Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Работа допущена к защите
зав. кафедрой
_____ В.М. Ицыксон
«___» _____ 2017 г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: Разработка программной системы генерации вариантов учебных задач по дисциплине "Компьютерная алгебра"

Направление: 09.03.01 – Информатика и вычислительная техника

Выполнил студент гр. 43501/1 _____С.М. Карандашов

Научный руководитель,
ст. преподаватель _____ И.А. Малышев

Санкт-Петербург – 2017

РЕФЕРАТ

Отчет, 68 стр., 9 рис., 2 табл., 22 ист., 1 прил.

КОМПЬЮТЕРНАЯ АЛГЕБРА, ВАРИАНТ ЗАДАЧИ, ПРОГРАММНАЯ СИСТЕМА ГЕНЕРАЦИИ, УЧЕБНАЯ ЗАДАЧА, МЕТОД ВОССТАНОВЛЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ШАГОВ РЕШЕНИЯ ЗАДАЧИ, ПАРАМЕТРИЧЕСКАЯ МОДЕЛЬ, ДРОБНО-РАЦИОНАЛЬНОЕ ПОЛИНОМИАЛЬНОЕ ПРЕДСТАВЛЕНИЕ

Разработана программная система генерации вариантов учебных задач по дисциплине «Компьютерная алгебра». Предложена форма спецификации учебной задачи в виде параметрической модели в классе дробно-рациональных полиномиальных представлений. Выбором значений параметров указанной модели обеспечена вариативность учебных задач. Предложен и разработан метод восстановления последовательности шагов решения задачи в обратном порядке (от ответа к условию), повышающий эффективность целевого программного генератора.

ABSTRACT

Report, 68 pages, 9 figures, 2 tables, 22 references, 1 appendicies

COMPUTER ALGEBRA, SOFTWARE GENERATION SYSTEM, LEARNING TASK, METHOD OF RESTORING THE SEQUENCE OF STEPS TO SOLVE A TASK, PARAMETRIC MODEL, VARIANT OF THE TASK, FRACTIONAL-RATIONAL POLINOMIAL REPRESENTATION

Software generation system of the variants of the learning tasks on discipline “Computer algebra” was developed. A form of the specification of the learning task in the form of a parametric model in the class of fractional-rational polynomial representations was proposed. The choice of the values of the parameters of this model ensures the variability of learning tasks. A method for restoring a sequence of steps to solve a problem in reverse order (from an answer to a problem specification), which increases the efficiency of the target software generator, was proposed and developed.

Содержание

ВВЕДЕНИЕ

Учебная дисциплина «Компьютерная алгебра» (КА) относится к образовательному модулю «Модуль профильной направленности» основной образовательной программы подготовки академических бакалавров по направлению «Информатика и вычислительная техника» на кафедре «Компьютерные системы и программные технологии» СПбПУ [1]. В рамках изучения этой дисциплины студентами приобретаются не только знания методов и алгоритмов компьютерной обработки символьной информации (что составляет содержание классических курсов компьютерной алгебры [2, 3, и др.]), но и практические навыки решения целевых задач преобразования представлений различных математических объектов в программных системах компьютерной алгебры (СКА) Maxima [4] и Maple [5]. Поэтому в методическое обеспечение преподавания КА обязательно должен быть включён банк учебных задач (БУЗ). Вне зависимости от выбираемых преподавателем форм контроля навыков решения задач, можно сформулировать следующие требования к задачам, входящим в БУЗ:

1) тематически относиться ко всем изучаемым разделам КА, а также, при необходимости, носить комплексный характер;

2) иметь разную степень сложности как для тренировочных, так и для контрольных задач;

3) содержать достаточное количество вариантов постановок и методов решения.

В настоящее время наблюдается дефицит специализированных сборников задач по КА, удовлетворяющих вышеуказанным требованиям к БУЗ[2]. Поэтому в большинстве случаев преподаватели КА используют в качестве методической основы сборники задач по математике [6,7 и др.], т.е. акцентируют внимание на аналогию форм постановки задач. К сожалению, такой подход к созданию БУЗ не учитывает в должной мере специфику символьных преобразований, свойственных методам решения математических задач в КА. Альтернативный подход, при котором именно методика решения (иными словами, последовательность преобразований символьных представлений математических объектов: от исходных к целевым) служит ключевым признаком типизации задачи, предлагается в настоящей выпускной работе бакалавра.

В бакалаврской работе реализован прототип целевого программного генератора вариантов учебных задач по КА. Он ориентирован на тематический раздел «Дробно-рациональные полиномиальные представления». Разработка программного продукта выполнена на языке Java в системе программирования NetBeans.

1. АНАЛИЗ ПРОБЛЕМЫ СОСТАВЛЕНИЯ УЧЕБНЫХ ЗАДАЧ И СПОСОБОВ ЕЕ РЕШЕНИЯ

1.1. Понятие задачи и методов её решения

Процесс обучения студентов состоит из изучения теоретического материала и практического решения задач. В широком смысле задача - это абстрактный объект, который допускает декларативную и процедурную интерпретации. При декларативной интерпретации под задачей подразумевается целевое требование указать состояние знаний о некоторой предметной области, удовлетворяющих определенным условиям. С другой стороны, процедурно, под задачей обычно понимается метод и/или алгоритм её решения. В учебных курсах под задачей чаще всего подразумевают некое действие или последовательность действий, которые требуется выполнить с помощью умозаключений, вычислений, построений и т.п.[8]

У каждой задачи есть следующие составляющие: условия задачи и ее требования (вопрос, цель). Рассмотрим пример задачи. Условие задачи: человек идет со скоростью 5 километров в час в течение 2 часов. Вопрос задачи: вычислить расстояние, которое пройдет человек.

Все задачи классифицируют по некоторым признакам.

По соотношению между условием и требованием задачи подразделяются на: 1) определенные; 2) неопределенные; 3) переопределенные.

Задачу, в которой данных в условии недостаточно для получения ответа на вопрос, называют неопределенной. Если же в условии задачи данных достаточно, а после отбрасывания одного из условий задача становится неопределенной, то такая задача называется определенной. Если же в условии задачи кроме достаточных данных, содержатся и еще какие-либо данные, то задача называется переопределенной.

По уровню сложности задачи подразделяют на простые и сложные. Если из задачи нельзя выделить какую-либо подзадачу, то задача называется простой, а если можно, то – сложной. Например, задача : «Решите уравнение: 2x=4», будет простой, а задача: «Решите уравнение: 2x+4=8», будет сложной, поскольку в последнем случае можно выделить подзадачу – упрощение выражения (приведение подобных слагаемых).

По отношению к различным разделам математики задачи подразделяют на: арифметические, алгебраические, геометрические, тригонометрические и др.

По дидактическим целям задачи делятся на познавательные, тренировочные и развивающие. Познавательные задачи служат, в основном, для получения новых знаний и особо широко применяются

на начальных этапах обучения для введения новых понятий, знаний. Тренировочные задачи предназначены для выработки прочных навыков и умений в решении задач, примеров, в применении знаний. Развивающие задачи служат для развития творческого мышления.

По типу мышления задачи делятся на алгоритмические, полуалгоритмические и эвристические.

Необходимое условие решения сложной задачи – умение решать простые задачи, к которым сводится любая сложная задача. При наличии такого умения вся проблема состоит в том, чтобы найти ту совокупность простых задач, решение которых приведет к выполнению требования основной задачи. Здесь возможны два конкурирующих пути поиска решения: синтетический и аналитический. Рассмотрим каждый из них.

