УДК 519.7
DOI: http://dx.doi.org/10.21686/1818-4243-2026-2-75-85

Ш.С. Намчыкай, М.К. Тюлюш

Тувинский государственный университет, Кызыл, Россия

Развитие алгоритмического мышления при обучении основам программирования на языке C++

Цель исследования *является разработка, теоретическое обоснование и экспериментальная проверка методики обучения направленной на эффективное развитие алгоритмического мышления у школьников в процессе освоения основ программирования на языке C++. Актуальность данной работы обусловлена необходимостью совершенствования существующих подходов к обучению программированию, учитывая современные требования к качеству подготовки специалистов в сфере информационных технологий.*

Материалы и методы. *Для достижения заявленной цели исследования использовался комплекс взаимодополняющих методов, обеспечивающих всестороннее изучение проблемы и выработку эффективных рекомендаций. Были применены как теоретические, так и эмпирические подходы. Среди теоретических методов особое внимание уделялось анализу существующей научной литературы, посвящённой вопросам развития алгоритмического мышления, конкретизации исходных данных, выявлению общих закономерностей и проведению сравнительного анализа известных подходов. Важнейшими этапами являлись процессы дедукции и содержательной интерпретации полученных результатов. Эмпирическая составляющая исследования включала проведение анкетирования обучающихся общеобразовательной организации, с последующим тестированием уровня сформированности алгоритмического мышления. Это позволило объективно оценить уровень подготовленности респондентов и выявить ключевые факторы, влияющие на успешность усвоения базовых концепций программирования.*

Результаты. *В статье представлена методика, нацеленная на целенаправленное развитие алгоритмического мышления в условиях учебной среды. Структура методики представляет собой чётко организованную совокупность взаимосвязанных компонентов, включая: принципы построения эффективного образовательного процесса; цели и задачи курса, направленные на формирование необходимых навыков и способностей; стратегии обучения, предусматривающие оптимальное сочетание традиционных и инновационных форматов подачи материала, конкретизированное содержание учебного материала, ориентированное на решение типовых и нестандартных задач, организационные формы проведения занятий, способствующие активному вовлечению обучающихся в процесс познания. Особое внимание было уделено выделению критериев оценки уровней сформированности алгоритмического мышления и составлению специализированного комплекса заданий, направленных на последовательное формирование каждого отдельного элемента мыслительной структуры. Комплекс заданий разрабатывался таким образом, чтобы постепенно повышать сложность решаемых задач, обеспечивая поступательное движение от простых примеров к решению сложных проблемных ситуаций.*

Заключение. *Реализованный педагогический эксперимент показал эффективность предложенной методики. Ученики, прошедшие обучение согласно разработанному курсу, продемонстрировали значительный прогресс в развитии алгоритмического мышления, что выразилось в улучшении способности решать практические задачи, повышении уровня самостоятельности и творческого подхода к выполнению проектов. Практическая значимость исследования состоит в возможности широкого распространения опыта в образовательной среде, предлагая действенный инструмент подготовки квалифицированных преподавателей информатики и улучшения программы дополнительного образования для одарённых детей, увлечённых изучением программирования. Разработанный подход способен существенно повысить качество профессиональной подготовки специалистов в области информационных технологий, содействуя интеграции новейших достижений науки и практики в учебный процесс.*

Ключевые слова: *алгоритмическое мышление, C++, программирование, информатика, алгоритм.*

Seanchalai S. Namchykai, Marta K. Tyulyush

Tuvan State University, Kyzyl, Russia

The Development of Algorithmic Thinking when Teaching Basic Programming in C++

The purpose of the study *is to develop, theoretically substantiate, and experimentally test a teaching methodology aimed at effectively developing algorithmic thinking in schoolchildren during the process of mastering the basics of programming in the C++ language. The relevance of this paper is due to the need to improve existing approaches to teaching programming, taking into account the current requirements for the quality of training specialists in the field of information technology.*

Materials and methods. *To achieve the stated research goal, a set of complementary methods was used to ensure a comprehensive study of the problem and the development of effective recommendations. Both theoretical and empirical approaches were applied. Among the theoretical methods, special attention was paid to analyzing existing scientific literature on the development of algorithmic thinking, specifying the initial data, identifying general patterns, and conducting a comparative analysis of known approaches. The most important stages were the processes of deduction and meaningful interpretation of the obtained results. The empirical component of the study included a questionnaire survey of students of a general education institution, followed by testing the level of formation of algorithmic thinking. This allowed for an objective assessment of the respondents' level of preparedness and the identification of key factors affecting the success of mastering the basic concepts of programming.*

Results. *The article presents a methodology aimed at the purposeful development of algorithmic thinking in the conditions of the educational environment. The structure of the methodology is a clearly organized set of interrelated components, including: the principles of creating an effective educational process; the goals and objectives of the course aimed at the formation of the necessary skills and abilities; learning strategies that provide for an optimal*

combination of traditional and innovative formats of presenting the material, the specified content of the educational material focused on solving typical and non-standard tasks, organizational forms of conducting classes that promote the active involvement of students in the learning process. Special attention was paid to identifying criteria for assessing the levels of algorithmic thinking development and creating a specialized set of tasks aimed at the consistent formation of each individual element of the mental structure. The set of tasks was designed to gradually increase the complexity of the problems being solved, ensuring a progressive movement from simple examples to solving complex problem situations.

