М. С. Можаров, С. В. Журавлев

НЕКОТОРЫЕ ОСОБЕННОСТИ МЕТОДИКИ ПРЕПОДАВАНИЯ ПРОГРАММИРОВАНИЯ ЗА РУБЕЖОМ

Ключевые слова: методы обучения программированию, теория когнитивной нагрузки, визуальная имитация выполнения программы, веб-ориентированная среда, парное программирование.

Keywords: methods of learning programming, cognitive theory load, visual program simulation, web-based environment, pair-programming.

Настоящая статья – попытка обобщить современные западные подходы к проблеме преподавания программирования. Нами были проанализированы диссертационные исследования, проведённые в ведущих университетах и отобраны результаты, разделяемые различными учёными.

В мире существует множество подходов к обучению студентов программированию. Мы проанализировали работы Г. Л. Уайта и М. П. Сивитанидеса; К. М. Хэнкока; М. Е. Касперсена; П. Д. ДеПаскуале III; Д. Тиг; А. Эккердал; Ю. Сорвы; Х. Лоде, Д. Е. Франчи и Н. Г. Фредериксена; Л. Ма; Д. А. Миллера; Н. Труонг; Т. Дженкинса. Рассмотрим некоторые вопросы методик преподавания программирования.

Одна из методик, описываемая в работе М. Касперсена (Университет г. Аргус в Дании), основана на теории когнитивной нагрузки.

Когнитивная нагрузка – нагрузка на рабочую память в процессе размышлений, рассуждений и решения задач.

Фундаментальная аксиома теории когнитивной нагрузки говорит о том, что результат обучения оказывается наилучшим, когда когнитивная нагрузка полностью использует часть рабочей памяти, необходимую для эффективного приобретения знаний. Слишком маленькая или слишком большая нагрузка приводит к низкому результату обучения. Следовательно, для оптимизации обучения необходимо добиться балансировки когнитивной нагрузки, не сводя её ни к минимуму, ни к максимуму.

Когнитивная нагрузка (L) в настоящее время делится на три непересекающиеся категории:

1) посторонняя когнитивная нагрузка (E) – нагрузка, которая мешает обучению, так как часто превышает пределы рабочей памяти;

2) уместная когнитивная нагрузка (G) – нагрузка, которая скорее способствует обучению, чем мешает, помогая приобретать знания. Она образуется путём повышения уровня когнитивных процессов, способствующих эффективному усвоению знаний;

3) внутренняя когнитивная нагрузка (I) – нагрузка, уменьшение которой приводит к снижению понимания. Она зависит от относительной сложности изучаемого материала и степени усвоения студентом предшествующего материала.

Отношения между L, E, G и I можно представить следующим образом:

L=E+G+I.

В этих условиях задача балансировки когнитивной нагрузки для оптимального обучения сводится к попыткам минимизации E и максимизации G.

Если рассматривать теорию когнитивной нагрузки применительно к обучению начинающих программистов, то одним из наиболее эффективных является метод, при котором процесс обучения осуществляется посредством предоставления преподавателем рекомендаций и скаффолдинга (поддержки, предоставляемой специалистом ученикам в выполнении какой-либо поставленной задачи). При таком подходе можно гарантировать, что студенты осваивают важные аспекты программирования, сохраняя при этом когнитивную нагрузку в пределах, способствующих успешному обучению.

Обучение на основе данного метода проводилось с использованием комплекса учебных средств (учебника, упражнений и заданий и видеоматериалов).

Учебник, который использовался в ходе обучения, создан на основе проблемно-ориентированного подхода и в соответствии с педагогическими принципами обучения объектно-ориентированному программированию на основе метода ученичества. Этот метод предполагает, что студенты сначала наблюдают за тем, как преподаватель демонстрирует использование новых методов или конструкций для создания программы, затем применяют новый материал к созданию проекта под руководством преподавателя и на последнем этапе разрабатывают программы самостоятельно.

Упражнения и задания, используемые в данной методике, включают в себя указания, оказывающие помощь студентам на определённых этапах разработки программы. На начальных этапах обучения даются подробные указания к выполнению заданий, а на завершающих этапах предоставляются лишь краткие рекомендации.