Синтетический метод решения

Для решения сложных задач обычно выбирается определенная пара данных из условия задачи и соединяется. Если полученная пара образует простую задачу, то эта простая задача решается, и получается промежуточное вспомогательное данное. Если простой задачи не получается, то выбирается другая пара данных. Действия повторяются, пока не получится простая задача, результат которой будет ответом на вопрос исходной задачи.

Рассмотрим этот метод на примере.

Задача. Две машины убирают снег за 6 часов. Однажды, после 3 часов совместной работы, первую машину отправили в другой район города, а оставшаяся машина закончила уборку за 5 часов. За сколько часов каждая машина отдельно может выполнить всю работу?

Р е ш е н и е.

1) Какую часть всей работы выполнили две машины, работая вместе?

3 : 6 = 0,5

2) Какую часть всей работы выполнила вторая машина, после ухода первой?

1 – 0,5 = 0,5

3) За сколько часов могла бы выполнить всю работу вторая машина, работая отдельно?

5 : 0,5 = 10

4) Какую часть всей работы выполнила вторая машина за 3 часа?

3 : 10 = 0,3

5) Какую часть работы выполнила первая машина за 3 часа?

0,5 – 0,3 = 0,2

6) За сколько часов могла бы выполнить всю работу первая машина отдельно?

3 : 0,2 = 15

О т в е т : 15 часов и 10 часов.

Синтетический метод широко применяется не только при решении задач арифметическим способом, но и, например, при решении геометрических задач на вычисление и построение.

Главным недостатком такого метода является отсутствие конкретного критерия по выбору стартовых данных, с которых начинается решение, и какие вспомогательные величины определять для дальнейшего продвижения к результату. Этот метод не пригоден для нахождения новых решений и не способствует обучению логически рассуждать, продуктивно мыслить. Используя этот метод, можно решить большое количество простых задач, без гарантии достижения цели исходной задачи.

Единственное, на что можно в некоторой степени опереться, применяя синтетический метод, – это прошлый опыт в решении задач, аналогия, ассоциации, которые может вызвать решаемая задача.

Достоинством синтетического метода является компактность, достигаемая при изложении готовых решений, полученных в процессе синтетического и/или аналитического поиска.

Несмотря на низкую поисковую и дидактическую эффективность синтетического метода, он пользуется популярностью

у всех участников образовательного процесса, поскольку весьма прост и не требует большого мыслительного напряжения.

Аналитический метод решения

В аналитическом методе решения исходят из требования задачи, а не из начальных данных, составляющих условие задачи.

Решение задач аналитическим методом начинается с постановки следующего вопроса, связанного с требованием решаемой задачи: «Что нужно знать, чтобы ответить на вопрос данной задачи (выполнить требование)?» Для правильного ответа на поставленный вопрос необходимо знать данные задачи и учитывать те зависимости, которые связывают их с искомым ответом.

В практике решения задач методы анализа и синтеза полностью разделить или изолировать друг от друга невозможно. Они полезно сочетаются. При аналитическом методе имеют место скрытые элементы синтеза. Например, преобразуя требование основной задачи в требования первой серии вспомогательных задач, мы неявно проверяем правильность этого преобразования, возможность синтезирования из искомых данных задач первой серии искомого ответа основной задачи.

Хотя путь поиска на основе аналитического метода решения не всегда однозначен, однако он все же менее многозначен и более определен, чем путь поиска при синтетическом методе решения.

Аналитический метод удобен для поиска пути решения новой (ранее не изучавшейся) задачи, он опирается на определенное умение обучающегося рассуждать и эффективно способствует развитию его продуктивного, логического и функционального мышления. В результате систематического применения аналитического метода решения задач у обучаемых быстрее формируется умение самостоятельно решать новые для них задачи, чем при использовании синтетического метода.

В компьютерной алгебре применяются алгебраические аналитические методы решения задач [9].

Под алгебраическим анализом мы будем понимать метод решения задач с помощью уравнений, систем уравнений или неравенств.

Этот метод имеет явно выраженные общие черты аналитического метода. Так, решая задачу с помощью уравнений, отправляются от неизвестного (или неизвестных, если составляют систему уравнений). В качестве неизвестного (неизвестных), как правило, выступает искомое (искомые) данные основной задачи. Но, иногда, для получения более простого решения целесообразно отправляться при составлении уравнения от вспомогательного неизвестного. В последнем случае на первом этапе имеет место преобразование основной задачи в серию из двух вспомогательных

задач, первая из которых решается методом алгебраического анализа, а вторая допускает исключение необходимости алгебраического анализа.

Второй основной этап применения алгебраического анализа – решение полученного уравнения и получение корней. На этом этапе важно следить за равносильностью преобразований, чтобы в итоге получить множество всех решений основной задачи и только его.

На третьем этапе предстоит выяснить достоверность полученных решений, отбросив те, которые не подходят по смыслу задачи. Например, если речь идет о длине отрезка, а в процессе решения появились отрицательные корни, то они, очевидно, не подходят. Иногда достоверность решения проверяется с помощью каких-то известных фактов (равенств, неравенств и т.д.). Если речь в задаче шла о длинах сторон треугольника, то ответ следует проверить на выполнение неравенства треугольника.

1.2. Задачи в компьютерной алгебре

Компьютерная алгебра – это наука об эффективных алгоритмах вычислений математических объектов. Компьютерная алгебра занимается разработкой и реализацией аналитических методов решения математических задач на компьютере и предполагает, что исходные данные, как и результаты решения, сформулированы в аналитическом (символьном) виде.[10]

Компьютерная алгебра рассматривает такие задачи, которые слишком трудно вычислимы для обычной алгебры, но при этом имеют слишком алгебраический характер для информатики.

В настоящее время для решения задач компьютерной алгебры используют специализированные программные системы.

Система компьютерной алгебры— это прикладная программа для символьных вычислений, то есть выполнения преобразований и работы с математическими выражениями в аналитической (символьной) форме.[11]

Символьные вычисления — это преобразования выражений и работа с математическими равенствами и формулами как с последовательностью символов. Они отличаются от численных расчётов, которые оперируют приближёнными численными значениями, стоящими за математическими выражениями. Системы символьных вычислений могут быть использованы для символьного интегрирования и дифференцирования, подстановки одних выражений в другие, упрощения формул и т. д.

При анализе математической модели результатом могут быть общие и частные аналитические решения сформулированной математической задачи и их интерпретации.

Аналитические решения чаще удаётся получить для наиболее простых моделей, реже — для более точных и сложных.

Системы компьютерной алгебры различаются по возможностям, но обычно они поддерживают следующие символьные действия:

- упрощение выражений до меньшего размера или приведение к стандартному виду, включая автоматическое упрощение с использованием предположений и ограничений
- подстановка символьных и численных значений в выражения
- изменение вида выражений: раскрытие произведений и степеней, частичная и полная факторизация (разложение на множители)
- разложение на простые дроби, удовлетворение ограничений, запись тригонометрических функций через экспоненты, преобразование логических выражений
- дифференцирование в частных и полных производных
- нахождение неопределённых и определённых интегралов (символьное интегрирование)
- символьное решение задач оптимизации: нахождение глобальных экстремумов, условных экстремумов и т.д.
- решение линейных и нелинейных уравнений
- алгебраическое (нечисленное) решение дифференциальных и конечно-разностных уравнений
- нахождение пределов функций и последовательностей

- интегральные преобразования
- оперирование с рядами: суммирование, умножение, суперпозиция
- матричные операции: обращение, факторизация, решение спектральных задач
- статистические вычисления
- автоматическое доказательство теорем, формальная верификация
- синтез программ

Для автоматизации построения множества задач существует программные системы. Рассмотрим наиболее часто используемые.

