Оглавление

[Аннотация 2](#_Toc12001245)

[Введение 3](#_Toc12001246)

[Постановка задачи 4](#_Toc12001247)

[Глава 1. Особенности автоматической генерации задач 5](#_Toc12001248)

[Тестирование в учебных заведениях 5](#_Toc12001249)

[Автоматическая генерация задач 6](#_Toc12001250)

[Простой способ генерации общего уравнения прямой 7](#_Toc12001251)

[Нахождение НОД. 8](#_Toc12001252)

[Другие примеры генерации задач 9](#_Toc12001253)

[Генерация ответов к задачам 11](#_Toc12001254)

[Глава 2. Реализация системы тестирования 12](#_Toc12001255)

[Краткое описание внутреннего устройства: 12](#_Toc12001256)

[Интерфейс 13](#_Toc12001257)

[Использованные функции геометрических типов данных и операций над ними. 14](#_Toc12001258)

[Примеры решения разных типов задач и алгоритмы генерации задач программой 15](#_Toc12001259)

[Вывод результатов 23](#_Toc12001260)

[Заключение 24](#_Toc12001261)

[Список используемой литературы 25](#_Toc12001262)

Аннотация

Данная работа направлена на разработку открытого интерактивного тестирования по теме «Прямые на плоскости».

В данной работе представлена теоретическая часть, посвященная поиску методов генерации заданий и методов проверки введенных ответов.

Отличительными особенностями этого тестирования является, во-первых, его открытость. То есть ответ не выбирается из списка, а вводится с клавиатуры. А во-вторых, разработаны алгоритмы генерации исходных данных.

Введение

Большинство тестирований построены по схеме выбора одного или нескольких ответов из предложенного списка. Такой подход имеет ряд недостатков. Например, трудоемкий процесс подбора заданий, ответов, причем составителю теста приходится подбирать как правильный, так и неправильные ответы. Такжесуществует возможность угадывания верного ответа, причем процент , что это случится очень велик.

Для решения таких проблем, как угадывание целесообразно проводить открытые тестирования, где ученику предлагается вводить ответ, а не выбирать его.

Для проведения подобного тестирования проверяющему знания студентов необходимо подобрать разнообразные задачи для каждого варианта. Основная сложность этого подбора состоит в том, что необходимым условием является одинаковый уровень сложности для каждого варианта.

Актуальным решением проблемы является автоматизированное порождение тестирований для проверки знаний по выбранной теме. Однако, если генерировать задания случайно, то при решении таких заданий ученик будет сталкиваться с трудностями, например с иррациональностью подсчета. Отсюда можно сделать вывод: нужен направленный перебор исходных значений. Формирование теста для каждого варианта происходит по следующей схеме: даны основные типы задач (набор задач одинаков для каждого варианта), а такие численные параметры, как, например коэффициенты в исходных уравнениях порождаются самой программой по индивидуальной схеме для каждой задачи.

В основной части дипломной работы подобранны некоторые типы задач по теме «Прямые на плоскости» и алгоритмы их решения тестируемым. А также алгоритмы подбора исходных коэффициентов для получения так называемых «красивыхответов».

Постановка задачи

Для разработки автоматизированного интерактивного компьютерного тестирования необходимо решить ряд первостепенных задач. Как и любой программе, тестированию нужен понятный интерфейс, который позволит ученику затрачивать внимание в основном на решение задач, но в свою очередь даст возможность преподавателю устанавливать некоторые настройки.

Необходимо разработать методы генерации исходных значений, которые позволят при решении заданий не испытывать трудностей с вычислениями.

А также разработка методов проверки правильности введенного учеником ответа.

Глава 1. Особенности автоматической генерации задач

### ****Тестирование в учебных заведениях****

Преподаватель в школе или университете должен проверять, какие знания присутствуют у его учеников. Для этого хорошо подходят контрольные задания. Ученикам выдается набор задач, которые они должны решить самостоятельно, но обеспечить это оказывается не очень просто. Этому мешает ряд проблем:

* Варианты должны быть различные у каждого ученика. Составление каждого варианта задачи ложится на преподавателя;
* Если варианты все таки повторяются, необходимо исключить возможность списывания. Обеспечить это часто бывает затруднительно;

В рамках этой работы рассмотрим вариант с отдельными вариантами для каждого ученика. Причем недостаточно сделать задачи для одной группы, так как тогда следующая за ней группа уже будет знать ответы. Поэтому необходимо делать отдельный вариант на каждого человека на курсе (в идеале, на каждого отдельного человека). Если в контрольной 4 задачи, а в группе 25 человек, то требуется создать 100 вариантов. Это может сделать очень трудолюбивый преподаватель, либо же это может быть сделано автоматически, с помощью алгоритмов генерации задач.

