Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет Бизнес-информатики
Отделение Прикладной математики и информатики
Кафедра Анализа данных и искусственного интеллекта

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Создание учебного курса по алгоритмам для школьников с использованием системы Кумир

Выполнила студентка группы 471
Евдокимова Валерия Юрьевна

Научный руководитель:
Доктор ф.-м. н., профессор,
Ройтберг Михаил Абрамович

Москва 2014

Содержание

Аннотация....3
Abstract....3
Введение....4
Глава 1.Выбор языка программирования....7
1.1Выбор парадигмы для языка программирования....7
1.2Языки с императивной парадигмой....9
1.2.1Язык C++....9
1.2.2Язык Python....9
1.2.3Язык КуМир....10
Результаты по главе 1....12
Глава 2. Тестирующая система....13
2.1Существующие интернет-ресурсы с интегрированными системами автоматического тестирования....13
2.2Существующие автономные тестирующие системы....14
2.3Составляющие тестирующей системы....15
2.3.1Набор тестов....15
2.3.2Модельное решение....15
2.3.3Генераторы....16
2.3.4Валидаторы....17
2.3.5Проверяющая программа....17
2.3.6Дублирующие и заведомо неверные решения....18
2.4Интеграция тестов в систему КуМир....19
Результаты по главе 2....20
Глава 3.Структура курса 21
3.1Арифметические операции....21
3.2Условный оператор....21
3.3Циклы....21
3.3.1Циклы «Для»....22
3.3.2Циклы «Пока»....22
3.4Типы данных....22
3.4.1Символы и строки....22
3.4.2Вещественные числа....22
3.5Одномерные массивы....23
3.6 Поиск в массиве....23
3.7Сортировка....23
3.8Рекурсия....23
3.9Игры....24
3.10Динамическое программирование....24
Результаты по главе 3....24
Глава 4.Применение курса на практике....24
4.1Использование курса учеником....25
4.2Использование курса учителем....27
Результаты по главе 4....28
Глава 5.Результаты работы....29
Результаты по главе 5....30
Заключение....31
Библиография....32
Приложения....33

Аннотация

Целью данной работы является разработка полного и удобного комплекса для школьников, позволяющего помочь им в изучении программирования в области алгоритмов. Курс не должен требовать как математической, так и программистской подготовки. Он должен помогать в освоении базовых понятий и/или в подготовке к Единому государственному экзамену.

Abstract

The purpose of this work is to develop a comprehensive and comfortable environment for russian high school students allowing them to study algorithms and basics of Computer Science. The course should require neither deep mathematical background nor programming experience. It should help understanding basics and/or prepare for the Unified State Exam in Informatics.

Введение

Современный человек не представляет своей жизни без компьютера. Он используется в учёбе, работе, для развлечений. Несмотря на то, что большинство людей являются простыми пользователями компьютера, с каждым годом все больше и больше людей стремятся изучить компьютер более подробно, освоить базовые навыки программирования для тех или иных целей. Безусловно, любое изучение программирования должно начинаться с самых базовых понятий и алгоритмов.

В современном мире существует большое количество литературы, лекций и других теоретических курсов, посвящённых изучению программирования на самых разных уровнях. Изучение программирования с помощью литературы обычно не предполагает выполнение упражнений на практике, или же данные упражнения никак не могут быть проверены. Кроме разнообразной литературы, существует множество различных курсов по обучению программированию.В том числе, такие курсы существуют для тех, кто только начал изучать программирование. Зачастую такие курсы, кроме теоретических знаний, включают упражнения и примеры. Однако такие курсы, если они не проводятся очно под руководством опытного преподавателя, не имеют обратной связи: ученик не знает, решил ли он задачу, есть ли в его решении что-то неоптимальное, и так далее.

Поскольку на подобных курсах иногда отсутствует личный контакт ученика и преподавателя, преподаватель не имеет возможности проверить корректность программы, написанной учеником. В таких случаях часто используются системы автоматического тестирования задач. Опыт использования автором подобных показывает, что это очень удобный способ проводить занятия во многих видах. Такие системы используются для подготовки к олимпиадам различных уровней, на кружках по обучению программированию, в различных профильных выездных школах, например Летней компьютерной школе, и во многих других школах. Отдельной задачей является подбор заданий под конкретный кружок или группу.

Однако разворачивание такой тестирующей системы вместе с подбором заданий может оказаться очень трудоёмким, особенно при отсутствии соответствующего опыта в настройке специфичного программного обеспечения.

В данной работе автор пытается обобщить опыт обучения программированию, ориентированный на несколько категорий учеников. Во-первых, это школьники 5–8 классов, не изучавшие в школе программирование, но заинтересовавшиеся этой темой, например, в результате посещений кружков по математике. Во-вторых, это школьники старших классов, которые хотят дополнить основной школьный курс и ускорить усвоение материала, самостоятельно или, предпочтительнее, на кружках с хорошей теоретической подготовкой. В-третьих, это школьники выпускных классов, которые зачастую в последние месяцы осознают необходимость подготовиться к Единому государственному экзамену по информатике, и желающие изучить программирование как важную часть тем этого экзамена.

Вместе с этим, курс становится автономной системой, не требующей особых навыков для её развёртывания. Таким образом, курс предполагается использовать всеми преподавателями, обладающими соответствующими методическими умениями, для автоматизации и облегчения практической части процесса обучения.

Таким образом целью данной работы автор видит создание практического курса для обучения программированию, который мог бы быть использован желающими для изучения основ программирования.

Основными задачами является подбор заданий для данного курса, подготовка для каждого задания определённого набора материалов и создание тестирующей системы, отвечающей определённым требованиям.

В результате работы поставленные задачи были практически выполнены, хотя существуют некоторые аспекты, в которых требуется доработка.

Работа имеет следующую структуру: работа состоит из 5 глав. В первой главе описан процесс выбора инструментария для решения поставленных задач. Объяссняется, почему был выбран именно язык программирования КуМир. Во

второй главе описывается механизм тестирования программ, написанных пользователями курса, Описывается механизм подготовки данных для этого тестирования. В третьей главе описана структура созданного курса с кратким описанием каждой из включённых тем. Четвёртая глава описывает процесс использования созданного курса учителем и учеником. В пятой главе приведены результаты работы.

Глава 1. Выбор языка программирования

Одной из важных задач был выбор основного языка программирования для автоматизированной системы курса. На её решение повлияли многочисленные факторы.

1.1 Выбор парадигмы для языка программирования

Сначала нужно было определиться с выбором парадигмы программирования, в которой изучать курс. Были рассмотрены логические, функциональные, объектно-ориентированные языки. Логические и функциональные языки менее распространены и ориентированы, в первую очередь, на математические задачи и математическую подготовку. Они сложнее для анализа и для преподавания школьниками и менее универсальны при переходе. Объектно-ориентированный же подход требует зачастую больших объёмов как теоретического материала, так и написания кода. Можно это проиллюстрировать длиной стандартного кода для решения простейшей задачи — в Java код, складывающий два числа и выводящий их на экран требует как минимум явного создания одного класса и нескольких экземпляров классов. В то же время, языки, которые либо не позволяют объектно-ориентированный подход, либо позволяют скрыть всё, что с ним связано, когда оно не нужно пользователю, позволяют обойтись двумя вызовами функций, не считая, в обоих случаях, арифметических операций.

