Петрозаводский государственный университет
Математический факультет
Кафедра прикладной математики и кибернетики

Дипломная работа

Использование платформы .NET для разработки обучающих систем

Выполнил:
студент 4-го курса группы 22403
М. М. Пробичев

_____
(подпись)

Научный руководитель:
д. т. н. профессор В. А. Кузнецов

_____
(оценка руководителя)

_____
(подпись)

Представлена на кафедру:
« ____ » _____ 2015 г.

_____
(подпись принявшего работу)

Итоговая оценка _____

_____
(подпись)

Петрозаводск
2015

Содержание

Введение

На данный момент существуют как минимум три проблемы обучения. Во-первых, большие территории с неравномерным распределением населения. Во-вторых, время, сегодня у большинства современных специалистов время расписано по минутам. И тем не менее, без новых знаний, без обучения в течение всей жизни никому не обойтись. И даже вечернее и воскресное образование данную проблему не решает. В-третьих, проблема денег. Это в большей степени касается высшего образования и подготовки к нему. Конкурс на бесплатные места высок, а платное обучение далеко не всем по карману.

Дистанционное обучение помогает в решении вышеперечисленных проблем. Это обучение "на дистанции т.е. на расстоянии, когда преподаватель и обучаемый разделены пространственно. Именно они и делают дистанционное образование дешевым и общедоступным, открывая возможности общения на больших расстояниях.

Выделяют три вида дистанционных технологий, применяемых в процессе обучения. Первый вид — кейс-технология на основе бумажных носителей. Это в первую очередь учебно-методические пособия, называемые рабочими тетрадями, которые сопровождаются тьютором. Тьютор поддерживает со студентами телефонную, почтовую и др. связь, а также может непосредственно встречаться со студентами в консультационных пунктах или учебных центрах. Вторая технология —- телевизионно-спутниковая. Она очень дорогая и пока мало используется. Главный ее недостаток - слабая интерактивность, то есть обратная связь. И, наконец, третья технология — это интернет-обучение, или сетевая технология.

В настоящее время объёмы учебной информации растут и часто традиционные способы передачи информации становятся менее эффективными. Использование современных информационных технологий раскрывает большие возможности по обучению.

Компьютерные технологии обучения представляют огромное количество способов передачи учебной информации: текст, графика, видео и аудио материалы, но особенно удобны интерактивные учебники. Они выполняют задачу усовершенствования и облегчения труда педагога. Преподаватель тратит меньше времяни на обучение, повторение одного и того же материала по многу раз, за него всё делает программа, а он только наставляет своего ученика, даёт советы и отвечает на дополнительные вопросы.

Обучающая программа позволяет, помимо передачи информации, ещё и тестировать ученика и следить за его результатами изменяя сложность заданий. Это в свою очередь сокращает количество времени потраченное на проверки практических работ.

Я считаю, создание обучающих программ актуальной темой, так как обучающие программы позволяют экономить время преподавателей и предаставляют новый уровень передачи информации, что оказывает существенное влияние на качество дистанционного обучения. Обучении становится нагляднее и увлекательней [1].

На данный момент обучающие программы самого разного назначения и уровня обучения становятся всё популярнее. Можно найти обучающие программы по иностранному языку, математики и т.п. В рамках данной дипломной работы будет рассматриватся обучение алгоритмам. В этом так же заключается актуальность данной дипломной работы, мной не были найдены в свободном доступе обучающие программы по алгоритмистике.

Данная дипломная работа носит практический характер. Целью работы является создание прикладного программного пакета для написания обучающей программы по различным алгоритмическим задачам.

Также были поставлены следующие задачи:

1. изучить теорию обучающих программ;
2. изучить платформу .Net и её использование для разработки обучающих программ;
3. спроектировать прикладной програмный пакет;
4. разработать прикладной программный пакет.

Практическая значимость данной работы состоит в написании обучающей программы и создании инструментария для последующего расширения данной программы.