1.3. Обзор существующих программных систем генерации задач

1.3.1 K-Commander

K-Commander [12] – это система для подготовки раздаточного материала контрольных работ, использующая встроенный редактор, написанный на TurboPascal. Эта система была создана в 2002-2005 годах. Код был написан на языке Pascal в среде TurboPascal 7.0. Внешний вид данной программы похож на Norton-оболочку.

Возможности данной системы:

- подготовить в сжатые сроки большое количество вариантов

-получить для каждого варианта набор ответов

-распечатать протокол для оформления результатов контрольной работы

- модифицировать и расширять базу задач

В данной программе нет как таковой генерации – лишь база данных и возможность представления задач в графическом виде.

1.3.2 UniTex

Система UniTex[13] представляет собой программу, написанную на языке среды DELHI-7, которая позволяет:

- создать базу задач, используя обычный встроенный текстовый редактор,

- структурировать подготовленную базу,

- вносить изменения в созданные задачи,

- просмотреть задачи в виде, в котором они будут выводиться на дисплей, принтер, в файл,

- создать базу контрольных работ (КР) на основе подготовленной библиотеки задач,

- структурировать базу КР,

- просмотреть на дисплее варианты любой КР из базы и ответы к ним,

- вывести на принтер необходимое количество вариантов подготовленной КР и ответы к ним,

- записать варианты и ответы выбранной КР в файл в PDF-формате,

- производить настройку программы по следующим направлениям:

1) выбор системы TEX (MikTex, TexLive, EmTex), осуществляющей реализацию языка LaTex на данном компьютере,

2) выбор текстового редактора для набора исходных файлов (WINSHELL, встроенный редактор DELHI),

3) выбор способа вывода готовых вариантов на внешнее устройство (печать DVI- файла, печать PDF-файла при помощи FOXIT-READER, вывод в PDF-файл)

4) установка значений различных числовых величин программы: число вариантов в КР, число вариантов КР (необходимых для различных групп учащихся), обнуление списка групп.

1.3.3 KaTex

К достоинствам данной системы KaTex[14] можно причислить:

- правильная печать математических формул: стиль печати таков, что его практически невозможно подделать при помощи других редакторов

- быстрая печать вариантов контрольной работы при готовой базе задач (весь процесс подготовки контрольной вместе с резанием бумаги занимает 10-15 минут),

- варианты компактно располагаются на бумаге, не разрываясь внутри,

- существует возможность печати ответов

- для каждой группы студентов можно получить свой комплект билетов данной контрольной работы,

- в любое время можно восстановить любой билет любой сформированной контрольной работы,

- система занимает очень мало места на жестком диске, так как программы написаны на языке Turbo-Pascal

1.3.4 TaskGen Plus

Гибкая автоматизированная система генерации заданий[15] для студентов курса «Сопротивление материалов», позволяющая каждому пользователю иметь набор заданий в зависимости от своих приоритетов и специфики преподавания курса для соответствующей специальности. Набор заданий динамичен, то есть допускает возможность модификации без существенной перестройки структуры программ, и пополняемым. Автоматизированная система генерации заданий обладает интуитивно понятным интерфейсом, не требующим ни специальных знаний, ни навыков работы

Генератор обеспечивает высокое визуальное качество заданий, весьма большое количество вариантов, минимальное время на подготовку заданий. Принципиально неограниченное множество вариантов определяется зависимостью генерации от года разработки задания, номера студенческой группы и номера варианта студента. Конечно, количество предлагаемых автоматизированных заданий ограничено, их ровно столько, сколько было запланировано в наборе заданий, однако в силу расширяемости системы генерации заданий к программе можно добавлять дополнительные плагины с заданиями, не перекомпилируя код программы. Требуется только разработать сам плагин на языке C++ в среде Microsoft Visual Studio по уже готовым шаблонам и образцам с кодом и по готовым библиотекам с графическими элементами. Практически вся разработка сводится к компоновке кода из вызовов уже разработанных функций. При загрузке системы генерации заданий она просканирует все существующие плагины, сгруппирует их по темам и добавит в меню программы.

Следует отметить, что автоматизированная расширяемая система генерации заданий имеет один существенный недостаток – для разработки новых заданий необходимо привлекать квалифицированных программистов. Поэтому авторы видят основное направление дальнейшего развития программы в ее перерастании из

системы генерации заданий в систему редактирования и генерации заданий.

1.3.5 Formula Tutor

Система Formula Tutor [16] предназначена для обучения работе с формулами и ориентирована на такие дисциплины, как физика, математика, отчасти химия. Система была разработана М.В. Аксёновым. Поясним работу системы на примере физики. Объектами дедуктивной системы в Formula Tutor являются величины, входящие в физические формулы и законы, а правилами — те формулы и законы, которые эти величины связывают. Такие правила удобно представлять так называемыми продукционными правилами или продукциями вида <антецедент> ^ <консеквент>.

В антецеденте находятся физические величины, позволяющие вычислить величину, стоящую в консеквенте. Функционирование данной дедуктивной системы позволяет формировать сложные формулы различной степени вложенности, что позволяет формулировать задачи на нахождение выражения для некоторой физической величины через другие величины. Так происходит генерация заданий (от искомой величины к величинам, необходимым для ее вычисления, после чего последние получают числовые значения). Анализ ответов в реализованной версии системы

осуществляется путем сравнения с ответами, вычисленными обучающей системой.

Основным недостатком указанной системы является отсутствие или труднодоступность этого генератора для предметной области «алгебра».

1.3.6 Сравнительный анализ программных генераторов учебных задач

Исходные данные для сравнительного анализа приведены в таблице.

	Генерация новых заданий	Возможность увеличения базы данных	Доступность	Простота использования	Простота расширения
K-Commander	-	+	+	+	+
UniTex	-	+	+	+	+
KaTex	-	+	+	+	+
TaskGen Plus	-	+	-	-	-
Formula Tutor	+	+	-	+	-

Результаты сравнительного анализа показывают, что наиболее полно отвечает требованиям задания к данной бакалаврской работе программа Formula Tutor, но её использование в качестве

основы для разработки проблематично. Поэтому принято решение произвести разработку самостоятельно.

1.4. Программные средства представления и обработки символьных выражений

1.4.1. Описание библиотеки jeuclid

Одним из широко распространённых в настоящее время языков представления математических выражений является язык MathML [17]. Для парсинга MathML выражений существует готовый метод MathMLParserSupport библиотеки jeuclid [18]. В данном классе описаны следующие основные функции:

public static DocumentBuilder createDocumentBuilder() – данная функция создает стандартный DOM документ, в который может быть парсирован MathML документ.

public static Document parseInputStreamXML(final InputStream inStream) – данная функция преобразует входной MAtML поток в структуру DOM tree. [19]

Использование данной библиотеки сопряжено с рядом трудностей. Основная проблема – сложность подключения библиотеки к проекту в связи с несоответствием версий среды Java.

1.4.2. Описание библиотеки Jasymca

Для работы с полиномами как с символьными выражениями может быть выбрана система компьютерной алгебры Jasymca.[20] В

ней на языке Java написаны все основные функции работы с полиномами и другими математическими выражениями, поэтому функционально Jasymca может рассматриваться как программная библиотека.

Рассмотрим некоторые полезные (с точки зрения разрабатываемой в настоящей бакалаврской работе программной системы генерации вариантов учебных задач в классе дробно-рациональных полиномиальных представлений) классы программных объектов, входящих в СКА Jasymca.