### ****Автоматическая генерация задач****

Такой автоматический генератор задач (или просто генератор) должен удовлетворять некоторым требованиям:

* Варианты не должны сильно отличаться по сложности
* Исходные данные и ответ должны быть “красивыми”, то есть не должны быть слишком большими или слишком маленькими
* Вариантов ответов может быть больше одного (иногда бесконечное количество), генератор должен быть способен корректно их обработать

В их соблюдении есть некоторые сложности и они варьируются в зависимости от задачи. Для обеспечения “красивости” данных в некоторых задачах можно просто генерировать “красивые” исходные данные – ответ сгенерируется посредством формул. В большинстве же случаев придется решать второстепенную задачу, уникальную для каждого типа задачи.

### ****Простой способ генерации общего уравнения прямой****

Для создания задач на тему прямых на плоскости можно оперировать общим уравнением прямой:

В таком случае, в большинстве типов задач для генерации прямой будет достаточно задать 3 случайных числа, удовлетворяющим условиям:

1. Коэффициент Aдолжен быть положительным;
2. Все числа должны быть взаимно простыми;
3. Все числа должны быть целыми.

Последнее важно для обеспечения “красивости” условия задачи и ответа на него.

Для данных целей можно сгенерировать 3 числа в определенном диапазоне, а затем провести над ними операцию нормирования:

1. Если А отрицательный – умножаем все коэффициенты на -1;
2. Делим каждый из коэффициентов на НОД всех трех.

### ****Нахождение НОД.****

НОД двух чисел находится с помощью алгоритма Эвклида:

1. Большее число делим на меньшее;
2. Если делится без остатка, то меньшее число и есть НОД;
3. Если есть остаток, то большее число заменяем на остаток от деления;
4. Переходим к пункту1.

Нахождение НОД трех чисел:

Следовательно, имея алгоритм нахождения НОД для двух чисел, его можно с помощью рекурсии преобразовать в алгоритм находения НОД произвольного числа чисел.

### ****Другие примеры генерации задач****

Однако не все задачи получится сгенерировать таким образом. В некоторых случаях использование данного способа приведет к дробным, а иногда даже к иррациональным ответам.

Например, в задаче нахождения расстояния от точки до прямой расстояние в общем случае будет числом “некрасивым”. В общем случае эта задача звучит так:

Дана прямая, заданная уравнением:

Найти расстояние от нее до точки .

Для решения этой задачи необходимо обратиться к формуле нахождения расстояния:

В данном случае числитель не представляет проблем, так как если все числа будут целыми, числитель тоже будет целым. Для того, чтобы в знаменателе не получилось иррациональное число, необходимо соблюдение условия:

,

где C–целое число. Тогда при извлечении корня получится целое число C, что в итоге даст дробное значение расстояния.

Такие тройки чисел называются пифагоровыми тройками. Формула Эвклида является основным средством их построения. Согласно ей для любой пары чисел m и n (m>n) целые числа , , образуют пифагорову тройку. Любая примитивная тройка получается из пары взаимно простых чисел mи n, одно из которых четно. Все остальные (не примитивные) тройки могут быть образованы путем умножения целого числа kна A, B и C:

, ,

Таким образом, можно расписать алгоритм генерации такой задачи:

1. Генерируются числа m, n и k;
2. Вычисляются A, B, C по формулам Эвклида;
3. Выбираются случайные числа C, и ;
4. Проводится операция нормирования коэффициентов прямой.

В получившейся задаче исходные данные будут состоять из целых чисел, а ответ будет рациональным.

### ****Генерация ответов к задачам****

Когда условия задачи уже сгенерированы, есть возможность сгенерировать ответ. Такой подход имеет ряд недостатков:

* Задача может иметь несколько или даже бесконечное множество решений
* Ответ нужно будет где то хранить и эту информацию будет возможно извлечь

Поэтому целесообразнее (а в некоторых случаях других вариантов просто нет) не генерировать ответ заранее. Вместо этого, введенный учеником ответ можно подставлять в готовую формулу проверки решения.