Таким образом, предпочитаемой парадигмой языка был императивный подход. Императивные языки ближе всего к пониманию блок-схем, входящих в программы информатики большинства школ, даже если программирование как таковое там не изучается — базовое понятие алгоритма иллюстрируется именно в императивном ключе.

Во-вторых, язык должен быть лёгким для изучения при полном или почти полном отсутствии теоретической базы как в программировании, так и в математике. Это обусловлено целевой аудиторией курса — школьники ещё не обладают чёткостью математического мышления, которое вырабатывается в

высших учебных заведениях. На лёгкость изучения оказывают влияние различные факторы. Это и сложность синтаксиса, и стандартность управляющих конструкций, и мощность и простота стандартной библиотеки.

Во-третьих, язык, компилятор или интерпретатор, среда и прочие необходимые компоненты должны быть свободно распространяемыми. Более того, лицензии, позволяющие модифицировать эти компоненты, являются ещё более удачным выбором, поскольку это позволяет реализовывать дополнительные проверки напрямую на уровень интерпретатора, либо в стандартную библиотеку языка на этапе компиляции.

Было произведено сравнение набора языков по приведённым выше принципам. Объективные данные, приведённые в таблице, включают: парадигму языка программирования, лингвистическую базу языка, если это применимо, наличие свободных реализаций языка.

Таблица 1.1 Сравнение языков программирования

Язык	Сложность изучения	Лингвистическая база	Парадигма	Свободные реализации
C	выше среднего	английская	императивная	GPL
C++	выше среднего	английская	объектно-ориентированная	GPL
Haskell	высокая	отсутствует	функциональная	GPL
Lisp	высокая	отсутствует	функциональная	GPL
Java	высокая	английская	объектно-ориентированная	разрабатываются
Python	низкая	английская	Объектно-ориентированная, функциональная, императивная	GPL
КуМир	низкая	русская	императивная	GPL
Pascal	низкая	английская	императивная	GPL
1C	средняя	русская	императивная	нет

1.2 Языки с императивной парадигмой

Рассмотрим конкретные языки, поддерживающие императивную парадигму программирования, и сравним их.

Во многих школах фактическим стандартом обучения на сегодняшний день оказывается Pascal в различных его реализациях: Delphi, Free Pascal, PascalABC. Среди его плюсов — распространённость, относительная простота изучения на базовом уровне, естественный подход к работе со строками. Однако этот язык одновременно и не является полностью учебным — в частности, он компилируется, делается значительный акцент на низкоуровневое программирование, управляющие конструкции громоздки, — так и не является языком, который в перспективе может оказаться языком реальных приложений в профессиональной деятельности. Промышленное программирование в современном мире использует более современные языки, такие как Java, C#, Python на высоком уровне и C на низком.

1.2.1 Язык C++

C++ — универсальный язык, позволяющий как изучать и применять императивную парадигму, так и объектно-ориентированную, а в свете последних расширений стандарта (C++11, C++14), и функциональную. Это фактический стандарт в разработке высокопроизводительных и сложных систем. Однако большие возможности позволяют и совершить больше ошибок, на поиск которых у начинающего уходит очень много времени. К таким особенностям можно отнести отсутствие встроенной проверки границ массивов, возможность произвольного доступа к памяти. Также понимание отдельных концепций, как-то указатели, ссылки, ASCIIZ-строки, требует значительной теоретической подготовки. Таким образом, C++ — не самый подходящий для обучения на базовом уровне язык.

1.2.2 Язык Python

Python — ещё один современный язык, совмещающий в себе различные концепции и парадигмы. Являясь языком исключительно высокого уровня, в отличие от C++, он не позволяет совершить значительную часть ошибок с

уровнями доступа. Также Python обладает лёгким синтаксисом с минималистичными управляющими конструкциями, что делает его более наглядным. В частности, автор курса уже не первый год преподает школьникам программирование в Летней компьютерной школе с использованием этого языка. Тем не менее, некоторые проблемы у языка все равно есть. Во-первых, язык предоставляет очень хорошую стандартную библиотеку, которая собирает универсальные средства для решения множества задач. Это демотивирует ученика реализовывать свои собственные или изученные алгоритмы. Во-вторых, Python является быстро развивающимся языком с хорошим сообществом, однако большинство его применений весьма практичны: это удобный скриптовой язык, он применяется как для повседневной аналитики, так и для создания сайтов и для других целей. Учебным языком на данный момент он считается в меньшей степени, поэтому в таком качестве его не развивают.

1.2.3 Язык КуМир

Исследование показало, что оптимальным языком для поставленных задач является КуМир. Во-первых, он исходно ориентирован на школьников, изучающих программирование. Это находит своё отражение и в русскоязычной базе — управляющие конструкции обозначены производными русских слов, а кириллические идентификаторы поддерживаются и являются естественными для языка программирования. Это полезно, потому что повышает мнемоничность для человека, родным языком которого является русский, а также не требует дополнительных усилий на разъяснение происхождений отдельных терминов при обучении. Императивная парадигма языка является наиболее естественной для обучения школьника, который на начальном этапе почти наверняка изучили блок-схемы, по сути являющиеся отражением инструкций императивной программы. Управляющие конструкции языка полностью аналогичны стандартным, встречающимся в языках программирования, начиная от Алгола (1958) и C (1972) и заканчивая императивными частями реализаций наиболее современных версий Python (2014). Дополнительным плюсом является разработанный авторами

интерпретатора комплекс, позволяющий объединять задачи в единый комплекс с встроенной навигацией между его элементами, а также дающий возможность включать методические рекомендации и учебные материалы наряду с текстами заданий. Что касается лицензии — GNU Public License (http://gnu.org/licenses/), по которой распространяется интерпретатор и весь комплекс КуМир, позволяет разрабатывать любые продукты на основе существующего исходного кода при условии сохранения лицензии, а также при условии распространения исходного кода вместе с продуктом. Это исключает возможные проблемы с получением необходимых для запуска комплекса программных продуктов. Кроссплатформенность даёт возможность использовать продукт все большему количеству людей, учитывая, что внедрение свободных операционных систем как в школах и других учебных заведениях, так и на личных компьютерах школьников, принимает все большие масштабы.

В целом, КуМир (расшифровывается как Комплект Учебных Миров) представляет собой не только язык программирования, разработанный в НИИСИ РАН. Это целая система, предназначенная для обучения программированию в школе. В системе КуМир использован алгоритмический язык, основанный на русской лингвистической базе. Интерпретация реализована таким образом, что все возможные ошибки тщательно анализируются и не приводят к падениям программ. Кроме того, сообщения обо всех ошибках в очень доступной форме выводятся на полях редактора напротив той строки, где была совершена соответствующая ошибка. Это позволяет пользователю быстро сориентироваться в написанной им программе и оперативно исправить все допущенные ошибки. Также в интерпретатор встроен удобный отладчик, соответствующий общепризнанным стандартам. Он позволяет пошагово выполнять программу и с помощью специальных средств наблюдать за тем, как изменяются значения переменных, которые были использованы в программе.

Популярности КуМир для обучения способствует и наличие различных исполнителей, которые позволяют изучать программирование не только в привычном виде программ, считывающих и выводящих символьные и

числовые данные, но и в более игровой форме, управляя виртуальным исполнителем-роботом. Кроме этого в системе существует исполнитель Водолей, который может быть применён для решения широкого класса математических задач про переливание и отмеривание определенного количества жидкости. Существует также гораздо боле простой в своем использовании комплекс ПиктоМир, основанный на платформе КуМир. Данный учебный комплекс ПиктоМир позволяет изучать программирование, собирая программу из готовых блоков, что актуально для учеников самых младших классов.