1 Теория обучающих программ

1.1 Общие сведения

Как уже говорилось ранее, степень важности обучающих систем в процессе нашей жизнедеятельности постоянно растёт, и в последние годы на рынке программного продукта стало появляться всё больше обучающих систем, охватывающие различные сферы нашей деятельности. Большое внимание здесь уделяется автоматизированным обучающим системам (АОС), о которых речь пойдет дальше.

Программа имеет право называться системой только в том случае, если она представляет собой набор взаимосвязанных компонентов, каждый из которых выполняет свои конкретные функции. Также необходимым фактором является присутствие в программе компоненты интерфейса пользователя, которая играет немаловажную роль для обратной связи с обучаемым, и компоненты, ответственной за реализацию необходимой логики решения задач.

Ниже, на рис.1, приведена классификация структурного построения АОС.

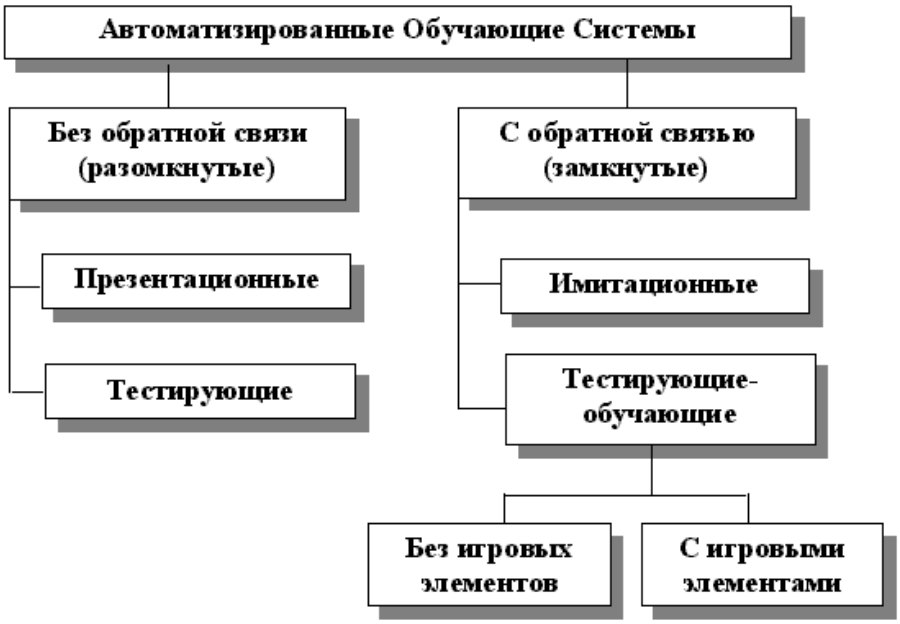

Рисунок 1 — Классификация АОС.

Как мы видим, все системы делятся на два больших типа - разомкнутые и замкнутые. К разомкнутым относятся системы, в которых не учитывается обратная реакция пользователя. Примером презентационной системы может служить, например, набор слайдов, дающий информацию о каком-нибудь языке программирования. В тестирующей системе, как несложно догадаться по названию, фигурирют тесты, которые пользователь проходит для проверки своих знаний по соответствующему разделу того или иного предмета. Теперь остановимся поподробнее на замкнутых системах.

Главным достоинством этого типа систем является обратная связь с обучаемым, т.е. на любое действие пользователя происходит конкретное действие системы, и эти действия непосредственным образом зависят от действий пользователя. Здесь также выделяется две основных основных разновидности таких систем — имитационные и тестирующие-обучающие.

Главным инструментом имитационных систем является моделирование реальной ситуации в той или иной сфере предметной области. В пример можно привести различные имитаторы и игровые тренажеры. В частности, имитационная программа компании Maxis позволяет имитировать развитие и управление городом. Здесь имитируются различные ситуации, начиная от размещения промышленных предприятий и транспортных сетей и заканчивая моделированием экстремальных ситуаций и путей их ликвидации (подробнее см. в источнике [2]).