Conclusion. *The implemented pedagogical experiment has shown the effectiveness of the proposed methodology. Students trained according to the developed course demonstrated significant progress in developing algorithmic thinking, which was expressed in improving the ability to solve practical problems, increasing the level of independence, and a more creative approach to the implementation of projects. The practical significance of the research lies in the possibility for widespread dissemination of experience in the educational environment, providing an effective tool for training qualified computer science teachers and improving the additional education program for the gifted children, interested in programming. The developed approach can significantly improve the quality of professional training for information technology specialists, facilitating the integration of the latest scientific and practical achievements into the educational process.*

Keywords: *algorithmic thinking, C++, programming, computer science, algorithm.*

Введение

Современное общество нуждается в людях, способных самостоятельно ставить учебные задачи, разрабатывать методы их решения и реализации, управлять и анализировать полученные результаты, строить собственные перспективы и мнения на основе оценки различных источников информации. Необходимость переработки большого объема информации в сжатые сроки привела к изменению механизма ее восприятия, а, следовательно, памяти и мышления современных школьников. Сегодня личности должны уметь определять направление развития в новом, технологически развитом и продолжающем развиваться высокотехнологичном мире. Этот динамичный и постоянно меняющийся мир требует соответствующего осознания нового, способности адаптироваться и постоянно совершенствоваться [1].

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 [2].

В современном мире программирование становится все более важным навыком, требующимся в различных областях деятельности, начиная от разработки программного обеспечения и заканчивая анализом данных и искусственным интеллектом. Алгоритмическое мышление является основой программирования и позволяет разработчику решать сложные задачи и находить оптимальные решения [3].

В условиях цифровой трансформации общества умение мыслить алгоритмами становится критически важным навыком, необходимым для успешной карьеры в сфере ИТ. Несмотря на значительный прогресс в методиках преподавания программирования, многие обучающиеся сталкиваются с трудностями при освоении сложных концепций и принципов структурирования решений. Поэтому разработка новых подходов к обучению, направленных на развитие алгоритмического мышления, остается актуальной задачей.

Анализ отечественной и зарубежной литературы показывает, что существующие методики обучения программированию недостаточно эффективны для полноценного формирования алгоритмического мышления. Многие авторы отмечают ограниченность традиционных подходов, которые сосредоточены преимущественно на поверхностном знакомстве с синтаксисом языка программирования, игнорируя глубокую работу над развитием навыков анализа и синтеза алгоритмов.

Данная работа направлена на устранение указанных недостатков путём разработки методики обучения, способствующей эффективному развитию алгоритмического мышления у студентов в процессе освоения основ программирования на языке C++. Центральное внимание уделяется созданию системы заданий, выстроенных по принципу нарастающего уровня сложности, и детальному изучению процедур и техник структурного программирования. Помимо традиционного теоретического изложения материала, используются активные формы обучения, стимулирующие инициативность и творческий подход к решению задач.

Проектирование методики развития алгоритмического мышления в процессе обучения основам программирования на языке C++

Проектирование методики обучения основам программирования для школьников должно учитывать психологические и познавательные особенности возраста, возможности восприятия нового материала, доступность инструментальных средств и специфику школьной образовательной среды [4, 5].

Процесс проектирования методики предполагает выделение совокупности компонентов (принципы, цель и задачи, содержание, организационный и процессуальный компоненты), необходимых для организации процесса обучения основам программирования.

I. Принципы

1. Развитие алгоритмической культуры

Алгоритмическая культура включает способность правильно понимать условия задач, выбирать наиболее эффективные подходы ее решения, структурировать процесс разработки программы и оценивать её эффективность. Она включает понимание принципов построения алгоритмов, умение выбирать оптимальные методы решения проблем, способность оценивать сложность вычислений и понимать ограничения используемых методов.

2. Использование современных технологий и инструментов

Современные технологии и инструменты позволяют значительно повысить эффективность обучения. Использование интерактивных платформ, онлайн-курсов, виртуальных лабораторий и симуляторов создает комфортные условия для освоения материала и повышает мотивацию обучения.

3. Последовательность и систематичность

Изучение должно начинаться с базовых понятий и простейших конструкций языка C++, постепенно переходя к более сложным концепциям и методам. Важно соблюдать последовательность этапов, каждый последующий уровень основывается на предыдущем, создавая прочную основу для дальнейшего продвижения.

4. Интеграция теории и практики

Теоретические знания должны постоянно подкрепляться практическими занятиями. Обучающимся важно видеть связь между теоретическим материалом и его применением в реальной практике программирования. Практические задания включают создание реальных решений задач, реализованных средствами языка C++.

5. Интерактивность и обратная связь

Оценка должна проводиться регулярно и систематически, обеспечивая обратную связь с учащимся относительно уровня их подготовки и направлений дальнейшего совершенствования. Обучение должно стимулировать активное участие учеников, включать элементы интерактивности.

II. Обучающие цели и задачи.

- Овладение основными конструкциями языка C++.
- Освоение синтаксиса и семантики языка, включая базовые типы данных, операции и управляющие конструкции.
- Понимание базовых принципов программирования.
- Формирование навыка написания программ для решения прикладных задач.
- Способность понимать и анализировать программный код.
- Освоение начальных навыков анализа и синтеза алгоритмов.
- Понимание основ оценки эффективности алгоритмов и их оптимизации.
- Умение формулировать и формализовать задачи, разделяя их на подзадачи.

III. Выбор стратегии обучения

Существует несколько распространенных подходов к обучению программированию:

- Традиционный: последовательное изучение конструкций языка («от простого к сложному»).
- Задача-ориентированный: обучение начинается с небольших задач, постепенно усложняясь.

IV. Тематическая структура курса обучения.

Структуру курса целесообразно разделить на модули, каждый из которых охватывает определенный круг вопросов:

Модуль 1: Основы программирования

- Введение. Языки программирования.
- Переменные, типы данных.
- Арифметические операции.
- Ввод-вывод данных.

Модуль 2: Разветвляющиеся алгоритмы

- Логические выражения.
- Разветвляющиеся алгоритмы. Оператор ветвления (if/else)
- Разветвляющиеся алгоритмы. Оператор выбора (switch).

Модуль 3: Циклические алгоритмы

- Понятие цикла. Оператор цикла с параметром (for).
- Операторы цикла с предусловием и постусловием (while, do while)
- Суммирование рядов и перебор чисел.

Модуль 4: Массивы

- Хранение и обработка больших объемов данных.
- Одномерные, двумерные массивы.
- Двумерные массивы.