Исходя из всего вышеописанного, можно сделать вывод, что система КуМир является наиболее удобной начального изучения программирования. Но, к сожалению, в данной системе есть некоторые недостатки. Например, некоторые базовые функции языка программирования имеют английские названия: функция взятия остатка от деления одного числа на другое называется div, а нахождение модуля числа -- abs. Такие названия вызывают сложности у школьников, поскольку они вынуждены просто заучивать данные команды, что очень часто приводит к путанице. В то время как русские названия команд позволяют пользователю вспомнить название и синтаксис команды исходя из тех действий, которые эта команда должна выполнять.

Результаты по главе 1

1. Проведён сравнительный анализ языков программирования, их достоинств и недостатков для реализации поставленных задач.
2. Обоснован выбор системы КуМир для решения поставленных задач.
3. Дано подробное описание системы КуМир.

Глава 2. Тестирующая система

В современной практике обучения программированию широкое распространение получили тестирующие системы, с помощью которых производится автоматическое тестирование кода программ и проверка их на правильность. Плюсы такого подхода очевидны: это и получение обратной связи учеником в реальном времени, и экономия времени преподавателя на формальное тестирование программ путём ввода больших объёмов входных данных вручную, и уменьшение вероятности ошибки при проверке. Следует, однако, отметить, что автоматизированное тестирование никогда полностью не может заменить анализ кода совместно с преподавателем, потому что такой анализ позволяет оценить не только формальный факт того, что программа решает поставленную задачу, но и оптимальность кода, слабо формализуемый стиль программирования и понятность написанной программы. Поэтому тестирующая система является лишь инструментом преподавателя и никогда не сможет стать его полноценной заменой.

2.1 Существующие интернет-ресурсы с интегрированными системами автоматического тестирования

На данный момент существует множество тестирующих систем, интегрированных в различные онлайн сервисы, позволяющих автоматически проверять задачи. Некоторые из сайтов с заданиями насчитывают тысячи задач, подготовленных для онлайн-тестирования. В качестве примеров можно рассмотреть архив задач Уральского федерального университета (http://acm.timus.ru), сайт сообщества спортивных программистов Саратовского государственного университета (http://codeforces.com), сайт компании TopCoder, специализирующейся на соревнованиях по программированию (http://topcoder.com/tc), учебный комплекс задач по информатике Московского центра непрерывного математического образования (http://informatics.mccme.ru). Стоит отметить, что у всех вышеперечисленных интернет-ресурсов есть две глобальные проблемы. Во-первых, онлайн-взаимодействие требует постоянного интернет-подключения на довольно высокой скорости, что зачастую сложно, особенно в отдаленных регионах

Российской Федерации. Таким образом, необходимость постоянного подключения к сети Интернет сужает круг пользователей данных ресурсов. Во-вторых, все подобные архивы задач, за исключением разве что сайта Дистанционной подготовки Московского центра непрерывного математического образования, хорошо структурированы в областях сложных задач и соревнований по программированию, но обладают слабой и не вполне стройной структурой в области учебных задач для начинающих. Все перечисленные архивы задач ориентированы в первую очередь на соревнования высокого уровня по программированию, например, чемпионат ACM International Collegiate Programming Contest, International Olympiad in Informatics и другие. Таким образом, все перечисленные интернет-ресурсы обладают рядом существенных недостатков, что делает невозможным их использование в рамках создаваемого учебного курса.

2.2 Существующие автономные тестирующие системы

В работе [10] рассматривается множество тестирующих систем, которые могут функционировать автономно при соответствующей настройке. Именно эти системы интегрированы в большинство вышеперечисленных интернет-ресурсов. Описанные в работе системы — eJudge, DOMJudge, dudge, TestSys, PC2, KATTIS, PCMS2 — обладают рядом особенностей. Система KATTIS не была рассмотрена, так как не распространяется свободно, и на ее основе невозможно построить свободно распространяемый курс. Остальные тестирующие системы были опробованы автором данной работы. Практика показала, что мощность и универсальность этих систем отрицательно сказывается на сложности начальной настройки. Зачастую автор был вынужден оставить попытки в полной мере настроить данные системы для использования в разрабатываемом курс. Сложности настройки многих систем были вызваны тем, что у некоторых из них отсутствует по0лная документация. Таким образом, при их использовании от преподавателя требуются нетривиальные навыки администрирования и, зачастую, системного программирования, что делает эти системы пригодными только для узкого круга преподавателей. Самостоятельное же использование таких систем школьниками, в особенности, целевой

аудиторией, изучающей программирование на начальном уровне, становится почти невыполнимой задачей.

Таким образом, становится понятным, что требуется разработать автономную, не требующую сложной настройки и хорошо структурированную тестирующую систему.

2.3 Составляющие тестирующей системы

Для того, чтобы понять, как можно автоматически тестировать задачи, рассмотрим несколько сущностей.

2.3.1 Набор тестов

Набор тестов используется для того, чтобы определить, является ли данная программа полным и корректным решением данной задачи. Программа, которую предоставляет ученик, должна пройти все тесты набора, то есть выдать корректный ответ на них. Набор тестов состоит из одного или нескольких тестов и проверяющей программы.

Тестом называется пара из двух наборов данных. Первый набор — это корректные входные данные для программы ученика. Второй набор — это пример корректных выходных данных, соответствующих входным данным этого набора. Корректные выходные данные либо создаются вручную, либо, предпочтительно, генерируются модельным решением. Ручной подход изредка оказывается предпочтительней, потому что позволяет показать конкретные особенности тестов, особенно если примеры этих тестов приводятся в условии задачи. Однако создание наборов корректных выходных данных автоматически позволяет в большей части случаев избежать ошибок.

2.3.2 Модельное решение

Модельным решением называется программа, написанная разработчиком задачи, решающая данную задачу в условиях, полностью соответствующих условиям ученика. Предполагается, что она полностью корректно работает на всех корректных входных данных. Модельное решение используется для двух целей. Во-первых, с его помощью генерируются корректные выходные данные для всех тестов набора. Во-вторых, эта программа может иллюстрировать

подход автора к решению задачи. Второе, однако, не всегда актуально в случае учебных курсов, потому что подход, ожидаемый от ученика, и подход, решающий задачу быстро и безошибочно, не всегда совпадают. В частности, эти подходы могут не совпадать, когда требуемые для наиболее эффективного подхода знания не входят в программу курса, но задача при этом отрабатывает другие навыки.

Входные данные для тестов могут создаваться двумя способами. Первый из них — ручной: автор задачи подготавливает данные без использования технических средств. Этот подход эффективен при создании небольших входных данных, например, для проверки крайних и вырожденных случаев и минимальных тестов. Второй способ состоит в использовании генераторов.

2.3.3 Генераторы

Генератор — это специальная программа, которая создаёт входные данные для данной задачи. Большинство тестов, размер которых больше десятков байт, эффективнее генерировать программой. Это позволяет чётко описать структуру теста, модифицировать его, масштабировать при необходимости и создавать тесты на базе нескольких уже существующих. По видам генераторы можно классифицировать на несколько типов.