****Глава 2. Реализация системы тестирования****

### ****Краткое описание внутреннего устройства:****

1. Среда разработки: Visual Studio Community 2019;
2. Язык программирования: C#;
3. Интерфейс программирования приложений Windows Forms
4. Управление списком заданий осуществляется в классе TaskService;
5. Все генераторы лежат в папке StraightLineTaskGenerators и унаследованы от класса StraightLineTaskGenerator;
6. Логика перехода между вопросами реализуется в основной форме приложения - StartForm. см. методы StartTest, ShowTestResult.

### Интерфейс

При запуске программы открывается диалоговое окно для ввода фамилии, имени и группы тестируемого (Рис. 1).

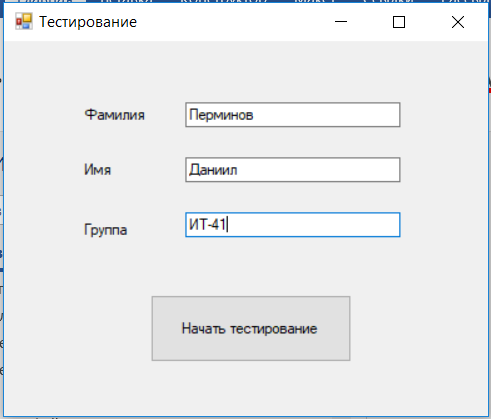


Рис. 1

После окончания ввода данных ученик нажимает на кнопку “Начать тестирование” и программа открывает окно тестирования (Рис. 2)

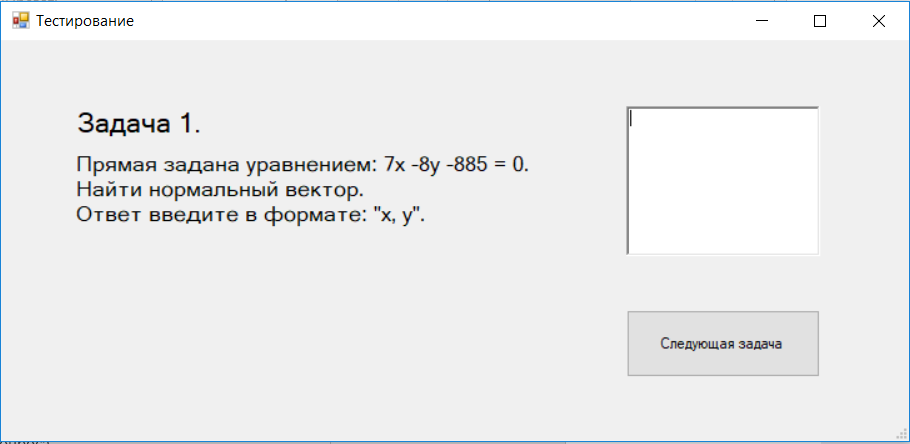


Рис. 2

### Использованные функции геометрических типов данных и операций над ними.

Геометрические типы:

1. Точка в двухмерном пространстве описывается двумя обыкновенными дробями (Mx и My);
2. Линия в двухмерном пространстве описывается тремя обыкновенными дробяи (A, B и C);
3. Отрезок в двухмерном пространстве описывается двумя точками (см. пункт 1).

Функции операций над геометрическими типами данных

1. Проверить, принадлежит ли точка линии

bool IsPointOnLine(Fraction x, Fraction y)

1. Форматирование уравнения прямой

void NormalizeLine()

1. Форматирование строки в виде Ax + By + C = 0

string GetLineString(Fraction a, Fraction b, Fraction c)

1. Форматирование точки в виде (x, y)

string FormatPoint(Fraction x, Fraction y)

1. Проверить, является вектор нормальным к прямой

bool IsVectorNormalToLine(Fraction x, Fraction y)

1. Проверить, является вектор направляющим к прямой

bool IsVectorParallelToLine(Fraction x, Fraction y)

1. Найти расстояние от точки до прямой

Fraction GetDistanceFromPoint()

### Примеры решения разных типов задач и алгоритмы генерации задач программой

В этом разделе речь пойдет о том, как формируется задача и как осуществляется проверка правильности ответа введенного учеником.