Каждый модуль должен включать практические задания, домашние задания и промежуточные контроль знаний и умений.

V. Формы и методы обучения

Методы обучения школьников должны сочетать теорию и практику:

- Лекции-презентации для объяснения сложных концепций.
- Демонстрационные примеры и упражнения.
- Индивидуальное выполнение лабораторных работ.
- Групповые соревнования и конкурсы.
- Творческие задания.

Практические занятия должны занимать большую часть времени (около 70%), так как программирование эффективно осваивается именно путем самостоятельного написания кода.

VI. Организация контроля и обратной связи

Эффективность обучения повышается благодаря регулярному контролю знаний и опера-

тивной коррекции проблемных моментов. Это может быть реализовано следующим образом:

- Текущие опросы и небольшие проверочные работы.
- Лабораторные работы, решаемые индивидуально или в группах.
- Выполнение домашних заданий.
- Итоговая аттестационная работа (создание небольшой программы, демонстрирующей владение всеми пройденными материалами).

Критерии оценки уровня развития алгоритмического мышления учащихся

Алгоритмическое мышление представляет собой способность структурировать мыслительные процессы, формировать четкую последовательность действий для решения поставленной задачи, абстрагироваться от второстепенных деталей и выделять ключевые элементы проблемы. Для объективной оценки уровня развития алгоритмического мышления необходима разработка системы критериев, позволяющих выявить степень владения основными компонентами данного типа мышления.

В данном исследовании предлагаются критерии оценки, основанные на анализе основных характеристик алгоритмического мышления, выделяемых современными исследователями [6, 7, 8, 9, 10].

Критерий 1. Абстрагирование

Это способность выделить главное в задаче, убрать несущественное и сосредоточиться на сути проблемы. Учитель обращает внимание на то, насколько ученик способен отделить второстепенную информацию от ключевой.

Уровни оценивания:

- Высокий уровень: правильно выделены ключевые компоненты задачи, дано точное описание необходимых шагов.
- Средний уровень: частично верно выделены ключевые компоненты задачи, дано точное описание необходимых шагов.
- Низкий уровень: задача воспринимается поверхностно, трудно выделить основное.

Критерий 2. Последовательность и логика рассуждений

Способность выстроить цепочку правильных логических умозаключений, позволяющую прийти к верному решению. Здесь учитывается наличие ясной последовательности действий и обоснованность выбора тех или иных операций.

Уровни оценивания:

- Высокий уровень: порядок действий продуман и последователен, каждое действие имеет обоснование.
- Средний уровень: порядок действий продуман, последовательность действий может содержать погрешности. Каждое действие имеет обоснование, но возможны небольшие неточности в обоснованиях действий.
- Низкий уровень: последовательность нарушена, отсутствуют обоснования некоторых шагов.

Критерий 3 Освоенность базовых конструкций программирования (следование, ветвление, циклы).

Уровни оценивания:

- Высокий уровень: ученики демонстрируют уверенную работу с базовыми конструкциями, успешно применяют их комбинации для выполнения поставленных задач. Представленные решения лаконичны и эффективны.
- Средний уровень: учащиеся умеют писать стандартные программы с использованием базовых конструкций, понимают их семантическую основу и могут применять их для решения несложных задач, хотя иногда допускают ошибки в организации логики.
- Низкий уровень: ученики знакомы с базовыми конструкциями программирования, способны написать простой код, содержащий одну конструкцию (например, простой цикл или простое условие). Однако возникают трудности при комбинировании разных типов конструкций или обработке сложных случаев.

Критерий 4. Гибкость мышления

Эта характеристика показывает, насколько свободно ученик подходит к поиску нестандартных решений, готов ли предложить альтернативные способы выполнения задачи.

Уровни оценивания:

- Высокий уровень: ученик предлагает разные варианты решения, видит проблему с разных сторон.
- Средний уровень: ученик демонстрирует умеренный уровень готовности предлагать нестандартные подходы и рассматривать возможные альтернативы. Хотя он способен видеть некоторые дополнительные пути решения проблемы, его идеи зачастую недостаточно разнообразны и требуют незначительной доработки или подсказок извне. Часто склоняется к одному-двум наиболее очевидным вариантам, редко проявляя инициативу искать принципиально новые методы решения задачи.
- Низкий уровень: решение однообразно, ограниченность в поиске вариантов.

Критерий 5. Ясность и точность представления

Качество письменного или устного изложения своего решения и идей. Необходимо проверить, насколько понятно и точно ученик описывает свою работу другим людям.

Уровни оценивания:

- Высокий уровень: ясное и чёткое представление мыслей, правильные пояснения и аргументы.
- Средний уровень: ясное и чёткое представление мыслей, в пояснении и аргументировании могут допускаться незначительные ошибки.
- Низкий уровень: неясное описание, пропуск важной информации, отсутствие пояснений.

Критерий 6. Возможность обобщения и переноса знаний

Показатель умения применять полученные знания в аналогичных ситуациях. Ученик должен уметь переносить ранее изученные приемы и подходы на новые задачи.

Уровни оценивания:

- Высокий уровень: эффективно решает похожие задачи, переносит знания из одной ситуации в другую.
- Средний уровень: ученик способен перенести ранее усвоенные знания и навыки на аналогичные задачи, но делает это неуверенно или эпизодически. Показывает достаточную готовность воспользоваться известными приемами, но испытывает трудности в адаптации этих приемов к новым или слегка отличающимся ситуациям.
- Низкий уровень: сложно применить знакомые техники в похожих условиях.

Система заданий, направленных на развитие алгоритмического мышления в процессе обучения основам программирования

Для реализации указанных критериев, разработана система оценочных заданий, направленных на формирование определенных аспектов алгоритмического мышления.