Первый из них — очень широкий: случайные тесты. Такие тесты основаны на генераторах псевдослучайных чисел [5]. В простых случаях эти тесты могут быть последовательностями таких чисел, однако в зависимости от сложности задачи, рассматриваемой структуры данных и существующих ограничений становятся гораздо разнообразнее. К примеру, в задачах, связанных с теорией графов, можно рассматривать случайные деревья или случайные простые пути в заданном графе — генерация таких структур представляет собой отдельную непростую задачу. Впрочем, даже в простых задачах, где входными данными является пара чисел, случайные тесты проверяют многое — они не регулярны, чем находят ошибки во многих решениях, не являющихся корректными.

Второй тип тестов — регулярные тесты. Для задач с последовательностями чисел это могут быть арифметические и геометрические прогрессии, зацикленные паттерны из небольшого количества чисел, функции от номера

числа в последовательности, рекуррентные функции и другие аналогичные вещи. Такие тесты часто находят алгоритмические ошибки в программах, а также часто являются хорошими тестами с точки зрения граничных случаев. Они проверяют переполнения типов данных, превышение максимального времени работы программы и другие ограничения.

К отдельному виду тестов можно отнести наборы данных, полученные совместным использованием разных подходов. Так, регулярный тест с псевдослучайными вкраплениями «шума» может быть, с одной стороны, хорошим тестом на максимальную нагрузку, а с другой не быть разобранным как частный случай: практика показывает, что ученики, находя ошибку в своём решении, чаще стараются разобрать ошибочные случаи отдельными управляющими конструкциями, чем пытаются найти общий подход к решению.

Впрочем, в любой задаче генератор может оказаться не входящим ни в один из перечисленных видов: структура данных каждой задач уникальна и требуется найти способ генерировать тест именно под неё.

2.3.4 Валидаторы

При создании большого количества задач для автоматизации процесса проверки тестов полезно использовать валидаторы. Валидатором называется программа, которая проверяет формальную корректность тестовых данных. Это позволяет как отследить наличие ошибок в формате входных данных, так и в соблюдении существующих ограничений. Концепция валидаторов в значительной мере совпадает с концепцией юнит-тестирования, широко применяющейся в мировой практике разработки программного обеспечения, и служит для аналогичных целей.

2.3.5 Проверяющая программа

Наконец, важной частью набора тестов является проверяющая программа. Проверяющая программа выполняет проверку корректности созданных программой ученика выходных данных, а также соответствие их входным данным. Для этого она, с одной стороны, может использовать уже существующие выходные данные, сгенерированные модельным решением. Это,

однако, не всегда возможно. В случае, когда ответ однозначен — например, в комбинаторной задаче требуется подсчитать количество некоторых объектов, — это наиболее простой и эффективный способ, потому что любая более сложная структура приведёт к дублированию алгоритма модельного решения алгоритмом проверяющей программы. В остальных же случаях — например, в задачах, требующих построить объект, удовлетворяющий некоторым свойствам, — проверяющая программа в обязательном порядке должна проверить соответствие этим свойствам. Отдельный вопрос вызывает случай, когда построить объект не представляется возможным, и корректным ответом является сообщение об этом. Иногда задача позволяет вывести сертификат отсутствия такого объекта, но зачастую это оказывается ненужным усложнением, не требуемым от задачи, особенно на начальном уровне обучения. В таких случаях проверяющей программе эффективнее всего положиться на корректность модельного решения, и факт существования объекта проверять по примеру выходных данных.

2.3.6 Дублирующие и заведомо неверные решения

Помимо модельного решения, большую пользу при создании набора тестов приносят дублирующие решения и заведомо некорректные решения. Дублирующие решения должны быть корректными, то есть должны решать поставленную задачу полностью. Их наличие позволяет валидировать набор тестов и, при необходимости, помогает в поиске ошибок в модельном решении. Заведомо некорректные решения позволяют валидировать полноту набора тестов, то есть убеждаться, что возможные ошибки в решениях выявляются некоторыми из тестов набора.

Хороший набор тестов должен проверять различные аспекты корректности решения. Он должен быть полностью корректным и совместным, то есть все данные должны быть соответствующими условию задачи, модельное и все дублирующие решения должны решать задачу, а все некорректные решения должны выявляться тестами.

Также хороший набор тестов должен быть, по возможности, небольшим. Существуют задачи, где можно включить в тестовые данные все возможные

комбинации корректных входных данных, однако это не является хорошей практикой. Более эффективно строить наборы тестов, каждый из которых выявляет некорректность как можно большего количества решений. В работе [4] рассматривается один из способов подготовки именно такого набора тестов. Генетический подход позволяет эффективно комбинировать тесты различной направленности, получая наборы данных, выявляющие больше ошибок в решениях при небольшом количестве самих данных.

2.4 Интеграция тестов в систему КуМир

Среда КуМир предлагает способ интеграции наборов тестов в задачи, близко связанный с самим языком. Программа ученика скрытым от него образом дополняется внешними частями кода, которые выполняют роли и генераторов тестов, и проверяющих программ. Заметим, что тесты, созданные вручную, также можно интегрировать как отдельные генераторы, а валидаторы реализовать и встроить на этапе, следующем за генерацией. Таким образом, от ученика требуется реализовывать функцию на языке КуМир, которая уже имеет известную сигнатуру, а автор курса реализует генераторы и проверяющие программы. В момент проверки среда интерпретирует генераторы, затем программу ученика, а затем и проверяющую программу.

Это несколько ограничивает автора курса в средствах программирования, однако язык КуМир является, во-первых, Тьюринг-полным, во-вторых, достаточно стандартным для реализации. Также в некоторых случаях используется альтернативный подход. Модельное решение выносится из интегрированной системы. После этого для всех наборов входных данных генерируются выходные данные, которые и интегрируются в систему напрямую. Это позволяет, во-первых, не запускать модельное решение каждый раз, во-вторых, уменьшить время работы тестирования. Когда модельное решение является коротким и достаточно простым, автор использует подход с непосредственным интегрированием модельного решения с систему. Решение ученика и модельное решение одновременно запускаются на одном и том же наборе входных данных, после этого результаты работы этих решений сравниваются. Если ответы совпадают, то считается, что решение ученика

проходит данный тест и тестирование продолжается аналогичным образом на следующих тестов. В том случае, когда авторское решение задачи является достаточно громоздким, правильнее вынести данное за рамки системы КуМир. В том числе, иногда бывает существенно проще написать модельное решение на каком-то другом языке программирования, создать корректные наборы входных и выходных данных и интегрировать в систему непосредственно эти данные. Тем не менее, первый из описанных подходов может быть применён во всех без исключения задачах курса, тем самым гарантируется возможность решения пользователем курса конкретной задачи.

Среди недостатков интегрированной системы тестирования курса в КуМир стоит отметить отсутствие возможности измерять время работы программы ученика и объем использованной ей памяти. Это не позволяет эффективно тестировать решения задач, где важна асимптотическая оценка времени работы — например, ученик может использовать сортировку вставками, имеющую оценку времени работы O(n^2) вместо сортировки слиянием, имеющей оценку времени работы O(n log n). Однако было принято решение в рамках данного курса не вводить подобных проверок, потому что в решениях базовых задач важнее сам факт наличия корректного алгоритма, чем его эффективность. Изучение асимптотических оценок времени работы и объёма используемой памяти является интересной задачей для последующего развития этого курса. Заметим, что интерпретатор КуМир позволяет модифицировать свой код, поэтому добавление в него ограничений по времени и/или по количеству исполненных операций вполне возможно и будет реализовано в дальнейшем.