Ещё одной особенностью метода скаффолдинга является использование видеоматериалов (видеозахват экрана), которые показывают процесс разработки программы. На видео специалист демонстрирует применение методов, принципов и приёмов программирования, изучаемых в данном курсе. Также видеоматериал может показывать мелкие детали, которые настолько просты, что о них обычно мало рассказывают студентам. Такой подход очень удобен, так как сильные студенты могут пропускать то, что им понятно, а слабоуспевающие могут останавливать и пересматривать видео несколько раз. [1]

Следующая методика, которую мы рассмотрим, была описана в работе Ю. Сорвы из Научной школы Университета Аалто (Финляндия). Эта методика – визуальная имитация выполнения программы (ВИВП) – основана на том, что студент берёт на себя роль исполнителя программы: читает код, выполняет команды в соответствующем порядке, отслеживает поток управления. Такая деятельность помогает ему понять принцип работы программы.

ВИВП может рассматриваться как упражнение, в ходе выполнения которого студенты практикуются в трассировке программы. Выполняя такое упражнение, обучающийся должен не просто понять, как работает программа, а ещё и конкретным образом продемонстрировать своё понимание преподавателю.

Обучать начинающих программистов выполнению визуальной имитации удобно с помощью систем визуализации программ – специальных программ, которые отображают ход выполнения программы в текстовом или графическом режиме.

Согласно проблемно-ориентированной классификации Малетика и др., системы визуализации программ, прежде всего, категорируются по следующим параметрам:

1) задачи – для чего необходима визуализация (например, обратная разработка, поиск места ошибки);

2) аудитория – кому необходима визуализация (например, опытный разработчик, руководитель группы);

3) цель – что визуализирует система (например, исходный код, результаты выполнения программы);

4) форма представления – как представлена визуализация (например, двумерная графика, трёхмерные объекты);

5) способ представления – каким образом представляется визуализация (например, отображение на экране, виртуальная реальность).

Если рассматривать данную классификацию применительно к обучению студентов, то для этой цели подходят системы визуализации программ со следующими параметрами:

1) задача – помощь в изучении вводного курса программирования;

2) аудитория – начинающие программисты и преподаватели вводного курса программирования;

3) цель – визуализация пошагового выполнения программ;

4) форма представления значения не имеет;

5) способ представления – визуализацию должно быть видно на экране.

В настоящее время системы визуализации программ существуют для различных языков программирования. Так, например, для изучения Pascal предназначены программы Amethyst, EROSI, PlanAni, для изучения Basic – Basic Programming, для изучения C – Bradman, CMeRun, VINCE, OGRE, PlanAni, VIP, ViLLE и др., для изучения Java – JAVAVIS, OOP-Anim, JavaMod, JIVE, Memview, JavaTool, PlanAni и др., для изучения Python – PlanAni, ViLLE, Jype, Online Python Tutor, UUhistle.

В конечном счёте, хотя и косвенно, визуальная имитация выполнения программы способствует совершенствованию навыков написания программ. [4]

Также заслуживает внимания работа Н. Труонг (Квинслендский технологический университет, Австралия). В ней описывается методика обучения студентов программированию на начальном этапе с использованием веб-ориентированной среды.

В ходе работы в стандартных средах программирования начинающие студенты сталкиваются с определёнными трудностями:

1) установка и настройка среды программирования;

2) использование редактора среды программирования;

3) понимание вопросов, связанных с программированием и применение знания синтаксиса языка программирования к написанию программного кода;

4) понимание ошибок компиляции;

5) отладка.

Эти трудности значительно возрастают, когда студенты изучают такие языки программирования, как C++, Java или C#, поскольку эти языки содержат абстракцию высокого уровня и разработаны в основном для опытных программистов.

С целью чтобы избавить студентов от необходимости установки среды программирования и обучения использованию её редактора, разработаны веб-ориентированные среды программирования.

В настоящее время существует много веб-ориентированных сред программирования, наиболее известными из которых являются:

1) CodeSaw – поддерживает языки C, C++, Java, Perl, Python, Ruby, XML, HTML, PHP, SQL, PL/SQL, Java-Script;

2) CourseMaster – поддерживает языки C++, Java;

3) WebToTeach – поддерживает языки Java, C, C++, FORTRAN, Ada, Pascal;

4) CodeLab – поддерживает языки Java, C, C++ и др.;

5) InSTEP – поддерживает языки Java, C, C++;

6) WWW for Learning C++;

7) W4AP – поддерживает языки Java, C и др.;

8) JERPA – поддерживает язык Java;

9) VECR – поддерживает языки Java, C;

10) Ludwig – поддерживает язык C++;

11) ALECS – поддерживает язык C++;

12) ELP – поддерживает языки Java, C, C#.