Прямая задана уравнением общего вида: Ax + By – C = 0

1. **Пересекает ли эта прямая отрезок [AB], если ,**

Решение

Последовательно подставляем координаты точкек A и B в уравнение прямой f(x, y), если полученные значения будут иметь различный знак, значит прямая с отрезком пересекаются.

Алгоритм генерации задания программой:

1. Задается случайно исходная прямая;
2. Задается случайно 2 точки  **и**
3. Вычисляются значения и
4. Если знаки совпадают, то отрезок не пересекает прямую, иначе пересекает.
5. Программа проверяет ответ ученика.

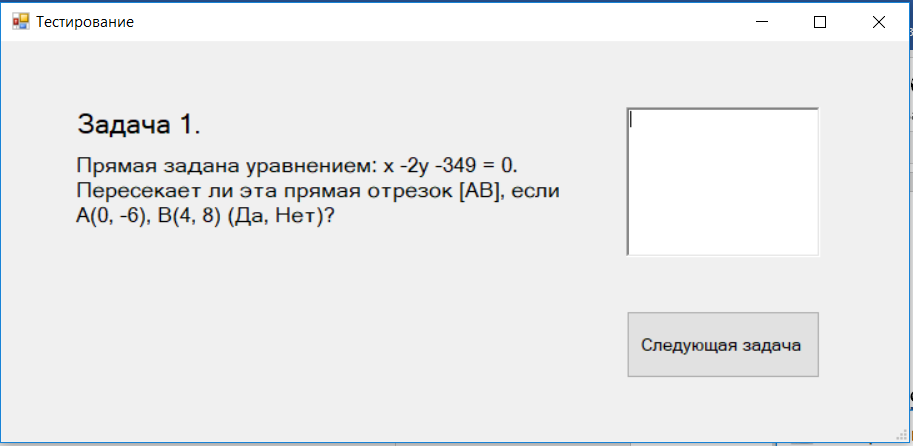


Рис.

1. **Найти направляющий вектор M(x, y).**

Решение

Пусть прямая задана общим уравнением Ax + By + C = 0 в прямоугольной декартовой системе координат. Тогда мы знаем, что вектор M(-B, A) является направляющим вектором прямой.

Алгоритм генерации задания программой:

1. Прямая задается случайным образом;
2. Ученик вводит ответ;
3. Программа ищет направляющий вектор M(-B, A) и сравнивает его с ответом ученика.

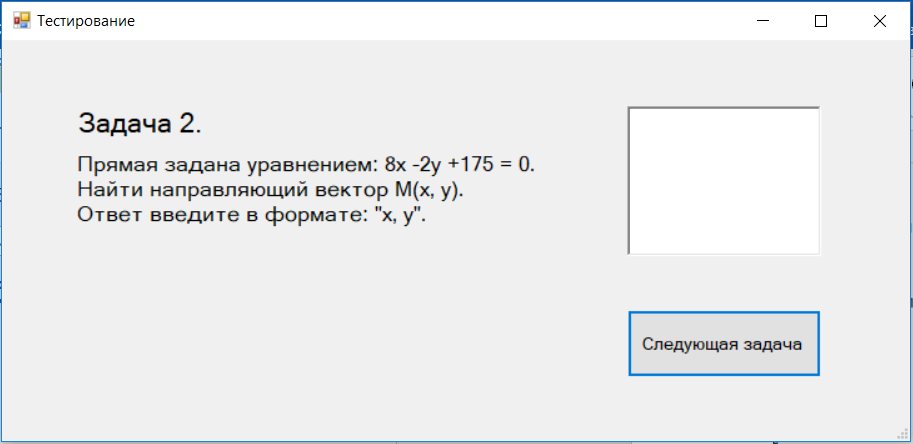


Рис. 4

1. **Найти расстояние от точки M0 (3, 5) до прямой**

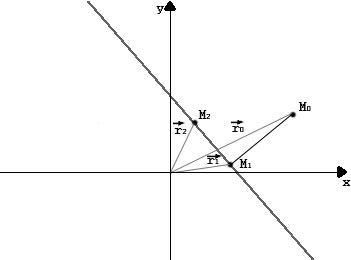
****

Рис. 5

Решение:

Опустим перпендикуляр к прямой из *M0.* Обозначим точку *M1(x1,y1)*. Пусть *M2(x2,y2) –* некоторая точка на прямой *l,* отличная от *M1(x1,y1)*. Тогда уравнение прямой запишем в нормальной векторной форме *A(x- x2)+B(y- y2)=0,* где *С= -Ax2-By2*, а  - вектор нормали. Или в векторной форме *(**)=0.* Очевидно справедливо следующие равенство , причем , поэтому . Умножив обе части равенства на вектор , получим (Рис. 5).