Результаты по главе 2

1. Проведён сравнительный анализ существующих тестирующих систем, описаны их достоинства и недостатки. Обоснована необходимость создания своей системы.
2. Описан механизм тестирования программ. Описаны методы тестирования задач, создания тестов, интегрирования тестирующего кода в систему КуМир.
3. Описаны возможные дополнения к созданной системе.

Глава 3. Структура курса

Курс разделен на главы, начальные из которых посвящены основным концепциям языков программирования и изучению управляющих конструкций, а последующие — базовому изучению алгоритмов. Главы сильно зависят одна от другой. Переход к следующей имеет смысл только после освоения текущей хотя бы на базовом уровне. В первую очередь это относится к начальным главам, но и в алгоритмической части такая зависимость есть.

Каждая глава содержит от трёх до восьми задач с подготовленными наборами тестов для автоматизированного тестирования. Решение всех задач не является необходимым для перехода к следующей задаче, однако последовательный подход позволяет гораздо лучше усвоить знания из курса.

Далее следует краткое рассмотрение всех главы курса с указанием их особенностей и направленности.

3.1 Арифметические операции

Первая глава посвящена простейшим операциям, которые может выполнять любой язык программирования: арифметические операции. Это необходимая база, без которой невозможно реализовать что бы то ни было более сложное. В данной части содержатся задачи, которые позволяют ученику попробовать применить все арифметические операции, которые возможны в языке Кумир.

3.2 Условный оператор

Условные операторы также являются базовой частью большинства языков программирования. С их появлением арифметика расширяется сравнениями, а также можно вводить понятие логического выражения и логического типа данных.

3.3 Циклы

Циклы — также универсальное средство императивных языков программирования. Данный раздел имеет деление на два подраздела,

посвящённых двум основным видам циклов.

3.3.1 Циклы «Для»

Циклы вида «Для» естественно обобщают понятие повторения выполнения действия фиксированное количество раз. Также эти задачи подготавливают ученика к работе с массивами, как одномерными, так и произвольной размерности.

3.3.2 Циклы «Пока»

Циклы «Пока» вводят альтернативный взгляд на понятие цикла. Теперь действие не просто повторяется фиксированное количество раз, программа может реагировать на различные ситуации. Эти циклы также необходимы для понимания базовых алгоритмов сортировки и поиска.

3.4 Типы данных

Переходя от математического понятия переменной к аналогичному в программировании, необходимо в некоторый момент заметить, что в информатике существуют не только числа. Язык КуМир предоставляет вполне естественный интерфейс работы со строками и с вещественными числами.

3.4.1 Символы и строки

Символы и строки открывают пользователю курса возможность работы с привычными понятиями, получая в качестве входных данных слова русского языка. Ученик может изучить различные методы обработки текста. Тут же вводятся понятия лексикографического сравнения и порядка, необходимые в дальнейших алгоритмических главах. Также рассматривается тема кодировок, соотношения между кодами символов и их представлениями, а также способы преобразования кодов символов согласно этой информации. Используется кодировка ASCII, как стандартная база современных кодировок, в том числе UTF-8.

3.4.2 Вещественные числа

Работа с вещественными числами позволяет решить некоторые прикладные задачи, её можно проиллюстрировать приближенным решением

уравнений и другими арифметическими и математическими задачами. Отдельно стоит сравнение вещественных чисел, само понятие чисел с плавающей точкой и понятие точности. Решение данных задач должно наглядно показать пользователю, что абсолютно точные вычисления невозможны из-за того, каким образом числа хранятся в памяти компьютера. При желании, этому вопросу можно уделить больше внимания в теоретической части курса.

3.5 Одномерные массивы

Начиная с этой главы, ученик уже может сохранять в памяти не только фиксированное количество переменных, но и произвольные их последовательности. Различные способы заполнения массивов, их обходы в различных порядках иллюстрируют циклы «Для» и являются базой для дальнейших алгоритмов.

3.6 Поиск в массиве

Эта глава, наряду со следующей, базируется на основных понятиях из фундаментальной работы «Искусство программирования» (Кнут, том 3). В этой главе изучаются линейный в массиве на примерах поиска максимума, поиска конкретного элемента, поиска элемента с заданными свойствами и разных других. С точки зрения языков программирования это простые и полезные способы применения циклов, в основном, вида «Пока». Данная глава является переходной от изучения языка программирования к изучению алгоритмов.

3.7 Сортировка

Сортировка является классической задачей, на которой изучается построение алгоритмов. Не усложняя задачи асимптотическим анализом и требованием построения оптимального алгоритма, курс позволяет подробно разобраться с понятием инверсии и реализовать сортировки методом выбора максимумов и методом пузырька. Эти задачи отлично иллюстрируют, в частности, вложенные циклы и сложные условия.

3.8 Рекурсия

Концепция рекурсивных вызовов является весьма сложной темой, для понимания которой требуется понимание как вызовов функций как таковых, так

и принципиального отличия функций от математического аналога. В частности, требуется понимать контекст выполнения, отличие локальных переменных, глобальных переменных и аргументов функций. Нужно выработать у ученика понимание стековой структуры вызовов. Задачи этой главы начинаются с несложных и прямолинейных вычислений рекурсивно определённых закономерностей и переходят к гораздо более сложным примерам.

3.9 Игры

Теория игр — интересный раздел математики, задачи которого могут быть зачастую решены с использованием несложных конструкций языка программирования, таких как условные операторы, циклы и массивы. Игры на графах особенно хорошо формализуются в рамках простых задач по информатике. Задачи на анализ дерева игры регулярно встречаются в Едином государственном экзамене по информатике, что хорошо соотносится с целью создания курса.

3.10 Динамическое программирование

Динамическое программирование — важная алгоритмическая тема, рассматриваемая в данном курсе. Во-первых, этот подход решает широкий класс задач. Во-вторых, это классическая тема, являющаяся основой для многих алгоритмов и структур данных. В-третьих, она даёт много примеров использования уже рассмотренных ранее инструментов языка, закрепляя эти темы.

Результаты по главе 3

1. Приведена структура курса с кратким описанием каждой из включённых тем.

Глава 4. Применение курса на практике.

В данном разделе будет описан возможный процесс использования данного курса учителем и учеником.

Полученный в результате работы курс планируется лицензировать по GPL и выложить в открытый доступ в сети интернет. Поэтому получение доступа к

данному курсу учениками и учителями будет простым - им необходимо будет скачать файл с курсом на свой компьютер. Отдельно стоит отметить, что файл данного курса един для учеников и учителей, при этом предоставляет им совершенно разные возможности.

4.1 Использование курса учеником.

Рассмотрим процесс работы ученика с данным курсом. После загрузки файла с курсом, пользователь должен будет открыть этот файл в программе КуМир в специальном разделе практикумов.