Класс Algebraic

Абстрактный класс, в котором определены основные алгебраические действия

Переменные:

String name – переменная, к которой привязано

double norm() – норма объекта

Основные функции:

abstract Algebraic add (Algebraic x) – абстрактное определение сложения двух объектов класса algebraic, входного и текущего. На выходе объект класса Algebraic

Algebraic sub (Algebraic x) – вычитание из текущего объекта принимаемого.

Algebraic div (Algebraic x)– делит текущий объект на входной

Algebraic pow_n(int n) – возведения в степень n текущего объекта

abstract Algebraic deriv(Variable var) – дифференциирует объект по переменной

abstract Algebraic integrate(Variable var) – интегрирует объект по переменной

Algebraic reduce() – уменьшает текущей объект до минимально возможного

boolean depends(Variable var) – проверяет зависимость текущего объекта от переменной. Возвращает булеановскую переменную.

boolean equals(Object x) – сравнивает два объекта

Класс Zahl

В этом классе определены некоторые шаблоны вариантов генерации объектов класса Unexakt с необходимыми параметрами. Примеры:

```
public static Zahl HALF     = new Unexakt (0.5);
public static Zahl ONE      = new Unexakt(1.);
public static Zahl TWO      = new Unexakt(2.);
public static Zahl THREE    = new Unexakt(3.);
public static Zahl MINUS    = new Unexakt(-1.);
```

Шаблоны комплексных чисел:

public static Zahl IONE = new Unexakt(0.,1.);

public static Zahl IMINUS = new Unexakt(0.,-1.);

Также в этом классе переопределены некоторые константы:

public static Polynomial PI = new Polynomial(new Constant("pi", Math.PI));

Класс Polynomial

Данный класс содержит в себе все основные возможности по работе с полиномами. В классе хранится:

public Algebraic[] a - массив объектов класса Algebraic, определяющие собой полином.

public Variable var – объект класса Variable, хранящий в себе переменную полинома.

Определенные методы:

public Polynomial(Variable var, Algebraic[] a) – конструктор, принимающий в себя имя переменной и массив элементов.

public Polynomial(Variable var, Vektor v) – массив элементов подается в виде объекта класса вектор.

public Polynomial(Variable var) – сами элементы не подаются и устанавливаются по умолчанию.

Variable getVar() – возвращает имя переменной в виде объекта класса Variable

Vektor coeff() – возвращает коэффициенты полинома в виде вектора

Algebraic coefficient(Variable var, int n) – возвращает коэффициент выбранного порядка в виде объекта класса Algebraic

public int degree() – возвращает степень полинома

Algebraic add(Algebraic p) – сложение двух алгебраических объектов

Algebraic mult(Algebraic p) – произведение двух алгебраических объектов

String toString() - представляет полином в виде строки

К сожалению, практическое использование выше рассмотренных классов и их методов сопряжено с рядом трудностей. Основная проблема совпадает с проблемой использования jeuclid – несовпадений версий среды.

Но, кроме того, у данной библиотеки существует ещё один недостаток. Большинство методов используют статические структуры данных (то есть массив с неизменяемым размером). В связи с этим генерация задания может быть сильно затруднена в связи с ограничением в виде заранее заданного ограничения объёма массива.

1.5. Постановка задачи

Заметим, что качественный генератор задач найти очень трудно. Большинство представленных на рынке продуктов не создают новые задания, они лишь занимаются графическим представлением имеющихся в базе данных задач, либо генерируют тесты с вариантами ответов. Потребительский спрос на подобные генераторы сводится на «нет» из-за все пополняющихся баз данных готовых решений и ответов.

Целью данной бакалаврской работы является написание прототипа генератора вариантов учебных задач по курсу «Компьютерная алгебра», изучаемого студентами кафедры КСПТ СПбПУ. Данный генератор должен отвечать следующим требованиям:

- управляемость. Пользователь должен иметь возможность управлять сложностью получаемых задач и выбирать их тематику.

- вариативность. Программа должна генерировать достаточное (с точки зрения пользователя программы) разнообразие вариантов задач с одним набором исходных данных.

- масштабируемость. Программа должна допускать функциональные и структурные расширения.

В соответствии с Заданием на бакалаврскую работу были поставлены следующие задачи:

1. Обзор существующих методов и средств генерации учебных задач

2. Выбор и разработка подхода к формализации процесса управляемой генерации вариантов учебных задач на основе параметрических моделей

3. Разработка программной реализации системы генерации вариантов учебных задач

4. Проверка работоспособности разработанного программного обеспечения на определенном классе задач компьютерной алгебры

2. РАЗРАБОТКА ПАРАМЕТРИЧЕСКИХ МОДЕЛЕЙ И АЛГОРИТМОВ ГЕНЕРАЦИИ ВАРИАНТОВ УЧЕБНЫХ ЗАДАЧ

2.1. Параметрическая модель

Модель — это система, исследование которой служит средством для получения информации о другой системе. Математическая модель представляют собой совокупность взаимосвязанных математических и формально-логических выражений. Под параметрической моделью понимается математическая модель, позволяющая установить количественную связь между функциональными и вспомогательными параметрами системы[21].

Предложенная в данной бакалаврской работе параметрическая модель предназначена для формального описания учебных задач по дисциплине «Компьютерная алгебра». Она состоит из набора допустимых шаблонов и набора допустимых операций над шаблонами. Шаблоном выражения называется некоторая неизменяемая конструкция. Под операциями над шаблонами подразумевается набор допустимых преобразований одних шаблонов в другие.

Рассмотрим в качестве примера использования параметрической модели учебную задачу, связанную с построением и преобразованием (например, упрощением) полиномов. Общий вид полинома: $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$. Полиномы первой или второй степени – это примеры шаблонов, у которых есть степень полинома, имя переменной и набор коэффициентов. Пример преобразования над шаблонами – превращение полинома второй степени в произведение двух полиномов первой степени.

Математические законы позволяют нам проводить следующие операции с полиномом: разложение на множители, группировка подобных мономов, перемножение полиномов и т.п.

В разрабатываемой программе в качестве входа допустимы следующие типы шаблонов:

- Константа
- Полином первой степени $(ax + b)$
- Отношение полиномов первой степени $\frac{(ax+b)}{(cx+d)}$
- Константа, деленная на полином первой степени
- Полином первой степени, деленный на константу.

Под константой (в качестве типа шаблона) в зависимости от сложности постановки учебной задачи может подразумеваться, в общем случае, целое или рациональное число. Однако в настоящее

время в программном проекте реализована поддержка только целочисленных констант.

Приведем пример сложного выражения, состоящего из шаблонов:

$$\frac{(x^3) * (x^2 + 1)}{4x} * \frac{(x - 5)}{(5x^7 - 20x^6 + 9)}$$

Выражения вида $(ax^2 + c)$, $(ax^7 + bx^6 + c)$ – являются шаблонами. Отношение двух полиномов тоже является шаблоном. В качестве примера операции над шаблонами превратим два шаблона вида (ax^3) и $(bx^2 + c)$ в шаблон $ex^5 + dx^3$.

2.2. Основные подходы к построению алгоритмов генерации

Для наглядного описания существующих и вновь предложенных в настоящей работе подходов к построению алгоритмов генерации учебных задач используем типовую задачу решения (нахождения корней) квадратного уравнения. Проанализируем постановку задачи. Квадратное уравнение имеет следующий вид: $ax^2 + bx + c = 0$. С точки зрения математики квадратное уравнение – это полином второй степени, приравненный к нулю. Нули полинома являются корнями квадратного уравнения.

Поэтому в контексте данной работы можно считать квадратное уравнение и полином второй степени синонимами.

