**МЕТОДИКА ПОЭТАПНОГО УСЛОЖНЕНИЯ ЗАДАЧ: ПУТЬ ОТ ИСПОЛЬЗОВАНИЯ ИНСТРУКЦИЙ К САМОСТОЯТЕЛЬНОМУ ПРОГРАММИРОВАНИЮ**

**Тематическое направление статьи «Методика обучения»**

***Сергиенко Дарья Михайловна****1\*****, Исмагулов Адильбек Нурланович****2\**

***1учитель-модератор NIS информатики***

***1Назарбаев Интеллектуальная школа химико-биологического направления г.Петропавловск, Республика Казахстан***

[***1\*sergiyenko\_d@ptr.nis.edu.kz***](mailto:1*sergiyenko_d@ptr.nis.edu.kz)

***2учитель-модератор NIS информатики***

***2Назарбаев Интеллектуальная школа химико-биологического направления г.Петропавловск, Республика Казахстан***

[***2\*ismagulov\_a@ptr.nis.edu.kz***](mailto:2*ismagulov_a@ptr.nis.edu.kz)

**Абстракт**

*В данной статье рассматриваются методы поэтапного усложнения задач при обучении программированию, направленные на переход от выполнения инструкций к самостоятельному написанию кода. Исследование проводилось на уроках информатики в 10 классе при изучении языка программирования C# и в 8 классе при изучении C++. Цель исследования — определить эффективность методики в зависимости от уровня академических способностей учащихся. Были применены четыре этапа: создание программы по готовым инструкциям (разбор готового кода), частичное программирование, кодирование по алгоритму и самостоятельное программирование. В работе использовались методы теоретического анализа, педагогического эксперимента и сравнительного анализа результатов учащихся с разным уровнем подготовки (A, B, C). Результаты показали, что постепенное снижение объема инструкций способствует развитию навыков программирования у учащихся с разным уровнем подготовки. Учащиеся с высоким уровнем подготовки (Ученик A) быстрее адаптировались к самостоятельному программированию, тогда как учащиеся со средним (Ученик B) и низким (Ученик C) уровнем подготовки показали значительный прогресс, но требовали дополнительной поддержки. Полученные данные могут быть использованы для разработки более эффективных учебных программ по программированию.*

***Ключевые слова:*** *программирование, методика обучения, C#, C++, поэтапное усложнение задач, самостоятельное написание программ, учебный процесс.*

**Введение**

Методика постепенного усложнения задач представляет собой подход к обучению программированию, который предполагает последовательный переход от работы с готовым кодом по инструкциям, представленным учителем на уроках информатики, к полному самостоятельному программированию. Эта методика основана на принципах конструктивистского подхода (Papert, 1980), согласно которому процесс обучения эффективнее, когда учащиеся активно конструируют знания, а не просто воспроизводят готовые решения по инструкциям. Конструктивистский подход предполагает, что учащиеся лучше усваивают материал, когда активно вовлечены в процесс создания знаний. Методика поэтапного усложнения задач, основанная на этом подходе, позволяет учащимся постепенно переходить от простых к сложным заданиям, что способствует развитию самостоятельного мышления.

Многие исследователи отмечают, что традиционные методы обучения программированию часто ориентированы на строгое выполнение инструкций, что не способствует развитию самостоятельного мышления и анализа (Bransford et al., 2000; Robins et al., 2003). Учащиеся, особенно с низким уровнем академических способностей, часто испытывают трудности при переходе от инструктивного обучения к самостоятельному программированию.

Актуальность данной методики обусловлена необходимостью развития навыков программирования у учащихся с разным уровнем академических способностей. В условиях цифровой трансформации общества навыки программирования становятся важным элементом образования. Однако традиционные методы обучения, основанные на строгом выполнении инструкций, не всегда эффективны для учащихся с разным уровнем подготовки. Это делает необходимым разработку новых подходов, таких как методика поэтапного усложнения задач, которая позволяет учитывать индивидуальные способности учащихся.  Согласно современным исследованиям (Guzdial & Soloway, 2002; Becker & Park, 2011), подобные подходы способствуют лучшему усвоению знаний и формированию устойчивых навыков программирования.