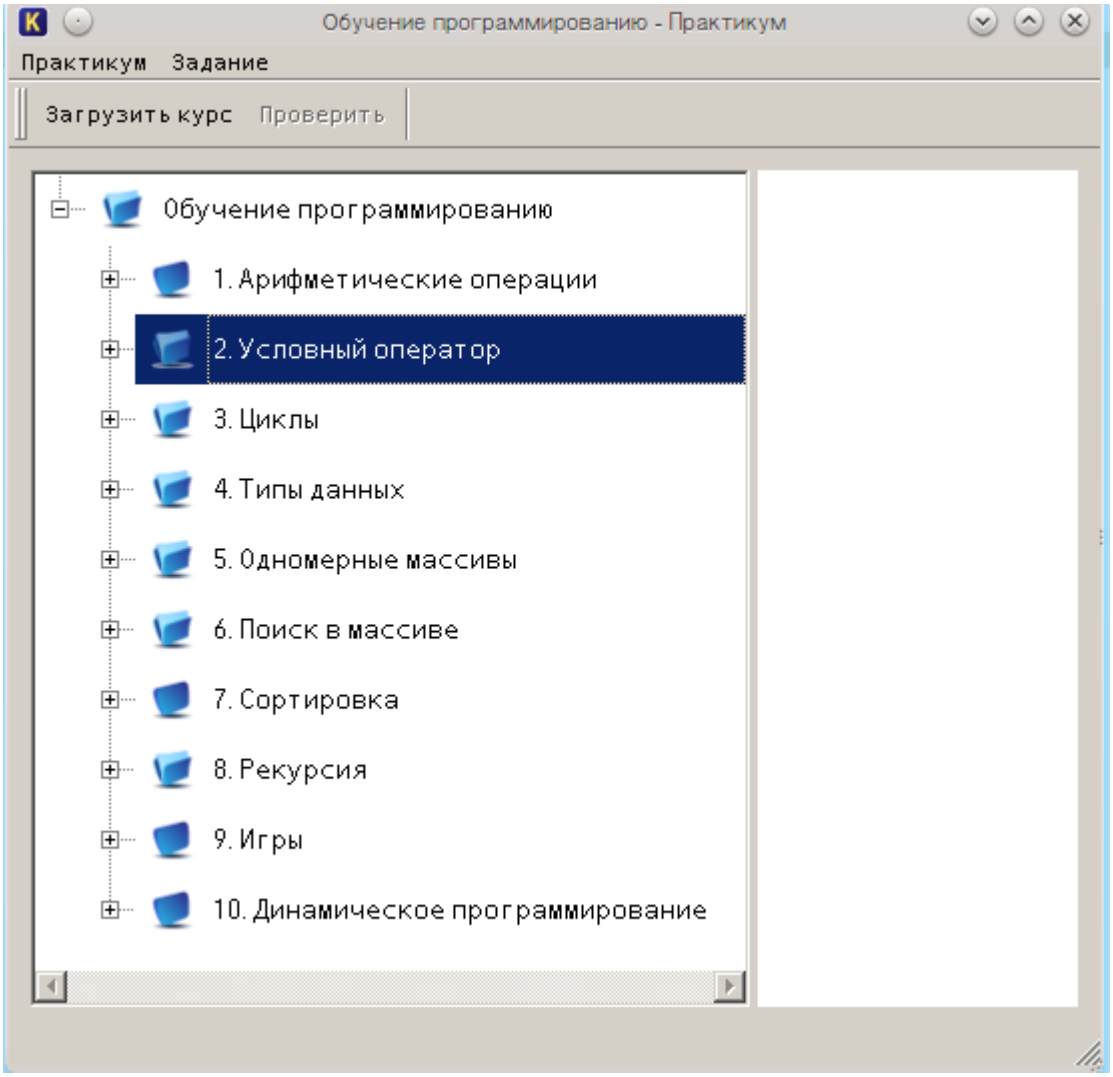

Рис. 4.1 Скриншот программы с загруженным практикумом

Дальше он выбирает конкретную задачу, которую в данный момент собирается решить. Условия всех задач включены в систему курса, что удаляет необходимость использования дополнительных источников информации для получения данных условий.

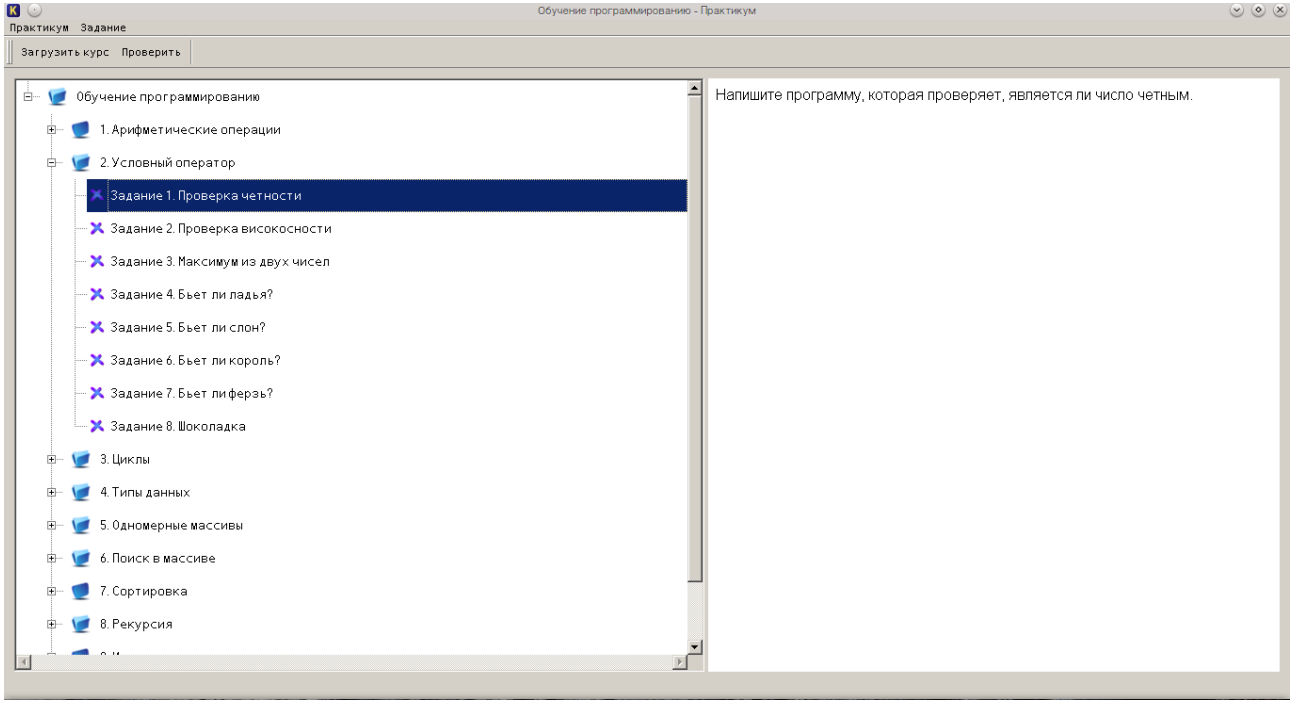

Рис. 4.2 Скриншот программы с процессом выбора задачи

После этого ученик пишет свой код решения задачи в специально обозначенном месте. У ученика уже есть сигнатура функции, которую он не может изменить, а должен написать свой код внутри неё. Он может объявлять любое количество дополнительных переменных и создавать дополнительные функции для решения поставленной задачи.