Так как точка *M1* лежит на прямой *l*, то , и следовательно. Подставляя в исходное равенство, найдем . Отсюда и *ρ(M0,l)=.* Переходя к координатной форме записи и учитывая, что *С= -Ax2-By2*, , имеем:

*ρ(M0,l)==.*

Алгоритм генерации задания программой:

1. Случайно задается точка;
2. Для генерации прямой используются заранее описанные пифагоровы числа (тройки), C задается случайно.
3. Ученик вводит ответ;
4. Программа считает расстояние по формуле ρ(M0,l)=и сравнивает получившийся ответ с введенным учеником ответом.

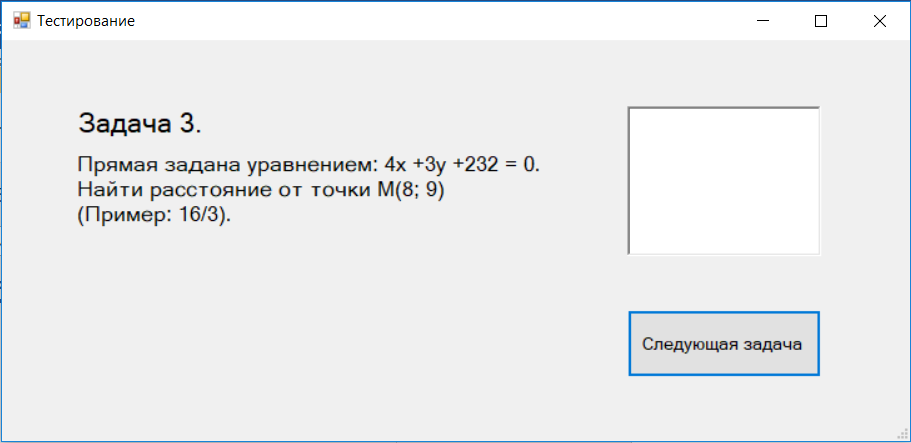


Рис. 6

1. **Найти 2 точки, лежащие на этой прямой.**

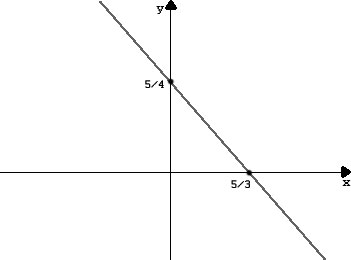
****

Рис. 7

Решение: Нам дана прямая, генерируемая без нулевых коэффициентов, а значит она пересекает обе координатные прямые. Таким образом, можно найти 2 точки (Рис. 7):

1. Возьмем *x = 0*, тогда

*By + C = 0*

*y = -C/B*

1. Возьмем *y = 0*, тогда

*Ax + C = 0*

*x = -C/A*

Так, если прямая задана общим уравнением *Ax + By + C = 0*, то имея только уравнение прямой мы можем найти 2 точки, принадлежащие это прямой *(0, -C/B) и (-C/A,0*).

Алгоритм генерации задания программой:

1. Случайно задаются коэффициенты A и B;
2. Высчитывается C = A \* B, далее C умножается на случайное целое число в диапазоне [-9, 9];
3. Ученик сам находит 2 точки;
4. Координаты подставляются в исходное уравнение прямой, если тождество сохраняется, то ученик ответил правильно, иначе ответ неверен.