2.2.1 Подход на основе непосредственной генерации задания

Подход основан на непосредственной генерации имен переменных и значений констант, образующих структуру исходных данных задания (условий задачи), и проверки этого задания на корректность. Под корректностью здесь подразумевается разрешимость данного задания в рамках требуемой базы знаний.

В этом подходе в случае с квадратным уравнением генерируются 3 случайных числа: коэффициенты полинома второго порядка a, b и c. Казалось бы, все просто, однако не всякое уравнение имеет решение в области целых чисел. Например, уравнение $x^2 + 1 = 0$ имеет решения только в области комплексных чисел. Из-за этого после каждой попытки генерации требуется проверять результат на допустимость, и отбрасывать неподходящие варианты.

Однако при проверке задания на корректность могут возникнуть большие сложности. Основная проблема заключается в том, что программный генератор «не знает», какой непосредственно путь (порядок действий) необходимо выбрать для получения результата. Кроме того, программа не сможет сама «понять», когда ответ найден, или когда ответа найти невозможно.

Преимущества данного подхода:

- высокая скорость работы генератора
- простота реализации алгоритма генерации

Недостатки:

- требуется проверка допустимости решения задания
-отбрасывается довольно большое количество сгенерированных задач.

2.2.2 Модифицированный подход на основе непосредственной генерации задания

В модифицированном подходе основная идея точно такая же, как и при непосредственной генерации, однако при этом заранее создаются условия для допустимости полученного решения.

Например, в случае с решением задачи факторизации полинома (вынесение множителя за скобки), при генерации коэффициентов производится контроль значения НОД, которое должно быть отличным от единицы. В случае квадратного уравнения требуется получить не 3, а 4 случайных числа (значения коэффициентов), ибо выражение принимает вид $(ax + b)(cx + d) = acx^2 + bcx + +adx + cd.$ В таком случае уравнение будет гарантированно иметь решение, так как его можно будет разложить по теореме Виета[22].

Преимущества данного метода:

-относительно высокая скорость генерации

-простота реализации алгоритма

-гарантированное наличие ответа

Недостатки:

-По завершении генерации необходимо выполнить решение задачи для получения ответа. (так как он в процессе генерации остаётся не известен)

-Отсутствие четких критериев правильности найденного ответа

2.2.3 Подход на основе метода восстановления последовательности шагов решения задачи

Предложенный в настоящей работе подход использует метод пошагового восстановления хода решения задачи, то есть последовательность преобразований задачи от исходных данных к ответу в обратном порядке. Основное преимущество данного метода заключается в том, что мы выбираем один единственный путь генерации.

Например, одним из способов нахождения корней полинома второй степени является теорема Виета, с помощью которой можно разложить полином на множители. Тогда получается, что для получения квадратного уравнения из ответа надо составить два полинома первой степени, перемножить их и сгруппировать значения.

Допустим исходное уравнение имеет вид: $x^2 - 5x - 6 = 0$. По теореме Виета, используя известные свойства корней, выразим значения свободного члена через произведение корней, то есть фактически разложим его на множители -6 и 1. Получим $(x - 6)(x + 1) = 0$

Если идти в обратном порядке, то у нас есть корни уравнения, 6 и -1.

Составим два полинома первой степени: $(x - 6)$ и $(x - (-1))$. Перемножим эти полиномы, и получится $x^2 - 6x + x - 6$. Сгруппировав значения, получим $x^2 - 5x - 6$.

На деле, в данном случае подход через теорему Виета неправилен, поскольку существует ограничение на количество сгенерированных таким образом уравнений. Поэтому для получения корректного решения применим вычисление дискриминанта.

$$x = \frac{-b \pm \sqrt{D}}{2a}$$

$$D = b^2 - 4ac$$

У нас есть два исходных числа, являющиеся корнями данного уравнения. Программа генерирует случайное число a, и происходит составление системы из 2-х уравнений с двумя неизвестными: – b и D.

После вычисления b через случайное число a, и полученные через указанную систему числа b и D, вычисляется число c, и получается исходное уравнение.

2.2.4 Сравнение и выбор метода

		Трудоёмкость нахождения задачи	Трудоёмкость нахождения ответа	Сложность реализации	Общая эффективность
1	Подход на основе непосредственной генерации задания	Крайне низкая	Крайне высокая	Низкая	Очень низкая
2	Модифицированный подход на основе непосредственной генерации задания	Низкая	Высокая	Низкая	Низкая
3	Подход на основе непосредственной генерации задания	Средняя	Заранее известен	Средняя, зависит от темы	Высокая

Для реализации в данном программном проекте в качестве основного был выбран подход 3. Данным методом можно получить гарантированно решаемую задачу, и, так как ответ заранее известен, проверять правильность ответа после генерации не надо.

2.3. Алгоритм генерации сложной дроби для задания на упрощения выражения. Процесс решения и восстановления сложной дроби на примере

Пусть дано следующее алгебраическое выражение:

$$\frac{(2x^4 + 4x^3 - 4x - 2)}{(x^3 + x^2 - x - 1)} * \frac{(x^3 - 7)}{(2x + 2)}$$

Процесс решения заключается в вынесении множителей за скобки, и сокращение, если это возможно. Один из вариантов такого решения представлен в виде последовательности шагов:

Шаг1. $\frac{\cancel{2}(x^4 + 2x^3 - 2x - 1)}{(x^3 + x^2 - x - 1)} * \frac{(x^3 - 7)}{\cancel{2}(x + 1)}$

Шаг2. $\frac{(x^4 + 2x^3 - 2x - 1)}{(x^3 + x^2 - x - 1)} * \frac{(x^3 - 7)}{(x + 1)}$

Шаг 3. $\frac{\cancel{(x+1)}(x^3 + x^2 - x - 1)}{(x^3 + x^2 - x - 1)} * \frac{(x^3 - 7)}{\cancel{(x+1)}}$

Шаг 4. $\frac{\cancel{(x^3 + x^2 - x - 1)}}{\cancel{(x^3 + x^2 - x - 1)}} * \frac{(x^3 - 7)}{1}$

Шаг 5. $\frac{1}{1} * \frac{(x^3 - 7)}{1}$

Ответ: $(x^3 - 7)$

Как получить из этого ответа снова сложную дробь?

Для этого надо пойти в обратном направлении и на каждом шаге умножать числитель и знаменатель на нужный полином.

Исходное данное: $(x^3 - 7)$

Шаг 1. Преобразуем полином в сложную дробь $\frac{1}{1} * \frac{(x^3-7)}{1}$

Шаг 2. Умножаем числитель и знаменатель левой части на полином 3-й степени $(x^3 + x^2 - x - 1)$: $\frac{(x^3+x^2-x-1)}{(x^3+x^2-x-1)} * \frac{(x^3-7)}{1}$

Шаг3. Умножаем числитель левой части и знаменатель правой части выражения на полином первой степени $(x + 1)$: $\frac{(x^4+2x^3-2x-1)}{(x^3+x^2-x-1)} * \frac{(x^3-7)}{(x+1)}$

Шаг4. Умножаем числитель левой части и знаменатель правой части выражения на константу 2:

$$\frac{(2x^4 + 4x^3 - 4x - 2)}{(x^3 + x^2 - x - 1)} * \frac{(x^3 - 7)}{(2x + 2)}$$

3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ГЕНЕРАТОРА ВАРИАНТОВ УЧЕБНЫХ ЗАДАЧ

3.1.Выбор среды и архитектуры программирования

С целью демонстрации работы алгоритма генерации заданий было разработано приложение-прототип.