Критерий 1. Абстрагирование

Критерий абстрагирования важен для развития алгоритмического мышления и понимания сущности задач программирования. Для его улучшения полезно создавать задания, которые учат выделять основное и отбрасывать лишнее. *Целью таких заданий* является научить школьника видеть общую картину задачи, сосредотачиваясь на её существенных аспектах и игнорируя детали, не влияющие на конечный результат.

Ключевые характеристики заданий, направленных на формирование и закрепление навыков абстрагирования:

- Каждое задание должно содержать избыточную информацию.
- Задача формулируется таким образом, чтобы фокусироваться исключительно на основной части вопроса.
- Решением задачи должна стать простая программа, отражающая лишь главные компоненты исходной постановки.

Пример 1.

Формулировка задачи: «Напишите программу, которая выводит среднее арифметическое трёх чисел.» Дополнительная вводная информация: Учащимся говорят, что числа были получены в результате измерений температуры воздуха в течение трех дней подряд. Но фактически эта дополнительная информация никак не влияет на саму задачу.

Решение: Учащиеся пишут код, который находит среднее арифметическое без учета ненужных деталей.

```
#include <iostream>
using namespace std;
int main()
{
    float a, b, c;
    cout << “Введите три числа: “;
    cin >> a >> b >> c;
        float avg = (a + b + c) / 3;
     cout << “Среднее арифметическое равно: “ << avg << endl;
    return 0;
}
```

Критерий 2. Последовательность и логика рассуждений

Развитие критерия последовательности и логичности рассуждений крайне важно для эффективного обучения программированию. Этот критерий подчеркивает важность правильного порядка шагов, четкости аргументации и обоснования каждого этапа решения задачи. Здесь важно добиться понимания строгой последовательности действий.

Методические аспекты разработки заданий для повышения уровня логической последовательности в рассуждении школьников:

Постановка задачи с чётким указанием целей. Важно ясно сформулировать условие задачи, подчеркивая желаемый результат и критерии успеха. Если задача подразумевает использование определённых конструкций языка C++, то это тоже нужно отразить в условии.

Создание задач с заранее обозначенным порядком шагов. Предлагайте задачи, где ученики сначала сами определяют порядок необходимых шагов, а потом реализуют их в программе. Это стимулирует осознанный подход к написанию кода.

Анализ и коррекция ошибочных решений. Регулярно предлагайте разбор неправильных решений задач, заставляя детей искать причину возникновения ошибки и понимать, почему выбранный путь неверен.

Использование графических схем и диаграмм. Помогайте ученикам визуализировать процессы принятия решений и логику выполнения задач посредством блок-схем перед непосредственной реализацией на языке C++.

Пример 2.

Формулировка задачи: «Составить программу, которая запрашивает рост и вес пользователя, а затем рассчитывает индекс массы тела (ИМТ) по формуле:

ИМТ = масса/(рост×2).»

Подсказка: Порядок шагов:

Запрашиваем массу и рост.

Рассчитываем ИМТ.

Выводим результат.

Программный код:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{    double weight, height;
cout << «Введите ваш вес (кг): «;
  cin >> weight;
 cout << «Введите ваш рост (метры): «;
  cin >> height;
  double imt = weight / pow(height, 2);
cout << «Ваш индекс массы тела: « << imt
<< endl;
return 0;}
```

Критерий 3 Освоенность базовых конструкций программирования (следование, ветвление, циклы)

Данный критерий предполагает обучение языку программирования в соответствии с обучающими целями и тематической структурой курса.

Критерий 4. Гибкость мышления

Для развития гибкости мышления учеников на занятиях по языку программирования C++, важно научить их мыслить творчески и подходить к решению задач с разных сторон. Развитие гибкости мышления является важным аспектом успешного изучения программирования, особенно на школьном уровне. Для школьников важна игровая форма подачи материала, визуализация процессов и доступные практические задания.

Для тренировки гибкости мышления предлагаются следующие методические рекомендации:

1. Решение одной задачи несколькими способами.
2. Использование игровых задач.
3. Программирование с ограничениями.

Пример 3.

Составь программу, которая решает классическую игру «Ханойская башня»:

- Переложи диски с одного стержня на другой, соблюдая правила игры.
- Попробуй реализовать задачу рекурсивно и не рекурсивно.

```
#include <iostream>
using namespace std;

void hanoi_tower(int n, char from_rod, char
to_rod, char aux_rod) {
    if (n == 1) {
        cout << “Диск 1 перемещается с “ <<
from_rod << “ на “ << to_rod << endl;
        return;
    }
    hanoi_tower(n-1, from_rod, aux_rod, to_rod);
    cout << “Диск “ << n << “ перемещается с
“ << from_rod << “ на “ << to_rod << endl;
    hanoi_tower(n-1, aux_rod, to_rod, from_rod);
}

int main() {
    int disks;
    cout << “Введите количество дисков: “;
    cin >> disks;
    hanoi_tower(disks, ‘A’, ‘C’, ‘B’);
    return 0;
}
```

Критерий 5. Ясность и точность представления

Качество письменного или устного изложения своего решения и идей. Необходимо проверить, насколько понятно и точно ученик описывает свою работу другим людям. Критерий ясности и точности представления играет важную роль при изучении языка C++, поскольку хорошее изложение своих мыслей и решения задачи существенно влияет на успешность коммуникации, позволяет педагогу понять возможности ученика и определить уровень развития того или иного критерия, представленного выше. Этот критерий особенно важен при формировании навыков правильного документирования кода, пояснения шагов и обоснования принятых решений.

Основные показатели критерия ясности и точности представления:

1. Четкое описание решаемой задачи.

Ученик должен уметь ясно сформулировать цель своей работы, подробно описать исходные данные и ожидаемый результат. Это помогает избежать недопонимания и ошибок.

Пример хорошего описания задачи: «Требуется разработать функцию, принимающую два аргумента: длину прямоугольника и ширину, и возвращающую площадь фигуры.»

Пример плохого описания задачи: «Площадь прямоугольника.»

2. Логичность и упорядоченность изложения

Представление идеи или решения должно быть последовательным и стройным. Недопустимы хаотичные объяснения, пропуск важных деталей или запутанные рассуждения.

Хорошее изложение: «Вначале объявляю переменную площади и инициализирую её значением произведения длины и ширины. Далее выполняю проверку правильности полученных данных. Если проверка пройдена успешно, возвращаю значение площади.»

Плохое изложение: «Сначала умножаю, потом проверяю, а ещё надо вернуть...»

3. Понятность и доступность терминологии

Необходимо правильно употреблять термины и понятия языка программирования. Все понятия должны быть раскрыты достаточно полно и однозначно.

Правильно подобранная терминология: «Переменная length хранит длину прямоугольника, а переменная width – его ширину.»

Неправильная терминология: «Записываем в lenghth дину прямоугольника, а в witdht – ширину.»

4. Корректность и лаконичность кода

Код должен быть написан аккуратно, соблюдая общепринятые стандарты оформления.

Названия переменных и функций должны быть информативными и легко читаемыми.

Хорошо написанный фрагмент кода:

```
int calculateRectangleArea(int length, int width) {
    return length * width;
}
```

Плохо написанный фрагмент кода:

```
int ca(int l,int w){return l*w;}
```

5. Правильное комментирование и документация

Комментарии должны дополнять код, раскрывая важные детали реализации и помогая другому разработчику быстро разобраться в структуре программы.

Полезные комментарии:

```
// Функция принимает длину и ширину прямоугольника и возвращает его площадь
int rectangleArea(int length, int width) {
    // Умножаем длину на ширину и возвращаем результат
    return length * width;
}
```

Нуждаются в улучшении комментарии:

```
// Площадь равна длине,  умноженной на ширину
int area(int len, int wid){
    return len*wid; // возращаем площадь}
```

Пример 4. Оптимизировать программный код (уменьшить размер занимаемой программой памяти).

Для оптимизации программы с точки зрения уменьшения размера занимаемой памяти давайте рассмотрим простейшую задачу: подсчёт суммы первых N натуральных чисел. Мы покажем две версии программы – первую неэффективную версию и её улучшенную версию, оптимизированную по размеру потребляемой памяти.

Неэффективная версия (больше занимает память):

Здесь мы будем хранить каждое число в отдельном массиве, что потребует дополнительной памяти:

```
#include <iostream>
using namespace std;
int main() {
    const int MAX_N = 1000000; // Максимально возможное значение N
    long long sum[MAX_N];          // Массив для хранения всех сумм
    int n;
    cout << «Введите количество чисел: «;
    cin >> n;
    // Заполняем массив суммами
    sum[0] = 0;
    for(int i = 1; i <= n; ++i) {
        sum[i] = sum[i-1] + i;
    }
    cout << «Сумма первых « << n << « чисел равна « << sum[n] << endl;
    return 0;
}
```

Проблема здесь очевидна: создаётся массив большого размера даже тогда, когда нам фактически нужен лишь итоговый результат.

Оптимизированная версия (меньше занимает памяти):

Мы можем легко решить задачу без выделения дополнительного массива, используя переменную для накопления промежуточных результатов:

```
#include <iostream>
using namespace std;
int main() {
    long long totalSum = 0;      // Здесь храним сумму
    int n;
    cout << «Введите количество чисел: «;
    cin >> n;
    // Вычисляем сумму последовательно
    for(int i = 1; i <= n; ++i) {
        totalSum += i;
    }
    cout << «Сумма первых « << n << « чисел равна « << totalSum << endl;
    return 0;
}
```

Критерий 6. Возможность обобщения и переноса знаний

Данный критерий касается способности ученика применять приобретённые знания и навыки программирования на языке C++ в новых и незнакомых условиях. Его важность обусловлена необходимостью подготовки ученика к будущим задачам, отличающимся от тех, которые были рассмотрены в ходе обучения.

Для развития способности применять знания программирования в новых условиях необходимо формировать и развивать:

1. *Способность распознавать сходства и различия между новыми и старыми задачами.*

Ученик должен научиться определять общие черты и отличия между известными типами задач и теми, с которыми сталкивается впервые. Чем точнее и быстрее он замечает подобие, тем легче ему перенести накопленные знания и приёмы решения.

Пример 5. Ученику была известна задача по поиску минимального элемента массива. Новая задача состоит в поиске минимальных элементов в нескольких массивах. Поиск минимального элемента можно оформить в виде функции пользователя.

2. Умение комбинировать ранее изученные элементы для решения новых задач

Способность комбинировать фрагменты известных решений, структур данных и алгоритмов для адаптации к новым условиям свидетельствует о высоком уровне владения предметом.

Пример 6. Задача состоит в подсчёте количества положительных чисел в массиве. Ученик уже знает, как пройти по массиву (циклы), как

проверять условие («if»), как хранить промежуточные результаты (переменные). Эти знания позволяют ему быстро составить новое решение.

3. Наличие осознанных моделей поведения при столкновении с трудностью

При встрече с новой задачей ученик должен демонстрировать разумную стратегию преодоления трудностей. Это может включать попытку проанализировать проблему, разделить её на части, рассмотреть похожие задачи, попробовать разные подходы и оценить их преимущества и недостатки.

Примеры хороших действий:

- Перед началом решения новой задачи записываются заметки и проводится предварительный анализ.
- Строится таблица соответствий между старой и новой задачей.
- Создается псевдокод или черновой набросок будущего решения.

Реализация методики развития алгоритмического мышления в процессе обучения основам программирования

Проведение эксперимента по апробации методики обучения основам программирования, направленной на развитие алгоритмического мышления, предполагает тщательную организацию процесса, фиксацию результатов и последующий анализ полученных данных. Экспериментальная работа была направлена на проверку эффективности методики в процессе обучения основам программирования в 10 классах в течение учебного года.

Выделим этапы проведения эксперимента.

1. Постановка целей и задач эксперимента

Цель данной экспериментальной работы заключается в проверке эффективности методики, направленной на формирование и развитие алгоритмического мышления учащихся в процессе изучения основ программирования.

Задачи:

- Проверка гипотезы о влиянии методики на формирование и развитие компонентов алгоритмического мышления, улучшение уровня навыков программирования.
- Установление динамики изменений в показателях уровня знаний и успеваемости учащихся.
- Оценка удовлетворенности участников экспериментальным процессом.

2. Определение выборки участников

Эксперимент проводился в двух параллельных десятых классах средней общеобразовательной школы № 1 города Шагонар Республики Тыва. Класс 10А (контрольная группа) проходил традиционное обучение по стандартной программе, в классе учится 25 учащихся. Класс 10Б (экспериментальная группа) обучался на основе разработанной методики, в классе – 24 ученика.

3. Предварительное тестирование

Перед началом эксперимента проводилась диагностика уровня начальной подготовленности учащихся обеих групп. Этот этап необходим для выявления стартовых условий и устранения возможных различий в уровне первоначальных знаний и навыков.

Использовались стандартные диагностические инструменты:

- Анкетирование по уровню самоощущения готовности к изучению программирования.
- Тесты на знание основ программирования и понимание базовых алгоритмов.

Предварительное тестирование показало, что у контрольной и экспериментальной групп нет больших отличий в начальном уровне подготовленности по программированию.

4. Реализация экспериментальной части

Каждая группа изучала основы программирования согласно соответствующей программе:

- Экспериментальная группа работала по новой методике, используя специальные подходы и формы обучения, ориентированные на развитие алгоритмического мышления.
- Контрольная группа продолжала занималась по стандартной программе обучения.

Эксперимент длился в течение учебного года.

5. Сбор данных и мониторинг прогресса

Во время проведения эксперимента фиксировались ряд показателей:

- Количество успешно выполненных задач каждым участником.
- Время, затраченное на решение стандартных тестовых заданий.