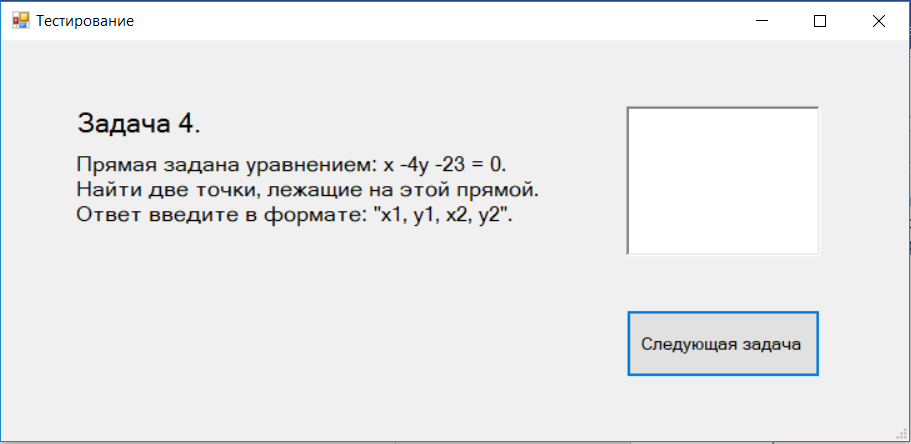


Рис. 7

1. **Найти нормальный вектор N(x, y).**

Решение

Пусть прямая задана общим уравнением Ax + By + C = 0 в прямоугольной декартовой системе координат. Тогда мы знаем, что вектор N(A, B) является нормальным вектором прямой.

Алгоритм генерации задания программой:

1. Прямая задается случайным образом;
2. Ученик вводит ответ;
3. Программа ищет нормальный вектор N(A, B) и сравнивает его с ответом ученика.

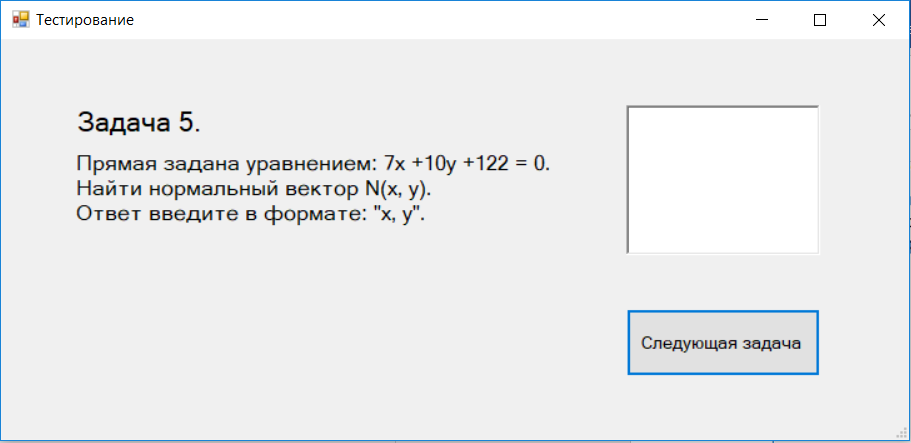


Рис. 8

1. **Составить уравнение перпендикуляра, опущенного на эту прямую из точки M(Mx, My).**

Решение

Направленный вектор искомой прямой, это вектор нормали исходной прямой.

, где *(x, y),* координаты заданной точки, а *(m,n)*коэффициенты вектора нормали прямой *l.*

*-* Общее уравнение прямой.

Алгоритм генерации задания программой:

1. Прямая задается случайно.
2. Точка задается случайно.
3. Прямая проходящая через точку является перпендикуляром для исходной прямой, если точка принадлежит искомой прямой и направляющий вектор искомой прямой совпадает с нормальным вектором исходной прямой.