Основыми задачами тестирующих-обучающих систем являются передача обучаемому необходимых знаний в изучаемой области и постоянный контроль полученных знаний, т.е. обеспечение обучаемого различными тестами, с помощью которых можно выявить уровень усовяемости изученного материала. Именно

от результатов этих тестов и зависит дальнейшее поведение системы, а именно — если ученик успешно сдал тест по пройденной теме, то можно переходить к следующей, если таковая имеется, иначе нужно предпринимать другие действия, например, такое действие как повторное обучение по данной теме.

Примеры обучающих систем можно посмотреть по следующим ссылкам: обучение английскому языку [3] и обучению алгебре [4]

Основная рекомендации по разработке обучающих программ взяты из электронных ресурсов: [5] и [7]

Обучающая система, написание которой ведется в рамках данной дипломной работы, относится как раз к тестирующим-обучающим системам.

2 Технологии пакета обучающих программ

Платформа .Net обладает большим инструментарием по работе с объектно-ориентированными программами, далее будет разобраны основные используемые технологии. Подробная информация в источнике [6].

2.1 Классы, члены класса и экземпляры

Классы — это основа объектно-ориентированного программирования. Класс представляет собой инкапсуляцию данных и методов для их обработки. В данной обучающей программе наиболее интересным является класс для инкапсуляции данных и методов конкретных модулей, благодаря чему, можно работать со всеми модулями в общем виде используя экземпляры класса. Каждый экземпляр будет предоставлять доступ к конкретному модулю посредством членов самого класса. В C# имеются следующие члены классов:

- поле — член-переменная, которая содержит некоторое значение;

- метод — реальный код, воздействующий на данные объекта;
- свойства — поля, которые, по сути, являются методами, так называемые smart fields;
- константы — поле, значение которого нельзя изменить;
- индексаторы - позволяют индексировать экземпляры класса или структур так же, как массивы. Напоминают свойства (так же используют аксессоры для доступа к данным), но их методы доступа принимают параметры;
- события — вызывает исполнение некоторого фрагмента кода. Подробнее про события см. ниже.

Как уже говорилось выше, обучающая программа содержит класс описывающий данные и методы модулей (обучение конкретному алгоритму). Создавая экземпляры данного класса и устанавливая нужные поля и свойства, можно использовать общий метод для всех модулей. Рассмотрим пример. Имеется класс

Листинг 1: Пример класса

```
1    public Task{
2      private someDataType data;
3      Task(someDataType Data){
4        this.data = Data;
5      }
6
7      public void Execute(){
8      /* необходимый... кодиспользущийнашиданные ... */
9      }
10   }
```

Как видим, в классе есть поле с приватными данными, устанавливаемыми при создании экземпляра, и есть метод Execute() который будет использовать

данные. Создаём набор экземпляров класса MyTask соответствующий нескольких задачам. Добавляем набор в какую-нибудь коллекцию. Теперь, взяв любой элемент из коллекции можно выполнить метод Execute(). В разрабатываемой обучающей программе набор классов модулей передаётся в элемент управления DropBox. При нажатии на кнопку на форме берётся выбранный экземпляр класса и выполняется метод отображения первого диалогового окна. Данный подход очень удобный и облегчает программирование, делая программу структурированной.

Ещё одна особенность, связанная с классами в разрабатываемой обучающей программе, это создание объектов внутри класса, вместо передачи уже созданных объектов. В данном случае, не используемые объекты не создаются раньше времени и не занимают ресурсы (время, память). Например, вместо того, чтобы хранить в экземпляре класса экземпляр формы, можно только указать тип формы (который может отличаться от экземпляра к экземпляру) и создавать её в каком-нибудь методе. Для этого, тип формы хранится в поле тип Type, а в самом методе вызывается Activator.CreateInstance(formType).