- Качество выполняемых заданий (оценивается преподавателем).
- Уровень вовлеченности и заинтересованности учащихся (анкетирование, наблюдения преподавателя).
- Фиксировалась информация о трудностях и проблемах, возникающих у учащихся в ходе занятий.

6. Постэкспериментальное тестирование

По завершении основного этапа вновь проводились тесты, контрольные работы и анкетирования, позволяющие оценить изменения в уровнях развития алгоритмического мышления и подготовки учащихся.

Основными метриками измерения являлись нами выделенные критерии развития алгоритмического мышления.

Чтобы вычислить среднее значение по критерию для каждого уровня были установлены коэффициенты значимости: высокий уровень – коэффициент значимости 3, средний уровень – 2, низкий уровень – 1. Исходя из этого, были подсчитаны средние коэффициенты критериев для контрольной и экспериментальной групп.

Средний коэффициент критерия 1 (уровень

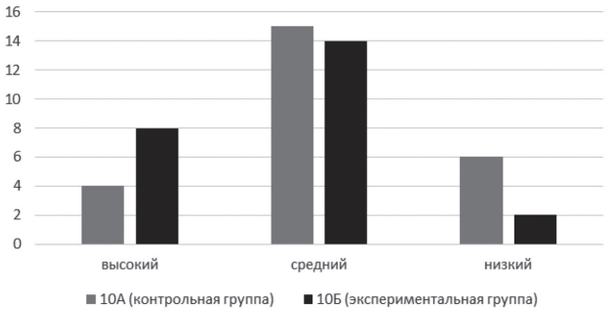

Рис. 1. Уровень абстрагирования в контрольном и экспериментальном классе

Fig. 1. The level of abstraction in the control and experimental classes

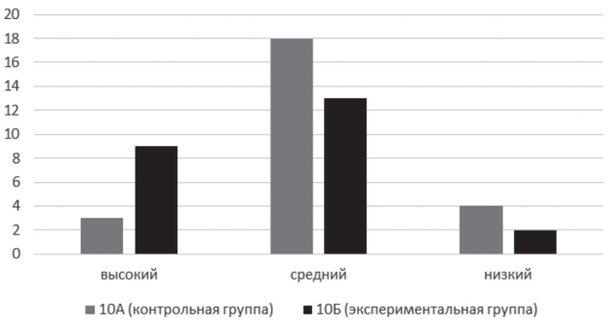

Рис. 2. Показатели критерия 2 (последовательность и логика рассуждений) контрольного и экспериментального классов

Fig. 2. Indexes of criterion 2 (consistency and logic of reasoning) for the control and experimental classes

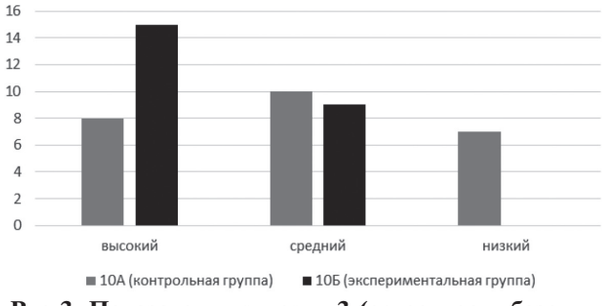

Рис 3. Показатели критерия 3 (освоенность базовых конструкций программирования) контрольного и экспериментального классов

Fig. 3. Indexes of criterion 3 (mastering basic programming constructs) for control and experimental classes

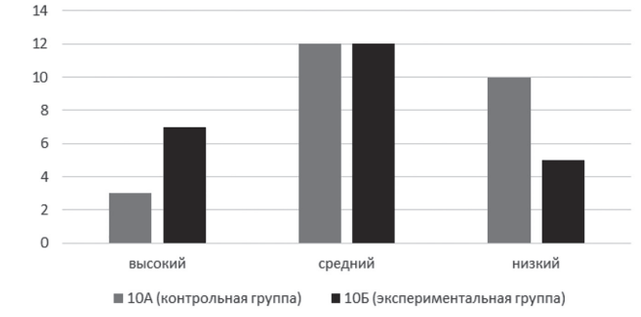

Рис. 4. Показатели критерия 4 (гибкость мышления) контрольного и экспериментального классов

Fig. 4. Indexes of criterion 4 (flexibility of thinking) for the control and experimental classes

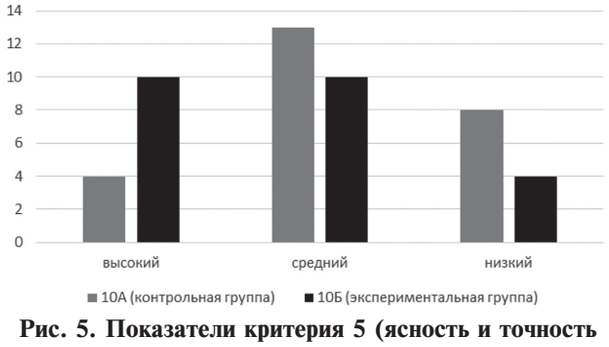

Рис. 5. Показатели критерия 5 (ясность и точность представления) контрольного и экспериментального классов

Fig. 5. Indexes of criterion 5 (clarity and accuracy of presentation) for the control and experimental classes

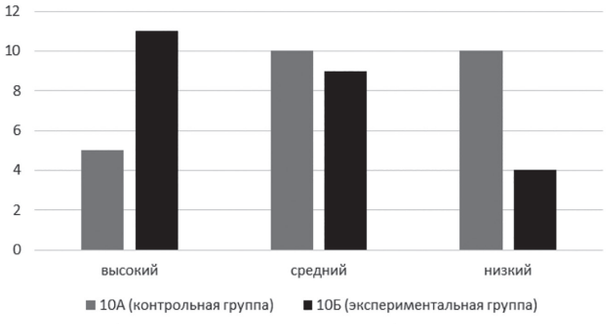

Рис. 6. Показатели критерия 6 (возможность обобщения и переноса знаний) контрольного и экспериментального классов

Fig. 6. Indexes of criterion 6 (ability to generalize and transfer knowledge) for control and experimental classes

абстрагирования) в контрольной группе – 1,92, в экспериментальной группе – 2,25 (рис.1).

Средний коэффициент критерия 2 (последовательность и логика рассуждений) в контрольной группе – 1,96, в экспериментальной группе – 2,29 (рис.2).

Средний коэффициент критерия 3 (Освоенность базовых конструкций программирования) в контрольной группе – 2,04, в экспериментальной группе – 2,63 (рис. 3).

Средний коэффициент критерия 4 (Гибкость мышления) в контрольной группе – 1,72 , в экспериментальной группе – 2,0 (рис. 4).

Средний коэффициент критерия 5 (Ясность и точность представления) в контрольной группе – 1,84 , в экспериментальной группе – 2,25 (рис. 5).

Средний коэффициент критерия 6 (Возможность обобщения и переноса знаний) в контрольной группе – 1,8, в экспериментальной группе – 2,2 (рис.6).

Для измерения динамики развития алгоритмического мышления только в экспериментальной группе были зафиксированы средние

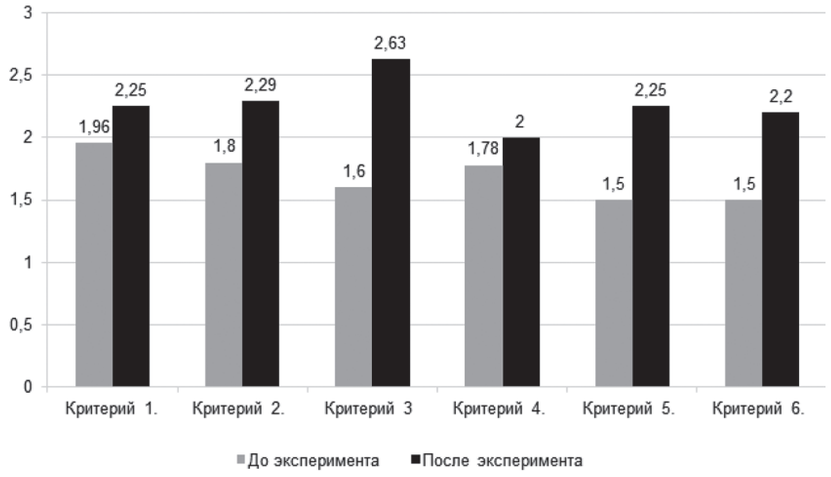

Рис. 7. Результаты измерения средних значений критериев в экспериментальном классе до и после эксперимента

Fig. 7. Results of measuring the average values of the criteria in the experimental class before and after the experiment

значения шести критериев до и после эксперимента (рис. 7). Результаты измерений показали положительную динамику роста по всем шести критериям.

Можно сделать вывод, что применение методики действительно оказывает положительное воздействие на развитие алгоритмического мышления учеников, навыков программирования, а также качество знаний и успеваемость учащихся по информатике.

Заключение

Исходя из результатов эксперимента, можно утверждать:

- о наличии более высоких показателей критериев алгоритмического мышления в экспериментальной группе по сравнению с контрольной;
- о более высоком уровне умений и навыков программирования в экспериментальной группе;
- о повышении качества знаний и успеваемости в экспериментальной группе.

Реализация разработанной методики показала свою эффективность в ходе педагогического эксперимента. Основными метриками измерения являлись выделенные критерии развития алгоритмического мышления, уровни которых были измерены в контрольной и экспериментальной группах. Анкетирование выявило положительные отзывы учащихся о готовности самостоятельно решать задачи по программированию, о высоком уровне уверенности в выполнении практических заданий.

Опыт показывает, что учащиеся демонстрируют значительное улучшение способности формулировать и реализовывать алгоритмы, улучшая качество выполняемых заданий и уменьшая количество ошибок.

Разработанная методика ориентирована на комплексный подход к обучению основам программирования на языке C++ и направлена на систематическое развитие алгоритмического мышления. Она включает продуманную систему заданий и контрольных мероприятий, позволяющих последовательно совершенствовать навыки учащихся. Практическое применение продемонстрировало её высокую эффективность, выражающуюся в повышении качества обучения и успеваемости учащихся.