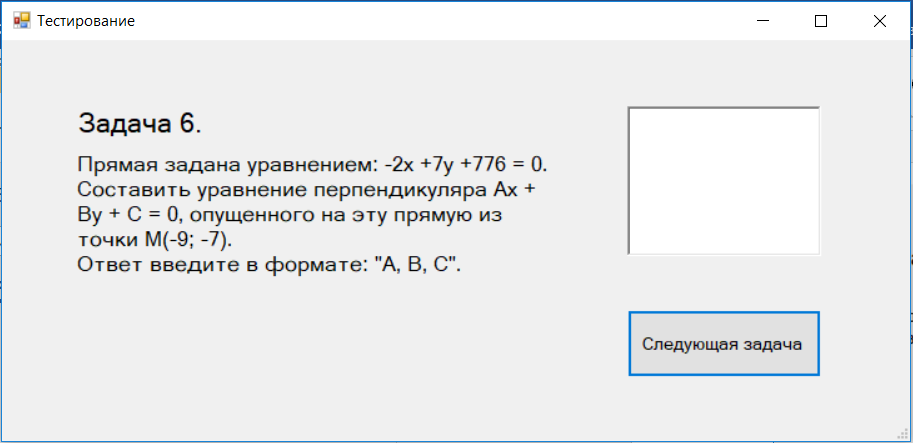
**

Рис. 9

Вывод результатов

После того, как ученик проделал все предложенные ему задания программа выводит результаты (Рис. 10). В них указывается численно и в процентном соотношении на сколько вопросов ученик ответил правильно, на сколько неправильно и сколько времени потратил на тестирование.

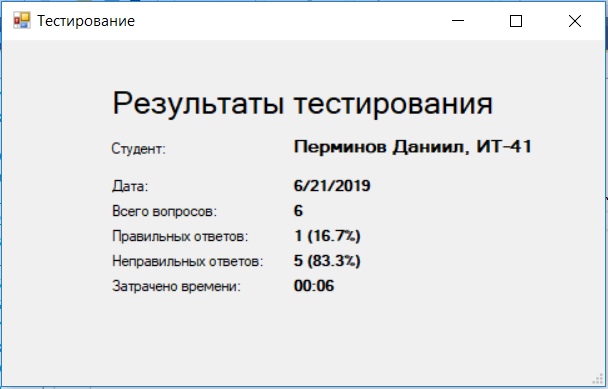


Рис. 10

Заключение

Выполнена поставленная задача разработки интерфейса, который позволяет ученику легко ориентироваться в приложении «Система тестирования» при его прохождении.

Разработаны методы генерации заданий. Трудоемкий процесс подбора заданий автоматизирован. Благодаря этому преподавателю не нужно тратить время на разработку различных вариантов тестирования.

Благодаря открытости тестирования, практически полностью исключена возможность угадывания ответов тестируемым. Исходные значения для теста каждый раз новые, что исключает возможность списывания.

Генерация исходных данных реализована таким образом, что при решении заданий тратится минимум времени на побочные вычисления. А именно при решении заданий получаются рациональные ответы. А проблема экономии времени актуальна как для преподавателей, так и для учеников.

Разработаны уникальные методики проверки правильности ответов. Причем особенность программы такова, что она не решает задания параллельно с тестируемым учеником, а проверяет правильность введенных им ответов. Практически каждое задание имеет как индивидуальную схему генерации заданий, так и уникальную систему проверки правильности введенных ответов.

Список используемой литературы

1. Математическая энциклопедия (в 5-и томах), Москва, «Советская Энциклопедия», 1982 г.
2. Прасолов В.В. Задачи по планиметрии. — 4-е изд., дополненное — М.: Изд-во Московского центра непрерывного математического образования, 2001.
3. А. Я. Архангельский, [100 компонентов общего назначения библиотеки Delphi 5](http://progbook.net/delphi/213-100-komponentov-obshhego-naznachenija.html), Бином, 1999 — 272 с.
4. Бобровский С.И., Delphi 7: Учебный курс, Питер, 2005 — 736с.
5. Кожушко В.В., Веременюк В.В., Практикум по математике: Подготовка к тестированию и экзамену Изд. 6-е, доп., ТетраСистемс, 2009 г, — 200с.
6. Шаповалов А.И., Зинченко Ю.Е., [Методика проверки хода решения математических задач](http://masters.donntu.edu.ua/2007/fvti/shapovalov/library/article1.htm). Доклад на первой Всеукраинской научно-технической конференции "Информационные процессы и технологии" г. Севастополь, 2007 г.
7. Ермакова М.Г., Андреева Л.Е. Вопросы разработки тестирующих программ. Информатика и образование. – 1997. №3.
8. Пак Н.И., Филиппов В.В. О технологии создания компьютерных тестов. Информатика и образование. – 1997. №5.
9. Родионов Б.У., Татур А.О. Стандарты и тесты в образовании. М., 1995.
10. Научная статья “Автоматическая генерация задач”, журнал “Компьютерные инструменты в образовании”

<https://cyberleninka.ru/article/n/avtomaticheskaya-generatsiya-zadach>