Целью исследования является определение эффективности последовательного сокращения инструкций и увеличения самостоятельности учащихся в процессе обучения программированию с учетом их индивидуальных академических особенностей.

Задачи исследования:

1. Проанализировать существующие подходы к обучению программированию.
2. Разработать методику, включающую четыре этапа обучения: создание программы по готовым инструкциям (анализ готового кода), частичное программирование, кодирование по алгоритму и самостоятельное программирование.
3. Применить методику на учащихся 10 класса (C#) и 8 класса (C++) с разными уровнями академических способностей (A, B, C).
4. Оценить эффективность методики, выявить сложные этапы и предложить адаптивные подходы.

Гипотеза исследования заключается в том, что методика постепенного усложнения задач способствует более эффективному освоению программирования по сравнению с традиционными методами, так как позволяет учащимся постепенно развивать навыки самостоятельного мышления и анализа.

Ожидаемые результаты: улучшение навыков программирования у учащихся с различными академическими способностями, снижение зависимости от инструкций при решении задач, повышение уровня самостоятельности учащихся при создании программ. Также предполагается, что учащиеся с низким уровнем подготовки покажут значительный прогресс при использовании адаптивных подходов.

**Методология**

Методология исследования направлена на проверку эффективности методики постепенного усложнения задач при обучении программированию. Исследование проводилось на основе теоретического анализа, педагогического эксперимента и сравнительного анализа.

Теоретический анализ включал изучение конструктивистских подходов к обучению программированию (Papert, 1980), и концепций активного обучения (Shneiderman, 1997). Эти подходы подчеркивают необходимость постепенного перехода от готовых решений к самостоятельному программированию через серию адаптированных заданий.

В педагогическом эксперименте приняли участие учащиеся 10 класса, изучающие объектно-ориентированное программирование на языке C#, и учащиеся 8 класса, изучающие процедурный язык программирования на языке C++. Каждая группа включала трёх учащихся с разным уровнем академических способностей:

Ученик A (высокий уровень) обладает хорошими аналитическими и логическими способностями, проявляет высокий интерес к программированию.

Ученик B (средний уровень) успешно справляется с типовыми задачами, но испытывает затруднения при решении более сложных задач без поддержки.

Ученик C (низкий уровень) испытывает сложности в понимании задач и требует дополнительной помощи на каждом этапе обучения.

Методика обучения состояла из четырех этапов, которые обеспечивали постепенное снижение уровня внешней поддержки и увеличение самостоятельности учащихся.

1. Создание программы по готовым инструкциям (анализ готового кода).
2. Частичное программирование (работа с фрагментами кода).
3. Кодирование по алгоритму (минимальные подсказки).
4. Самостоятельное программирование (без поддержки).

Использование четырех этапов оправдано необходимостью обеспечить плавный переход от инструкций к самостоятельному программированию, что особенно важно для учащихся со средним и низким уровнем подготовки.

Для оценки эффективности методики проводился сравнительный анализ результатов выполнения заданий на каждом этапе. Уровень усвоения проверялся с использованием формативного оценивания, включающего короткие проверочные задания в начале уроков, а также практические задания, выполняемые на уроках информатики.

**Результаты**

Результаты исследования подробно показывают, как учащиеся справлялись с заданиями на каждом этапе методики.

1. Создание программы по готовым инструкциям. Учащимся предоставлялись подробные инструкции по созданию интерфейса и написанию программного кода. Они анализировали готовый код и воспроизводили его в ходе практических работ, что позволяло им освоить базовые элементы синтаксиса и структуры программ.

Цель: научить создавать пользовательский графический интерфейс, понимать код и воспроизводить его.

Применение: в 10 классе — создание приложений в C#; в 8 классе — использование циклических алгоритмов и работа с одномерными массивами на C++.