Литература

1. Артемова Е. В. Развитие алгоритмического мышления путем решения задач по программированию // V Всероссийская научно-практическая конференция «Образование. Технологии. Качество.» (Саратов, 26 марта 2021 г.). М.: Издательство «Перо», 2021. С. 19–23.

2. Фомичева Е. И. Формирование алгоритмического мышления учащихся при обучении объектно-ориентированному программированию // Современное образование Витебщины. 2019. № 4(26). С. 23–25.

3. Калитина В.В. Алгоритмические ментальные карты эффективное средство обучения программированию // Международная научно-практическая конференция «Фундаментальные и прикладные научные исследования» (Москва, 17 мая 2015 г.). М.: ООО «Европейский фонд инновационного развития», 2015. С. 179–181.

4. Беспалько В. П. Слагаемые педагогической технологии. М.: Педагогика, 1989. 192 с.

5. Изместьев Н. С. Особенности формирования и развитие алгоритмического мышления с помощью объектно-ориентированного программирования // Конференция «Современные тенденции и проекты развития информационных систем и технологий». 2015. № 4(13). С. 167–172.

6. Блинова О. С. Методики диагностики уровня развития алгоритмического мышления у младших школьников // Вестник Томского государственного педагогического университета. Серия: Психолого-педагогические науки. 2021. Т. 23. № 4. С. 56–64.

7. Грибова Н. П. Критерии оценивания компетенций алгоритмического мышления будущих педагогов информатики // Информатика и образование. 2022. № 6. С. 45–51.

8. Абрамян Е. А. Формирование и оценка алгоритмической культуры школьников средствами программирования // Педагогическое образование и наука. 2020. № 8. С. 114–121.

9. Дмитриев А. И. Оценка сформированности элементов алгоритмического мышления у обучающихся технических вузов // Высшее образование в России. 2023. № 3. С. 111–119.

10. Фадеева С. Д. Программирование как основа развития алгоритмического мышления и саморазвития личности // Педагогических исследований. 2020. № 6. С. 29–32.

References

1. Artemova Ye. V. Development of algorithmic thinking by solving programming problems. V Vserossiyskaya nauchno-prakticheskaya konferentsiya «Obrazovaniye. Tekhnologii. Kachestvo.» = V All-Russian scientific and practical conference «Education. Technologies. Quality.» (Saratov, March 26, 2021). Moscow: Pero Publishing House; 2021: 19-23. (In Russ.)

2. Fomicheva Ye. I. Formation of students' algorithmic thinking when teaching object-oriented programming. Sovremennoye obrazovaniye Vitebshchiny = Modern education of the Vitebsk region. 2019; 4(26): 23-25. (In Russ.)

3. Kalitina V.V. Algorithmic mental maps are an effective tool for teaching programming. Mezhdunarodnaya nauchno-prakticheskaya konferentsiya «Fundamental'nyye i prikladnyye nauchnyye issledovaniya» = International scientific and practical conference «Fundamental and Applied Scientific Research» (Moscow, May 17, 2015). Moscow: European Foundation for Innovation Development LLC; 2015: 179-181. (In Russ.)

4. Bespal'ko V. P. Slagayemyye pedagogicheskoy tekhnologii = Components of Pedagogical Technology. Moscow: Pedagogy; 1989. 192 p. (In Russ.)

5. Izmest'yev N. S. Features of the Formation and Development of Algorithmic Thinking with the Help of Object-Oriented Programming. Konferentsiya «Sovremennyye tendentsii i proyekty razvitiya informatsionnykh sistem i tekhnologiy» = Conference «Modern Trends and Projects for the Development of Information Systems and Technologies». 2015; 4(13): 167–172. (In Russ.)

6. Blinova O. S. Methods for Diagnostics of the Level of Development of Algorithmic Thinking in Primary School Students. Vestnik Tomskogo gosudarstvennogo pedagogicheskogo universiteta. Seriya: Psikhologo-pedagogicheskiye nauki = Bulletin of Tomsk State Pedagogical University. Series: Psychological and Pedagogical Sciences. 2021; 23; 4: 56–64. (In Russ.)

7. Gribova N. P. Criteria for Assessing the Algorithmic Thinking Competencies of Future Computer Science Teachers. Informatika i obrazovaniye = Computer Science and Education. 2022; 6: 45–51. (In Russ.)

8. Abramyan Ye. A. Formation and Assessment of Schoolchildren's Algorithmic Culture Using Programming Tools. Pedagogicheskoye obrazovaniye i nauka = Pedagogical Education and Science. 2020; 8: 114-121. (In Russ.)

9. Dmitriyev A. I. Assessment of the Development of Elements of Algorithmic Thinking in Students of Technical Universities. Vyssheye obrazovaniye v Rossii = Higher Education in Russia. 2023; 3: 111–119. (In Russ.)

10. Fadeyeva S. D. Programming as a Basis for the Development of Algorithmic Thinking and Personal Self-Development. Pedagogicheskikh issledovaniy = Pedagogical Research. 2020; 6: 29-32. (In Russ.)

Сведения об авторах

Шончалай Саяновна Намчыкай
Тувинский государственный университет,
Кызыл, Россия
Эл. почта: nshonchalay@mail.ru

Марта Кан-Ооловна Тюлюш
Тувинский государственный университет,
Кызыл, Россия
Эл. почта: oorshak_2010@mail.ru

Information about the authors

Seanchalai S. Namchykai
Tuvan State University,
Kyzyl, Russia
E-mail: nshonchalay@mail.ru

Marta K. Tyulyush
Tuvan State University,
Kyzyl, Russia
E-mail: oorshak_2010@mail.ru