2.2 Интерфейсы

Интерфейсы в C# предоставляют высокую степень абстракции. Можно определить основные члены и реализовывать разные классы на основе интерфейса. Тогда, можно использовать объекты, ничего не зная об их конкретном устройстве, используя члены интерфейса.

Рассмотрим конкретный пример из разрабатываемой обучающей программы.

Листинг 2: Пример интерфейса

```
1    public interface IQuest
2    {
3        string Name { get;}
4        string Question { get;}
5        string Answer { get;}
6        Boolean CheckAnswer(string answer);
7    }
```

Как видим, интерфейс простой, описывает задания для пользователя. Конкретная реализаций CheckAnswer позволит проверять любой вид вопросов (да-нет, выбор из множества, открытый вопрос и т.д.). Можно просто вызывать метод CheckAnswer, не думая, для какого типа вопросов нужно проверить ответ.

2.3 Формы и элементы управления

Немного поговорим о формах и элементах управления. .Net обладает огромным набором всевозможных элементов управления для графических приложений, это кнопки, меню, выпадающие списки и т.п. Но иногда, стандартных элементов управления не хватает, либо необходимо как-то модифицировать стандартные элементы управления. Тогда можно создать пользовательский элемент управления (user control), который может состоять из стандартных элементов, либо наследоваться от существующего элемента управления с целью расширения возможностей. Также, использование собственных контролов позволяет вынести с нескольких форм общие группы элементов, написать соответствующие обработчики и использовать. Так программа становится компактнее, код перестаёт дублироваться и настроенный компонент можно использовать в любом месте вашего приложения. Хорошей практикой создания собственных контролов является вынесение их в отдельный проект. Тогда контролы можно будет

использовать, где захотите. В разрабатываемой программе отдельно вынесены контролы для ввода ответа на вопросы. В форме с вопросом, нужный контрол программно встраивается в нужном месте.

2.4 События

События позволяют вызывать исполнения некоторого фрагмента кода. События позволяют организовать взаимодействие с пользователем: пользователь нажал кнопку — возникло соответствующее событие, пользователь двинул мышку — возникло новое событие. События также могут возникнуть и в результате работы программы. Можно описать собственное событие и соответствующий делегат обработчик и использовать их в программе, для вызова и обработки события. Рассмотрим на примеры обучающей программы.

Листинг 3: Пример событий

```
1  public class QuestEvents
2  {
3        public delegate void QuestEventHandler(object sender,
           QuestEventArgs e);
4
5        public class QuestEventArgs : EventArgs
6        {
7            public IQuest Quest
8            {
9                get;
10               private set;
11           }
12           public Coordinate Coord
13           {
14               get;
15               private set;
```

```
16              }
17              public QuestEventArgs(IQuest quest, Coordinate coord)
18              {
19                  this.Quest = quest;
20                  this.Coord = coord;
21              }
22          }
23      }
```

Представленный выше класс нужен для генерации и обработки события. Событие должно возникать, когда программа собирается задать пользователю вопрос.

Реализовав обработчик события и вызвав событие в процессе работы алгоритма позволит задать пользователю вопрос, проверить ответ и в случае успеха продолжить выполнение алгоритма.

2.5 Многопоточность

Но напрямую, просто вызвав событие, поставить программу на паузу не получится, так как нужно дождаться ответа пользователя, прежде чем переходить к следующему шагу алгоритма. Возникает проблема: с одной стороны у нас пользовательский интерфейс, на котором отображается вопрос и принимается ответ, с другой стороны реализация какого-нибудь алгоритма. Мы встретили две задачи которые нужно выполнять отдельно друг от друга, при этом периодически одна задача будет ждать другую. Для решения такой проблемы можно использовать несколько потоков. Пользовательский интерфейс выполняется стандартным потоком, а для алгоритма мы создаём ещё один, выполнением которого можно легко управлять.

Потоки в C# представлены объектом Thread, подробнее о котором можете почитать в руководстве на сайте mdsn.com. Создав поток для алгоритма, мы может генерировать в нём событие, которое будет перехватывать основной поток. На это время, выполнение алгоритма приостанавливается, а на форме появляется вопрос. Вычисления продолжатся после ввода значения.