Алгоритм программы:

1) На вход программы подаются необходимые данные:

- Требуемый ответ, который должен будет быть получен учащимся в результате решения сгенерированного задания
- Уровень сложности задания

2) Генерируется задание согласно разработанному алгоритму

3) Выводится в консоль результат генерации

Для реализации был выбран язык Java и среда NetBeans.

3.2.Алгоритм получения сложной дроби из ответа

Для реализации метода пошагового восстановления хода решения задачи был разработан следующий алгоритм:

1) Распознается шаблон выражения.

2) Производятся начальные преобразования над исходным выражением – создается 4 вектора, 2 для знаменателя и 2 для числителя.

3) Генерируется случайный полином требуемой степени. Степень варьируется случайным образом от 0 до 2-х.

4) Одно из числителей и один из знаменателей перемножается на сгенерированный полином.

5) Счетчик уровня сложности уменьшается на один

6) Повторяются шаги 3-5, пока счетчик не станет равным 0.

7) Формируется окончательный результат.

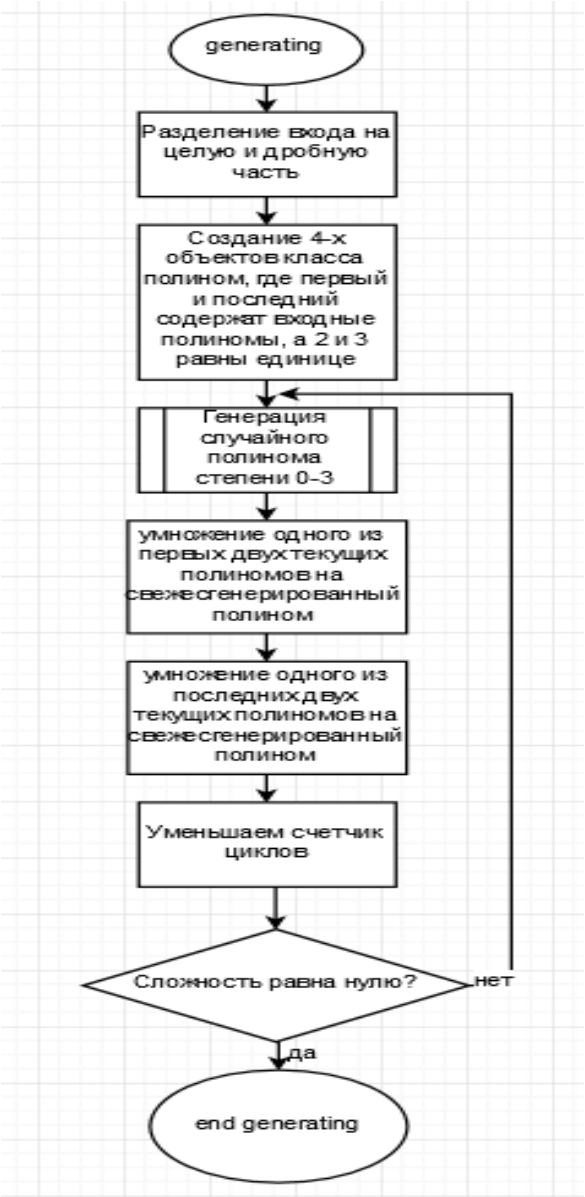

Рис. 1. Схема алгоритма генерации сложной дроби

3.3. Модуль генерации вариантов

3.3.1. Главный класс, реализующий основную обработку программы

В качестве основного связующего звена модулей программы был разработан главный класс main (листинг А.1 приложения А). В данном классе производится считывание шаблона и уровня сложности генерируемого задания. Затем производится вызов генератора и вызов функции печати сгенерированного задания.

3.3.2. Класс, содержащий основные возможности по работе с полиномами.

Для хранения данных о полиноме и основных функций по работе с ним был разработан класс Polynom (листинг А.2 приложения А). В данном классе хранится:

Vector<Integer> poly – динамическая структура данных для хранения целых коэффициентов полинома. В ячейке с нулевым индексом хранится свободный член полинома, в последующих ячейках хранятся коэффициенты мономов начиная с первой степени.

Integer exp; - степень полинома

Основные методы данного класса:

public Polynom mult(Polynom pol) - в этой функции реализовано умножение двух полиномов. Функция принимает

умножаемый полином и умножает его на текущий. Возвращает объект класса полином.

public void read(int n) - в данной функции реализовано считывание коэффициентов полинома из консоли. На вход принимается степень считываемого полинома.

public String toString() – функция перевода массива коэффициентов в строковый вид.

3.3.3. Класс генерации дробно-рационального полиномиального представления

В классе arrPoly (листинг А.3 приложения А) реализована генерация требуемого дробно-рационального полиномиального представления, а так же возможность вывода его в консоль.

Хранимые переменные:

private static final int RANDOM_POLYNOMIAL_MAX_SIZE = 3 – в этой константе хранится максимальный размер автоматически генерируемых полиномов

private static final int RANDOM_COEFFICIENT_MAX_SIZE = 10 – хранится максимальный размер коэффициентов генерируемых полиномов.

Polynom chisl1, Polynom chisl2 – хранятся числители генерируемого дробно-рационального полиномиального

представления вида «сложная дробь». В Polynom znam1 и Polynom znam2 хранятся знаменатели.

Реализованные основные функции:

public void read(int n) – считывает заданный ответ из консоли и распознает шаблон считанного ответа

public void Podgotov(Polynom pol1, Polynom pol2) – преобразует считанный шаблон в необходимый для дальнейшей работы вид.

public Polynom generate() – генерирует полином случайной степени со случайными коэффициентами. В рамках данной программы коэффициенты ограничены целыми числами, а максимальная степень генерируемого полинома равняется двум.

public void Generator(int sl) – функция генерации исходного задания. Реализован генератор по алгоритму, описанному в пункте 3.2. данной работы.

public void print() – функция печати в консоль дробно-рационального полиномиального представления.

private String addSpace(String s, int maxs) – вспомогательная функция для заполнения пробелами строки до требуемой длины.

4. ПРАКТИЧЕСКАЯ АПРОБАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ ГЕНЕРАЦИИ ВАРИАНТОВ УЧЕБНЫХ ЗАДАЧ

4.1.Демонстрация работы программы

Для проверки работоспособности программы были протестированы ее следующие функциональные возможности:

1) Изменение вида шаблона выражения задаваемого ответа
2) Изменение уровня сложности генерируемого задания

Приведены примеры исходных данных и сгенерированных на основе этих данных заданий:

```
Введите количество входных элементов
1
Введите степень полинома (0, если константа или 1, если полином первой степени)
0
Введите константу
11

Введенный ответ
(11)   (1)
____ X ___
(1)    (1)
Введите уровень сложности
1

                                Сгенерированное задание
                                (99x^2 + 110x + 44)    (1)
                                ___________________ X ___
                                (9x^2 + 10x + 4)       (1)
```

Рис.2. Пример сгенерированного задания уровня сложности 1 из шаблона вида константа.

```
Введите количество входных элементов
1
Введите степень полинома (0, если константа или 1, если полином первой степени)
1
Введите коэффициенты полинома
4 7

Введенный ответ
(1)    (4x + 7)
___ X _________
(1)    (1)
Введите уровень сложности
3

                                    Сгенерированное задание
                                    (12x + 18)    (4x + 7)
                                    __________ X _________
                                    (6x + 9)      (2)
```