Анализ перечисленных веб-ориентированных сред показал, что наиболее эффективной является система ELP.

ELP является клиент-серверной средой. Клиент работает через веб-браузер, для его запуска необходим пакет Java Runtime Environment. В ELP предусмотрены режимы «Студент» и «Преподаватель».

В варианте для студентов представлена система упражнений типа «заполнить пропуски», которые хранятся в базе данных на сервере. Каждое упражнение представляет собой программу с недостающими строками кода, которые обучающиеся должны дописать. При этом остальная часть кода недоступна для редактирования. Таким образом, студентам не нужно тратить время на обучение использованию редактора среды программирования, и они могут непосредственно сосредоточиться на изучении программирования.

Упражнения расположены по мере увеличения сложности. Несколько первых упражнений вообще не содержат пропусков, студенты сразу компилируют и запускают программу. Остальные задания могут содержать как один, так и несколько пропусков разного размера.

Для студентов предусмотрена возможность работать в трёх режимах: hint (совет), solution (решение) и My Program. Основной режим, в котором они выполняют упражнение, My Program. Перейдя в режим hint, они получат советы по заполнению пропусков, а в режиме solution отображается правильный ответ.

Заполнив недостающие строки кода, обучающиеся должны сохранить программу, прежде чем компилировать её. Упражнение отправляется на сервер для компиляции, и результаты отображаются на консольной панели под текстом программы. Если в программе нет ошибок, студенты могут загрузить и запустить её на своих компьютерах в режиме «offline». В противном случае компилятор выдаёт сообщения об ошибках, которые дополнены дружественными сообщениями и ссылками, позволяющими обучающимся легко находить строки кода, содержащие ошибки, а также комментарии к качеству и правильности программы, сформированные на основе комбинации статического и динамического анализа.

Таким образом, среда ELP предоставляет студентам мгновенную обратную связь, на основании которой они получают подробную информацию об ошибках в программе и могут их исправить.

В режиме «Преподаватель» помимо режимов hint, solution и My Program существует также режим template (шаблон), в котором преподаватели могут создавать и удалять пропуски в упражнении. Также они могут изменять советы в режиме hint, вносить изменения в текст программы в режиме My Program. Кроме того, преподаватели имеют возможность получать подробную информацию о выполнении студентами упражнений: количество доработок программы, интервал времени между попытками компиляции, успешность каждой попытки. Проанализировав эту информацию, они могут сделать вывод об умении студентов программировать и определить, кому необходимо предоставить дополнительную помощь. [6]

В завершение рассмотрим метод обучения программированию, который был описан в работе Д. Тиг (Квинслендский технологический университет, Австралия). Это метод парного программирования.

Парное программирование является одной из практик экстремального программирования, заключающейся в том, что два программиста работают бок о бок за одним компьютером. Один из них берёт на себя роль «ведущего», т. е. печатает код и решает проблемы с тактической точки зрения, другой становится «штурманом» – он мыслит стратегически, задаёт вопросы и отслеживает ошибки в коде. Они часто меняются местами.

По результатам исследований, проведённых на производстве, можно выделить ряд преимуществ парного программирования:

1) меньшее количество ошибок;

2) лучшее понимание кода;

3) более качественный код;

4) возможность учиться у партнёра;

5) улучшение дизайна;

6) постоянный просмотр кода;

7) креативность и мозговой штурм;

8) улучшение качества тестирования и отладки;

9) быстрое решение возникающих проблем.

При обучении студентов на основе метода парного программирования возникают некоторые проблемы, связанные со следующими факторами:

1) различия в способностях между студентами в паре;

2) понимание студентами правил парного программирования и готовность соблюдать их;

3) посещаемость студентов;

4) контроль со стороны преподавателей за соблюдением студентами правил парного программирования.

Для того чтобы обучение было эффективным, необходимо руководствоваться следующими принципами:

1) желательно, чтобы в пару входили студенты, обладающие схожими способностями;

2) проводить лабораторные занятия, на которых студенты работают в парах;

3) студентам, образующим пару, должно быть удобно сидеть рядом друг с другом, они оба должны иметь лёгкий доступ к монитору, мыши и клавиатуре;

4) состав пар должен меняться в течение семестра;

5) обучающиеся должны понимать, что о проблемах с партнёром нужно немедленно сообщить преподавателю, чтобы дать ему шанс исправить ситуацию;

6) программисты в паре должны работать для достижения общей цели;

7) лучше предлагать задания, которые могут быть выполнены на еженедельном лабораторном занятии, чем ожидать, что обучающиеся будут встречаться во внеучебное время, возможно, испытывая при этом трудности с материально-техническим обеспечением;

8) установить стандарт кодирования, которого должен придерживаться каждый студент;

9) придерживаться принципа сотрудничества, взаимного уважения и общей ответственности в отношениях между обоими студентами в паре и преподавателями;

10) контролировать посещаемость и опоздания, чтобы обучающиеся не оставались без партнёра;

11) осуществлять контроль над балансом между индивидуальной и совместной работой студентов;

12) преподаватели должны побуждать пары самостоятельно искать ответы на возникающие вопросы, а не предоставлять им их.