Все учащиеся успешно выполнили задания, так как инструкции были четкими и последовательными. Этап создания программы по готовым инструкциям способствует эффективному развитию навыков программирования у всех учащихся. Однако наблюдались различия в подходах к выполнению заданий среди учеников с разным уровнем подготовки. Ученик A, обладающий высоким уровнем способностей, после нескольких примеров начал пропускать этот этап, предпочитая самостоятельно решать задачи. Такой подход указывает на стремление ученика A к более глубокому освоению материала через самостоятельную практику. Однако это также создаёт риск пропуска важных элементов базовой подготовки, что может привести к пробелам в понимании сложных концепций. Для повышения мотивации ученика A использовались дифференцированные задания, требующие применения новых алгоритмов. Это позволило ему глубже освоить материал.

2. Частичное программирование. Учащиеся использовали готовые фрагменты кода, которые требовалось адаптировать для решения поставленной задачи, внося изменения в код, добавляя новые функции.

Цель: развитие способности учащихся модифицировать код и использовать готовые элементы.

Анализ результатов показал, что на данном этапе учащиеся B и C испытывали затруднения при адаптации готового кода, но использование инструкций способствовало снижению уровня стресса и улучшению понимания структуры программного кода. Это позволило учащимся B и C постепенно переходить к более сложным заданиям.

3. Кодирование по алгоритму. Учащиеся выполняли задания с минимальными подсказками, основываясь на представленных алгоритмах и логике решения задач.

Цель: формирование навыков самостоятельного перевода алгоритмов в код.

Этап оказался наиболее привлекательным для учащегося уровня A, поскольку предоставлял возможность самостоятельно интерпретировать алгоритмы и переводить их в программный код, что стимулировало его аналитическое мышление и творческую активность. В то же время учащиеся уровней B и C проявляли значительно меньший интерес к данному этапу, что выражалось в их неохотном подходе к выполнению подобных задач. Переход к минимальным подсказкам затруднял процесс программирования в основном для учащихся C.

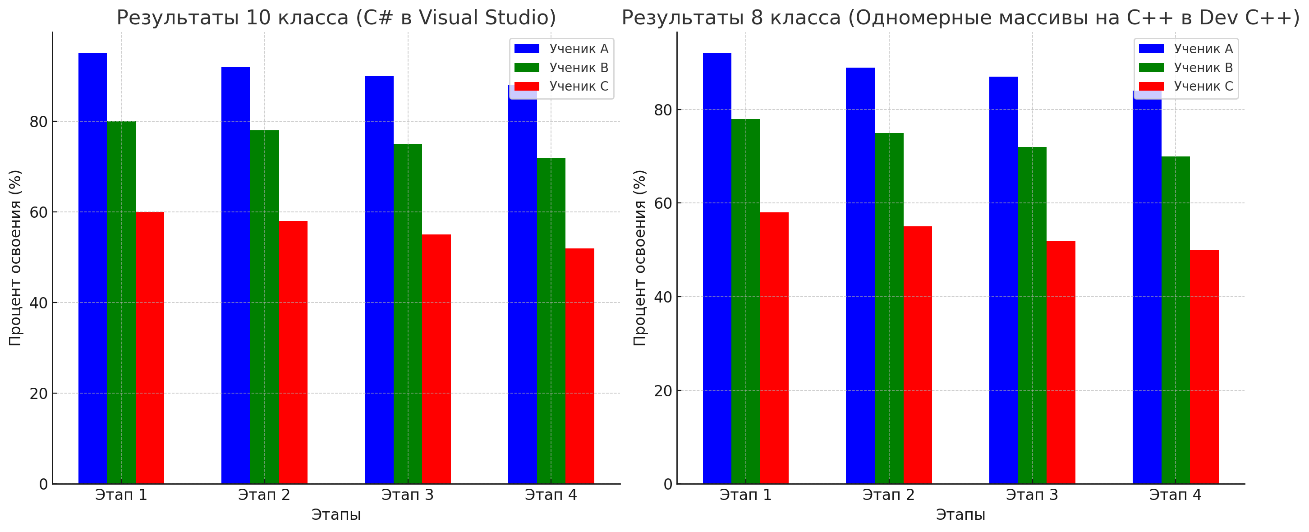
4. Самостоятельное программирование. Учащиеся самостоятельно формулировали алгоритмы и реализовывали их в коде.

Цель: проверка способности учащихся к самостоятельному анализу, планированию и реализации программных решений.