Рис.3. Пример сгенерированного задания уровня сложности 3 из шаблона вида полином первой степени

```
Введите количество входных элементов
2
Введите степень полинома в числителе(0, если константа или 1, если полином первой степени)
0
Введите константу
6
Введите степень полинома в знаменателе (0, если константа или 1, если полином первой степени)
1
Введите коэффициенты полинома
1 7

Введенный ответ
(1)         (6)
_________ X ___
(1x + 7)    (1)
Введите уровень сложности
5

Сгенерированное задание
(56x^4 + 107x^3 + 116x^2 + 86x + 9)                 (480x^3 + 1080x^2 + 1680x + 1080)
___________________________________________ X ____________________________________
(32x^5 + 284x^4 + 553x^3 + 1021x^2 + 711x + 567)    (140x^3 + 320x^2 + 200x + 20)
```

Рис.4. Пример сгенерированного задания уровня сложности 5 из шаблона вида «константа, деленная на полином первой степени»

```
Введите количество входных элементов
2
Введите степень полинома в числителе(0, если константа или 1, если полином первой степени)
1
Введите коэффициенты полинома
1 5
Введите степень полинома в знаменателе (0, если константа или 1, если полином первой степени)
2
Введите коэффициенты полинома
3 8

Введенный ответ
(1x + 5)    (1)
________ X ___
(3x + 8)    (1)
Введите уровень сложности
4
```

```
Сгенерированное задание
(1x + 5)                (800x + 400)
_____________________ X ____________
(48x^2 + 152x + 64)     (50)
```

Рис.5. Пример сгенерированного задания сложности 4 из шаблона вида «отношение полиномов первой степени».

4.2.Проверка корректности сгенерированных заданий

Проверка решаемости задания на примере продемонстрированного на рис.5. сгенерированного задания.

```
Сгенерированное задание
(1x + 5)                (800x + 400)
_____________________ X ____________
(48x^2 + 152x + 64)     (50)
```

Шаг 1. Разделим правую часть числителя и левую часть знаменателя на константу (полином нулевой степени) 8:

```
(1x + 5)             (100x + 50)
________________ X ___________
(6x^2 + 19x + 8)     (50)
```

Шаг 2. Разделим правую часть числителя и левую часть знаменателя на полином первой степени (2x+1):

```
(1x + 5)    (50)
________ X ____
(3x + 8)    (50)
```

Шаг 3. Разделим правую часть числителя и знаменателя на константу 5:

```
(1x + 5)    (10)
________ X ____
(3x + 8)    (10)
```

Шаг 4. Разделим правую часть числителя и знаменателя на константу 10:

```
Введенный ответ
(1x + 5)    (1)
________ X ___
(3x + 8)    (1)
```

Была продемонстрирована корректность (решаемость) сгенерированного задания, а так же правильность уровня сложности.

4.3.Анализ влияния уровня сложности задания на скорость работы программы.

Проверим скорость генерации задания уровней сложности 0,1,10 и 100 из шаблона типа «константа». В приведенных ниже результатах есть погрешность в виде человеческого фактора.

```
Сгенерированное задание
(1)    (1)
___ X ___
(1)    (1)
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 3 секунды)
```

Рис. 6. Время генерации задания уровня сложности 0.

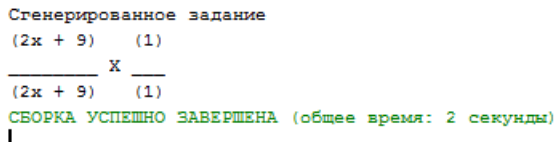

Рис.7. Время генерации задания уровня сложности 1.

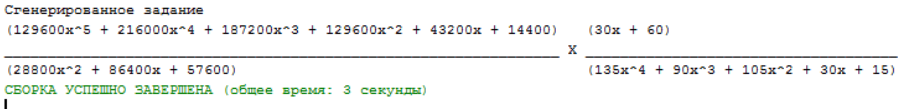

Рис.8. Время генерации задания уровня сложности 10.

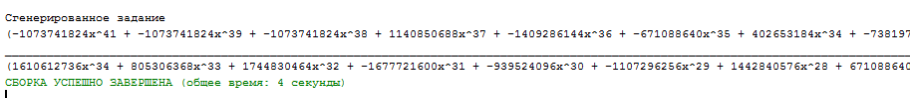

Рис.9. Время генерации задания уровня сложности 100.

Из представленных результатов видно, что при значительном изменении уровня сложности генерируемого задания время генерации практически не изменяется

ЗАКЛЮЧЕНИЕ

Содержательно настоящая выпускная работа является поисковой, поэтому разработанный в ней программный генератор вариантов учебных задач по дисциплине «Компьютерная алгебра» не претендует ни на функциональную полноту, ни на потребительскую ценность. Вместе с тем, в ходе проектирования были достигнуты следующие результаты:

1) предложена и разработана параметрическая модель учебной задачи по КА;

2) разработан алгоритм генерации вариантов учебных задач по тематическому разделу КА «Дробно-рациональные полиномиальные представления»;

3) выполнена программная реализация вышеуказанных модели и алгоритма;

4) проверена работоспособность программной системы для выбранного класса задач КА.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет-страница учебного курса «Компьютерная алгебра» КСПТ СПБПУ Петра Великого. - http://kspt.icc.spbstu.ru/course/comp-algebra
2. Интернет-страница кафедры алгоритмических языков ВМК МГУ. - http://al.cs.msu.ru/seminars/catfl
3. Интернет страница курса «Компьютерная алгебра» Нижегородского государственного университету им.Н.И.Лобачевского. - http://www.itlab.unn.ru/?dir=201
4. Maxima, a Computer Algebra System. - http://maxima.sourceforge.net/
5. Maplesoft products. - http://www.maplesoft.com/
6. Рывкин А. Сборник задач по математике с решениями для поступающих в вузы. – М., ОНИКС 21 век, 2003.
7. Петерсон Л.Г., Абраров Д.Л., Чуткова Е.В. Математика. 7 класс. Учебник в 3 ч.. – М., Ювента, 2011.
8. Интернет-страница сайта Студопедия. Обучение решению задач. - http://studopedia.ru/2_32232_obuchenie-resheniyu-zadach.html
9. Акритас А. Системы компьютерной алгебры с приложениями. – М., Мир, 1994.

10. Дьяконов В. П. Энциклопедия компьютерной алгебры. — 1-е изд., в двух томах. — М., ДМК-Пресс, 2009. — 1264 с.
11. Система компьютерной алгебры. Страница сайта Wikipedia - https://ru.wikipedia.org/wiki/Система_компьютерной_алгебры
12. Карнаухов В.М. Использование компьютерного генератора контрольных работ в преподавании высшей математики.
13. Описание системы UniTex. - http://matmsuee.narod.ru/nauk/monographi/231.pdf
14. Страница проекта KaTex на сайте github. - https://github.com/Khan/KaTeX
15. Бакушев С.В. Куликов А.В. Расширяемая система генерации заданий. // Текст научной статьи по специальности «Автоматика. Вычислительная техника»
16. Братчиков И.А. Генерация тестовых заданий в экспертно-обучающих системах. // Текст научной статьи по специальности «Народное образование. Педагогика».
17. Страница MathML на сайте W3C - https://www.w3.org/Math/
18. JEuclid MathML rendering solution. - http://jeuclid.sourceforge.net/
19. Интернет-страница википедии с описанием интерфейса DOM - https://ru.wikipedia.org/wiki/Document_Object_Model
20. Dersch H. Jasymca - Java Symbolic Calculator. - https://webuser.hs-furtwangen.de/~dersch/jasymca2/indexEN.html

21. Уёмов А.И. Логические основы метода моделирования. - М., Мысль, 1971. — 311 с, с.48
22. Винберг Э.Б. Алгебра многочленов. Учебное пособие для студентов-заочников III—IV курсов физико-математических факультетов педагогических институтов – М., Просвещение,1980. – с.26

