия. Постепенно усложняя задания на выполнение проектов, мы заставляем обучающегося получать новые знания и навыки. Важно, что их получение мотивируется необходимостью выполнить задание. Учащийся на практике убеждается в том, что эти знания и навыки ему необходимы.

Очень важен **набор проектов, предлагаемых в процессе обучения**. Переход от одного проекта к другому не должен быть резким, объем получаемых знаний и навыков в результате такого перехода должен быть доступным для усвоения и не слишком большим.

Возможен следующий набор проектов:

1. Вывести на форму строку символов с приветствием.
2. Запросить имя пользователя и вывести строку с приветствием пользователя по его имени.
3. Вычислить сумму двух целых чисел.
4. Вычислить разность и произведение целых чисел.
5. Вычислить частное двух чисел.
6. Калькулятор: для двух чисел организовать вычисление суммы, разности, произведения или частного.
7. По введенной строке с текстом прогноза определить, будет ли сегодня дождь, и выдать рекомендацию, брать с собой зонт или нет.

8. По введенному тексту со словами «мороз» и «солнце» или их отсутствию определить, является ли сегодняшний день чудесным согласно известному утверждению А. С. Пушкина «Мороз и солнце — день чудесный».

9. По введенным значениям длины, ширины и высоты кирпича проверить, пройдет ли он в прямоугольное отверстие заданного размера.

Мы не будем приводить здесь весь список проектов. Отметим только, что в списке проектов есть проекты, реализующие работу с графикой, мультимедиа, файловой системой Windows, временем и многие другие.

Выводы:

- Необходимо знакомить обучающихся с современными тенденциями в программировании, методами и средами разработки программного обеспечения.
- Вследствие необходимости подготовки к ЕГЭ в старших классах изучение современных методов и сред программирования, включая объектно-ориентированное программирование, целесообразно перенести на более ранний период, возможно начинать изучение программирования с ООП, используя современную среду разработки.
- Для такого изучения должна использоваться специальная методика, учитывающая возможности и возрастные особенности обучающихся и базирующаяся на своей аксиоматике, вводящей понятия ООП как базовые.
- Возможна коллизия при изучении обучающимися в X—XI классах классических императивных языков программирования для успешной сдачи ЕГЭ после того, как они прошли курс ООП.

Список использованных источников

1. *Кащей В. В.* Диалектика единого государственного экзамена. Проблемы подготовки // Итоги научных исследований. Сб. статей Международной научно-практической конференции (5 марта 2015 г., г. Москва). М.: Европейский фонд инновационного развития, 2015.

2. *Кащей В. В.* О методике использования программно-аппаратных средств при преподавании информатики // Практическая педагогика и психология: методы и технологии. Сб. статей Международной научно-практической конференции (10 июня 2016 г., г. Казань). Уфа: АТЕРНА, 2016.

3. *Кащей В. В.* О стратегии выбора платформы для обучения алгоритмике и программированию в общеобразовательной организации // Академический вестник Академии социального управления. 2016. № 3 (21).