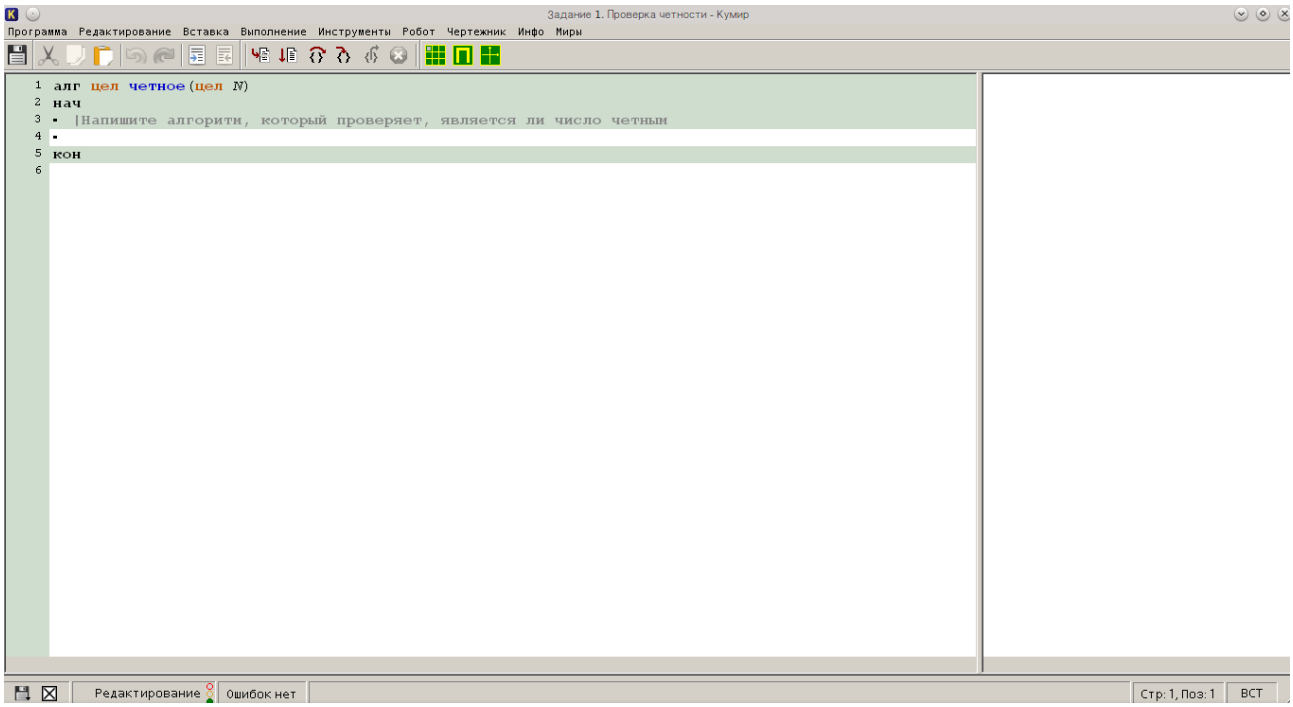

Рис. 4.3 Скриншот программы с процессом написания кода учеником

После того, как код написан и отлажен, программа сохранена, ученик запускает механизм проверки. Написанный им код тестируется на наборе

тестов, о котором говорилось выше. В качестве результатов ученик видит либо сообщение о том, что его программа отработала на всех тестах верно, либо сообщение о том, что некоторые тесты не пройдены и ему стоит подумать над решением задачи ещё некоторое время. Кроме этого, ему в любом случае выставляется некоторое количество баллов. Обычно, полным баллом за задачу является 10 баллов, которые получаются суммированием баллов за каждый пройденный тест. Исключение составляют те задачи, в которых для полной проверки решения необходимо использовать более 10 тестов, что приводит к увеличению суммарного балла, который можно получить за эту задачу. В случае, когда решение работает не на всех тестах, результатом ученика является сумма баллов за те тесты, на которых его решение сработало правильно. Эти баллы могут быть учтены, например, учителем при выставлением оценки за работу.

4.2 Использование курса учителем

Теперь рассмотрим возможности, которые система КуМир предоставляет учителю. Он может запустить программу в специальном учительском режиме с помощью определённой команды.

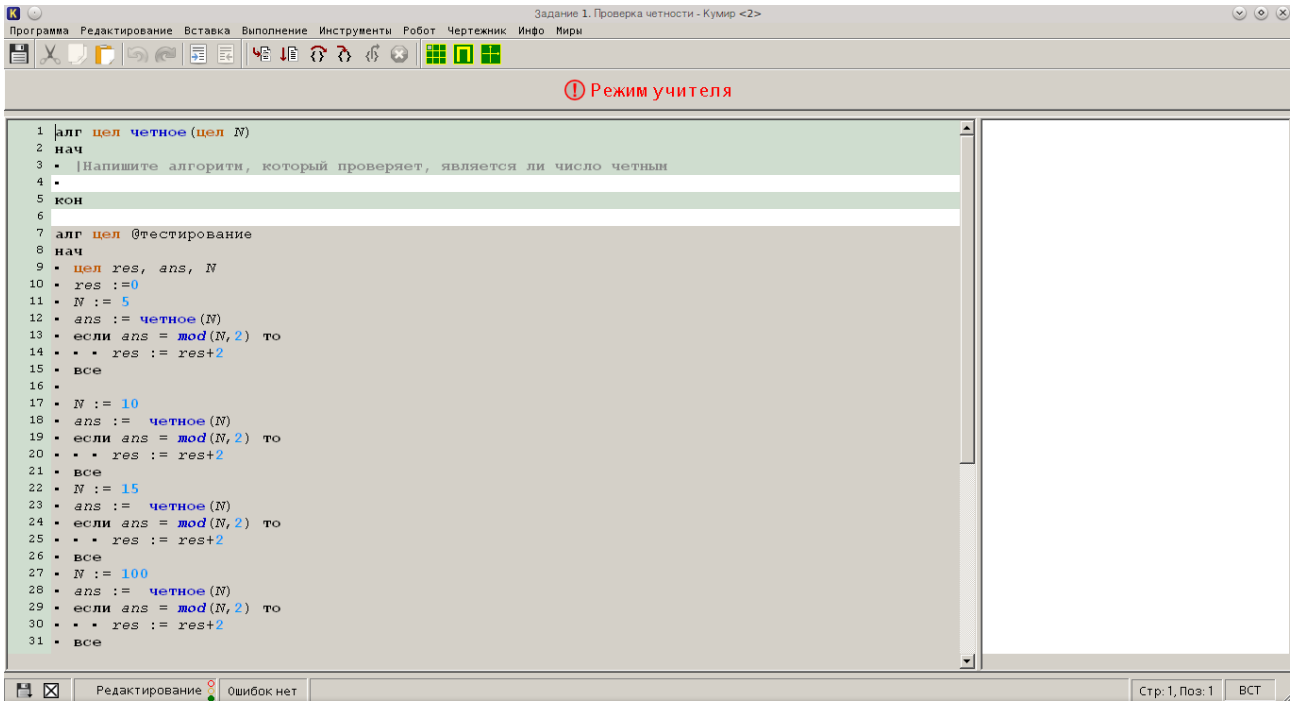

Рис. 4.4 Скриншот программы запущенной в учительском режиме.