ПРИЛОЖЕНИЕ А

ТЕКСТ ПРОГРАММЫ

Листинг А.1. Исходный код класса main

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package baka_low;

import java.util.Scanner;
import java.util.Vector;

/**
 *
 * @author Verwulf
 */
public class main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

```
        Scanner in = new Scanner(System.in);
        System.out.println("Введите количество входных элементов");
        int kolv = in.nextInt();
        arrPoly uravnenie = new arrPoly();
        uravnenie.read(kolv);

        System.out.println("");
        System.out.println("Введенный ответ");
        uravnenie.print();

        System.out.println("Введите уровень сложности");
        int sl = in.nextInt();
        uravnenie.Generator(sl);
        System.out.println("");
        System.out.println("Сгенерированное задание");
        uravnenie.print();

    }

}
```

Листинг А.2. Исходный код класса Polynom

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this templte file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package baka_low;

import java.util.Random;
import java.util.Scanner;
import java.util.Vector;

/**
 *
 * @author Verwulf
 */
public class Polynom {



    Vector<Integer> poly = new Vector<Integer>(); //коэффициенты
    Integer exp; //степень

    public Polynom() {
    }

    public Polynom(Vector<Integer> pol) {
        this.poly = pol;
        this.exp = poly.size();
    }

    public Polynom mult(Polynom pol){
```

```
            Polynom vspPol = new Polynom();
            int n = poly.size()+pol.poly.size()-1;
            for (int i=0; i<n; i++)
            {
               vspPol.poly.addElement(0);
            }
            for (int i=0; i<pol.poly.size(); i++)
            {
               for(int j=0; j<poly.size(); j++)
               {
                  int vsp = vspPol.poly.elementAt(i+j) +
pol.poly.elementAt(i)*poly.elementAt(j);
                  vspPol.poly.set(i+j, vsp);
               }
            }
            return vspPol;
         }



         public void read(int n)
         {
            Polynom pol = new Polynom();
            pol.exp=n-1;
            Scanner in = new Scanner(System.in);
            System.out.println("Введите коэффициенты полинома");
```

```
            for(int i=0; i<n; i++)
                pol.poly.addElement(in.nextInt());
            for(int i=0; i<pol.poly.size(); i++)
                poly.addElement(pol.poly.elementAt(pol.poly.size()-i-1));
        }

        public String toString()
        {
            String s = "";
            s = "(";
            if(poly.size()==1)
            {
                s=s+poly.elementAt(0).toString();
            }
            else if(poly.size()==2)
            {
                s=s + poly.elementAt(1).toString() + "x + " +
poly.elementAt(0).toString();
            }
            else
            {
                for(int i=this.poly.size()-1; i>1; i--)
                {
                    if(poly.elementAt(i)!=0)
                        s=s+poly.elementAt(i).toString()+"x^"+String.valueOf(i)+" + ";
                }
```

```
                s=s + poly.elementAt(1).toString() + "x + " +
poly.elementAt(0).toString();
                }
                s=s+")";
                return s;
            }
        }
```

Листинг А.3. Исходный код класса arrPoly

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package baka_low;

import java.util.Random;
import java.util.Scanner;

/**
 *
 * @author Verwulf
 */
public class arrPoly extends Polynom{

    private static final int RANDOM_POLYNOMIAL_MAX_SIZE = 3;
```

```
private static final int RANDOM_COEFFICIENT_MAX_SIZE = 10;

Polynom chisl1 = new Polynom();
Polynom chisl2 = new Polynom();
Polynom znam1 = new Polynom();
Polynom znam2 = new Polynom();
Random random = new Random();

public arrPoly(){
}

public void Podgotov(Polynom pol1, Polynom pol2)
{
   int i = random.nextInt(2);
   switch(i)
   {
      case 0: chisl1 = pol1;
            chisl2.poly.addElement(1);
         break;
      case 1: chisl2 = pol1;
            chisl1.poly.addElement(1);
         break;
   }
   i = random.nextInt(2);
   switch(i)
   {
      case 0: znam1 = pol2;
```

```
                    znam2.poly.addElement(1);
                    break;
                case 1: znam2 = pol2;
                    znam1.poly.addElement(1);
                    break;
            }
        }

        public void read(int n)
        {
            Polynom vh1 = new Polynom();
            Polynom vh2 = new Polynom();
            Scanner in = new Scanner(System.in);
            if (n==1)
            {

                System.out.println("Введите степень полинома (0, если константа или 1, если полином первой степени)");
                int st = in.nextInt();
                if (st==0)
                {
                    System.out.println("Введите константу");
                    vh1.poly.addElement(in.nextInt());
                    vh2.poly.addElement(1);
                    Podgotov(vh1, vh2);
                }
                else
```

```
                {
                    vh1.read(2);
                    vh2.poly.addElement(1);
                    Podgotov(vh1, vh2);
                }
            }
            else
            {
                System.out.println("Введите степень полинома в числителе(0, если константа или 1, если полином первой степени)");
                int st = in.nextInt();
                if (st==0)
                {
                    System.out.println("Введите константу");
                    vh1.poly.addElement(in.nextInt());
                }
                else
                {
                    vh1.read(2);
                }
                System.out.println("Введите степень полинома в знаменателе (0, если константа или 1, если полином первой степени)");
                st = in.nextInt();
                if (st==0)
                {
                    System.out.println("Введите константу");
                    vh2.poly.addElement(in.nextInt());
```

```
                    }
                    else
                    {
                        vh2.read(2);
                    }
                    Podgotov(vh1, vh2);
                }
            }

            public Polynom generate(){
                Polynom pol = new Polynom();
                Random random = new Random();
                pol.exp = random.nextInt(RANDOM_POLYNOMIAL_MAX_SIZE)+1;
                for(int i=0; i<pol.exp; i++)
                {

pol.poly.addElement(random.nextInt(RANDOM_COEFFICIENT_MAX_SIZE)+1);
                }
                return pol;
            }

            public void Generator(int sl)
            {
                for(int i=0; i<sl; i++)
                {
                    Polynom pol = new Polynom();
                    pol = generate();
```

```
            System.out.println("----------");
            System.out.println(pol.toString());
            System.out.println("");
            int j = random.nextInt(2);
            switch(j)
            {
               case 0: chisl1 = chisl1.mult(pol);
                  break;
               case 1: chisl2 = chisl2.mult(pol);
                  break;
            }
            j = random.nextInt(2);
            switch(j)
            {
               case 0: znam1 = znam1.mult(pol);
                  break;
               case 1: znam2= znam2.mult(pol);
                  break;
            }
            print();
            System.out.println("---------");
         }
      }

      public void print()
      {
         String chisl = "";
```

```
                String znam = "";
                String znak = "";
                int maxs = Integer.max(chisl1.toString().length(),
znam1.toString().length());
                chisl = addSpace(chisl1.toString(), maxs+3);
                znam = addSpace(znam1.toString(), maxs+3);
                for (int i=0; i< maxs; i++)
                   znak=znak+"_";
                znak=znak+" X ";
                maxs = Integer.max(chisl2.toString().length(), znam2.toString().length());
                chisl = chisl + addSpace(chisl2.toString(), maxs);
                znam = znam + addSpace(znam2.toString(), maxs);
                for (int i=0; i< maxs; i++)
                   znak=znak+"_";
                System.out.println(chisl);
                System.out.println(znak);
                System.out.println(znam);
             }

             private String addSpace(String s, int maxs)
             {
                for (int i=s.length(); i<maxs; i++)
                   s=s+" ";
                return s;
             }
          }
```