Применение: В 10 классе осуществлялась разработка проектов на языке программирования C#, включая создание простых игр и калькуляторов. В 8 классе акцент делался на написание мини-проектов, направленных на решение прикладных задач, связанных с повседневными жизненными ситуациями.

Ученик A легко справлялся с заданиями, демонстрируя уверенное владение материалом, ученик B испытывал трудности на этапе планирования алгоритма, но справлялся с реализацией кода после дополнительного разбора требований, ученик C нуждался в дополнительной поддержке на всех этапах работы кроме этапа «Создание программы по готовым инструкциям».

**Диаграмма 1.** Динамика результатов в 10 и 8 классах по информатике



Диаграммы демонстрируют прогресс учащихся в 10 и 8 классах на каждом этапе обучения. Ученик A сохранял высокий уровень освоения на всех этапах, тогда как ученики B и C показывали снижение на этапе 4 (самостоятельное программирование). Однако все учащиеся показали более высокие результаты по сравнению с классами, где не применялась данная методика.

**Обсуждение/заключение**

Результаты исследования подтверждают, что методика поэтапного усложнения задач позволяет повысить уровень программирования у всех учащихся, однако выявлены различия в уровне подготовки. Ученик C испытывал значительные трудности при переходе к самостоятельному программированию, что потребовало дополнительной адаптации через декомпозицию задач. Разделение сложных задач на более мелкие (ввод данных, их обработка, вывод результатов) позволило ученику C последовательно решать каждую часть, снижая когнитивную нагрузку и уровень тревожности. Постоянная обратная связь после завершения каждого этапа, содержащая комментарии о допущенных ошибках и рекомендации по их устранению, способствовала укреплению уверенности и развитию навыков структурирования задач у ученика C.

Ученик A, обладающий высоким уровнем способностей, активно помогал ученику C, объясняя сложные моменты и предлагая альтернативные решения, что способствовало не только прогрессу ученика C, но и углублению понимания материала самим учеником A. Это подчеркивает важность взаимного обучения и командной работы в процессе освоения программирования.

Четырёхэтапная методика позволяет оптимально организовать процесс обучения, делая его гибким и доступным для учащихся с различным уровнем академических способностей. Если исключить один из этапов, например, «Частичное программирование» или «Кодирование по алгоритму», переход к самостоятельному программированию станет слишком резким, что приведёт к значительным затруднениям, особенно у учеников группы B и C.

Проведенное исследование подтвердило эффективность методики поэтапного усложнения задач для обучения программированию. Применение данной методики позволило значительно улучшить уровень программирования учащихся 10 и 8 классов.

Основные результаты:

1. Учащиеся с высоким уровнем способностей (Ученик A) лучше адаптировались к снижению количества инструкций и быстрее переходили к самостоятельному программированию.
2. Учащиеся со средним уровнем способностей (Ученик B) демонстрировали прогресс, но требовали больше времени на освоение алгоритмов.
3. Учащиеся с низким уровнем способностей (Ученик C) столкнулись с трудностями при переходе на самостоятельное программирование, что требует дополнительной поддержки на всех этапах и адаптации методики.

Методика может быть адаптирована для различных языков программирования и уровней сложности. Для улучшения эффективности рекомендуется: увеличить количество заданий на этапе частичного программирования, добавить промежуточные задания для учащихся группы C, а также создать дополнительные мини-инструкции для сложных этапов.

В дальнейшем планируется создание адаптивных методик, учитывающих уровень подготовки учащихся, а также разработка упражнений, направленных на постепенное формирование самостоятельного мышления и программирования. Это позволит улучшить результаты и сформировать более устойчивые навыки программирования.

**Список использованной литературы**

1. Shneiderman, B. (1997). An empirical study of programming problem-solving. Proceedings of the 2nd international conference on Human-computer interaction (pp. 299–304). ACM.
2. Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books.
3. Becker, K. H., & Park, K. (2011). Integrative approaches among science, technology, engineering, and mathematics (STEM) subjects on students’ learning: A meta-analysis. Journal of STEM Education: Innovations and Research, 12(5), 23–37.
4. Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.). (2000). How People Learn: Brain, Mind, Experience, and School. National Academy Press.
5. Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172.
6. Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. Communications of the ACM, 45(4), 17–21.