Проведённые исследования подтвердили эффективность использования метода парного программирования в обучении студентов. [5]

Таким образом, в качестве основных методических подходов к преподаванию программирования можно выделить:

1) подход, основанный на теории когнитивной нагрузки;

2) подход, основанный на визуальной имитации выполнения программы;

3) обучение с использованием веб-ориентированной среды программирования;

4) обучение с использованием метода парного программирования.

Эффективность описанных подходов была доказана экспериментальным путём. Преподавателям российских ВУЗов будет полезно подробнее ознакомиться с этими и другими подходами и научиться использовать их при обучении студентов.

Литература

1. Можаров М. С. О развитии содержательной линии «Моделирование и формализация» в школьном курсе «Информатика и ИКТ» / М. С. Можаров, С. Д. Коткин // Информатика и образование. – 2010. – №4. – С. 95-99.

2. Можаров М. С. Практикум по решению задач в среде Lazarus с использованием модульно-рейтинговой системы оценивания: учебно-методическое пособие для студентов / М. С. Можаров, Ю. И. Валеева, Л. В. Попова. – Новокузнецк: изд-во КузГПА, 2013. – 150 с.

3. Можаров М. С. Формирование нового содержания образования как важнейшая составляющая профессиональной деятельности учителя информатики / М. С. Можаров // Педагогическое образование и наука. – 2009. – №9. – С. 84-87.

4. Caspersen M. E. Educating novices in the skills of programming: dissertation for the PhD degree / M. E. Caspersen. – Aarhus, 2007. – 311 p.

5. DePasquale III P. J. Implications on the learning of programming through the implementation of subsets in program development environments: dissertation for the degree of doctor of philosophy in computer science and applications / P. J. DePasquale III. – Blacksburg, 2003. – 575 p.

6. Eckerdal A. Novice programming students' learning of concepts and practice: digital comprehensive summaries of Uppsala dissertations from the faculty of science and technology / A. Eckerdal. – Uppsala, 2009. – 76 p.

7. Eckerdal A. Novice students' learning of object-oriented programming: dissertation for the degree of licentiate of phylosophy in computer science / A. Eckerdal. – Uppsala, 2006. – 114 p.

8. Hancock C. M. Real-time programming and the big ideas of compulational literacy: dissertation for the degree of doctor of philosophy in media arts and sciences / C. M. Hancock. – Cambridge, 2003. – 121 p.

9. Jenkins T. The motivation of students of programming: thesis for the degree of master of science / T. Jenkins. – Canterbury, 2001. – 211 p.

10. Lode H. Learning games for programming: thesis for the degree of master of science of media technology and games / H. Lode, G. E. Franchi, N. G. Frederiksen. – Copenhagen, 2012. – 111 p.

11. Ma L. Investigating and improving novice programmers' mental models of programming concepts: thesis for the degree of doctor of philosophy / L. Ma. – Glasgow, 2007. – 208 p.

12. Miller J. A. Promoting computer literacy through programming python: dissertation for the degree of doctor of philosophy (Education) / J. A. Miller. – Ann Arbor, 2004. – 288 p.

13. Sorva J. Visual program simulation in introductory programming education: doctoral dissertation for the degree of doctor of science in technology / J. Sorva. – Aalto, 2012. – 422 p.

14. Teague D. Pedagogy of introductory computer programming: a people-first approach: thesis for the degree of master of information technology (research) / D. Teague. – Queensland, 2011. – 129 p.

15. Truong N. A web-based programming environment for novice programmers: dissertation for the degree of doctor of philosophy / N. Truong. – Queensland, 2007. – 286 p.

16. White G. L. A theory of the relationships between cognitive requirements of computer programming languages and programmers' cognitive characteristics / G. L. White, M. P. Sivitanides // Journal of information systems education. – 2002. – Vol. 13, №1. – P. 59-66.