После этого ему станут видны все скрытые от ученика части кода, имеющие отношение к процессу тестирования программ. Таким образом,

учитель может по своему усмотрению изменять процесс тестирования, добавляя при необходимости новые тесты, изменяя систему оценивания каждого из тестов. Кроме того, учитель имеет возможность добавлять в курс новые задачи и новые темы, что позволяет подстроить уже существующий курс под надобности конкретного класса или даже конкретного школьника.

Результаты по главе 4

1. Описан процесс использования созданного курса учеником и учителем.

Глава 5. Результаты работы

В результате проделанной автором работы, был создан учебный курс, состоящий из 10 тематических глав и содержащий 46 практических заданий по программированию (Приложение 1). Задачи в данном курсе были подобраны так, чтобы покрыть основные темы начального изучения программирования, дать возможность пользователю на практике закрепить свои теоретические знания в области программирования. К каждой задаче автором курса был подготовлен набор тестов для оценивания полноты и корректности решений пользователей курса.

Несмотря на то, что данный курс покрывает достаточно обширный список тем, в него не были включены некоторые не менее важные темы, например «Двумерные и многомерные массивы», «Геометрические задачи», «Алгоритмы на графах», «Бинарный поиск» и т.д. Тема «Двумерные и многомерные массивы» не была включена в данный курс, поскольку автору не удалось найти интересные задачи по данной теме для включения в курс. Поскольку данная тема является достаточно важной, автор планирует в ближайшее время все-таки решить данную проблему. Темы «Геометрические задачи», «Алгоритмы на графах», «Бинарный поиск» не были включены в данный курс, поскольку они являются достаточно сложными для освоения и не могут быть отнесены к темам для начинающих. Кроме того, многие задачи в данных темах подразумевают оценивание оптимальности решения. К сожалению, на данный момент предложенная автором система тестирования не позволяет оценивать эффективность программ по памяти и времени выполнения. Автор планирует в дальнейшем модифицировать систему тестирования, чтобы функционал оценивания эффективности программ стал доступен. Ещё одна опция. которую на данный момент не поддерживает тестирующая система автора - проверка на правильность не только значения, возвращаемого функцией, но и тех данных, которые могли бы быть выведены на экран. Таким образом, в рамках данного курса не покрывается базовая тема считывания и вывода данных на экран или в файл. Автор считает возможным оставить данную тему без проработки для самостоятельного изучения пользователями.

Что касается практической апробации курса, он был протестирован в ходе индивидуальных занятий автора с учениками 6 и 7 классов. В процессе применения курса на практике, в него были внесены изменения, сделавшие его более простым для понимания пользователем.

Автор планирует в дальнейшем поддерживать и расширять данный курс, добавляя в него новые задачи и увеличивая спектр покрываемых курсом тем для создания на базе данного курса для начинающих полноценного курса для изучения программирования, в том числе планируется вышеописанная доработка системы тестирования и включение в курс сложных тем. Кроме того, автор рассматривает возможность создания теоретической основы данного курса в виде предоставления в том или ином виде базовой необходимой для решения всех задач теоретической информации и полного разбора решения каждой задачи курса.

Данная работа была очень полезна автору в связи с тем, что автор планирует продолжить свою деятельность в педагогической сфере.

Результаты по главе 5

1. Описаны результаты, полученные в ходе выполнения работы.
2. Описаны недоработки, которые планируется исправить в дальнейшем.
3. Описаны планы развития созданного курса.

Заключение

Разработанный курс может быть применён при обучении программированию школьников и всех желающих, не имеющих математической и программистской базы. Программа курса покрывает темы Единого государственного экзамена по информатике, связанные напрямую с программированием, что позволяет использовать полученный курс для подготовки к этому экзамену. Курс как технологическое решение автономен, не требует особой сложной настройки.

Курс предполагает развитие как силами автора, так и силами пользователей: имеется возможность как расширять количество покрываемых тем и усложнять задачи, так и видоизменять инструменты проверки, например, добавляя проверку оптимальности кода по времени выполнения и объему использованной памяти.

Библиография

Печатные издания на русском языке:

1. Андреева Е.В. «Программирование - это так просто, программирование - это так сложно. Современный учебник программирования», М.: МЦНМО, 2009. — 184 с. — ISBN: 978-5-94057-534-4
2. Ахо А., Лам М., Сети Р., Ульман Дж. «Компиляторы: принципы, технолгии и инструментарий» М.: Вильямс, 2008. - 652 с.
3. Ахо А., Хопкрофт Д., Ульман Дж. «Структуры данных и алгоритмы.» М.: Вильямс, 2010. – 400 с.
4. Буздалов М.В., «Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных алгоритмов», выпускная квалификационная работа магистра, Санкт-Петербург, 2011.
5. Кнут Д. «Искусство программирования, том 1. Основные алгоритмы» = «The Art of Computer Programming, vol.1. Fundamental Algorithms.» — 3-е изд. — М.: «Вильямс», 2006. — 720 с. — ISBN 5-8459-0080-8
6. Кнут Д., «Искусство программирования, том 3. Сортировка и поиск» = «The Art of Computer Programming, vol.3. Sorting and Searching.» — 2-е изд. — М.: Вильямс, 2007. — 824 с. — ISBN 5-8459-0082-4
7. Кушниренко А.Г, Леонов А.Г «Методика преподавания основ алгоритмизации на базе системы КуМир» // Информатика. – 2009. – №17-24
8. Поляков К.Ю. КуМир и школьная информатика // Информатика. – 2011. -- №9. – стр. 16-17
9. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. СПб.: Питер, 2002. – 688 с
10. Раскин М. А. «Введение в теорию игр» // Летняя школа «Современная математика». — Дубна, 2008.
11. Романовский И. В. «Дискретный анализ.» СПб.: Невский диалект, 1999. – 240 с.
12. Станкевич А. С., «Методология и технические решения для проведения

олимпиад по информатике и программированию» — Диссертация на соискание учёной степени кандидата наук. Санкт-Петербург, 2011

13. Шень А. «Программирование: теоремы и задачи.» М.: МЦНМО, 2004. – 296 с.

Печатные издания на английском языке:

14. Gárcia-Mateos G., Fernández-Alemán J. L. A course on algorithms and data structures using on-line judging / ITiCSE '09: Proc. 14th ann. SIGCSE conf. on Innovation and technology in Comp. Sci. education, ACM. 2009, pp. 45 – 49.

Ресурсы сети интернет

15. Архив задач Уральского федерального университета http://acm.timus.ru
16. Сайт летней компьютерной школы. http://lksh.ru
17. Сайт московских олимпиад по информатике. http://acm.msu.ru
18. Сайт системы DOMjudge. http://domjudge.sourceforge.net.
19. Сайт проверяющей системы dudge. http://code.google.com/p/dudge.
20. Сайт проверяющей системы KATTIS. http://kattis.csc.kth.se
21. Сайт Санкт-Петербургского тренировочного центра. http://spbtc.ru.
22. Сайт системы проведения соревнований по программированию
23. eJudge. http://ejudge.ru.
24. Сайт системы проведения соревнований по программированию PC2
25. http://www.ecs.csus.edu/pc2/.
26. Сайт соревнований по программированию CodeForces. http://codeforces.ru.
27. Сайт Константина Полякова, посвящённый практикумам в системе КуМир. http://kpolyakov.narod.ru/school/kumir.htm

Приложения

1. CD-диск с файлами всех задач, включённых в курс, и файлом практикума.