Ожидание ввода можно сделать при помощи следующей конструкции:

Листинг 4: Пример ожидания

```
1  while (!pressed)
2    {
3                System.Threading.Thread.Sleep(100);
4    }
```

Где $pressed = false$, пока не будет введён ответ.

Но, при использовании нескольких потоков может возникнуть проблема обращения потока к объекту, созданному в другом потоке (например, элементы управления). Но выход есть, необходимо использовать InvokeRequired и Invode (подробнее см. официальной документации):

Листинг 5: Пример решения проблемы с потоками

```
1    public delegate void SetQuestionCallback(string text);
2
3    // функциядляустановкивопросаиздр . потока
4        public void SetQuestionText(string text)
5        {
6                if (QuestionLabel.InvokeRequired)
7                {
8                    SetQuestionCallback deleg = new SetQuestionCallback
                         (SetQuestionText);
9                    this.Invoke(deleg, new object[] { text });
```

```
10                  }
11                  else
12                  {
13                      QuestionLabel.Text = text;
14                  }
15          }
```

QuestionLabel.InvokeRequired проверяет, из того ли потока происходит обращение, и если не из того, то указывает какой поток должен исполнять метод.

3 Пакет Прикладных Программ "AlgoTeacher"

3.1 Общая структура

В общем виде наш пакет состоит из следующих основных компонентов:

1. Меню выбора алгоритма (главное меню, позволяющее выбрать алгоритм)
2. Обучающий модуль (предоставление основной информации по теме)
3. Тестирующий модуль (тестирование пользователя по данной теме)

Програмный пакет имеет следующую структуру:

- AlgoTeacher.Forms — Формы:
  - MainForm— Главная форма;
  - InformationForm — Информационная форма, описывающая алгоритм;
  - IntroForm — Вступительная форма для соответствующего модуля;
  - LanguageForm — Форма выбора языка.
- AlgoTeacher.Events — События:

  - QuestEvents — Событие задачи вопроса пользователю
  - FillEvents — Событие заполнение данных на форме

- AlgoTeacher.Controls — Пользовательские элементы управления:
  - MatrixGridView — Модифицированый GridControl для работы с матрицами;
  - RotatingLabel — Модифицированый LabelControl с возможностью поворачивать текст;
  - RotatingLabelControlPainter — Вспомогательный класс для отрисовки поворачиваемого label'a;
  - QuestionControlBase — Базовая форма для ввода ответа;
  - StringQuestionControl — Форма для ввода ответа в виде строки;
  - TwoStringQuestionControl — Форма для ввода ответа в виде двух строк (например количество строк-столбцов);
  - ThreeStringQuestionControl — Форма для ввода ответа в виде трёх строк (например координаты и значение);
  - TwoVariantsQuestionControl — Форма для ввода ответа вида "да/нет".

- AlgoTeacher.Modules — Модули с алгоритмами решения конкретных задачь:
  - MatrixMultiplyForm — Форма алгоритма перемножения матриц;
  - KnapsackProblemForm — Форма алгоритма задачи о рюкзаке;
  - TransportTaskForm — Форма алгоритма транспортной задачи;
  - MatrixMultiply — Алгоритм перемножения матриц;
  - KnapsackProblem — Алгоритм задачи о рюкзаке;

  - TransportTask — Алгоритм транспортной задачи;

- AlgoTeacher.Model — Модели:
  - Coordinate — Класс представления и работы с координатами;
  - Matrix — Класс представления и работы с матрицей;
  - Task — Класс представления и работы с алгоритмом;
  - TaskHelp — Класс представления и работы с подсказками;
  - Information — Класс представления и работы с информацией по алгоритму.

- AlgoTeacher.Interfaces — Интерфейсы:
  - IQuest — Интерфейс классов для вопросов

- AlgoTeacher.Quests — Виды вопросов:
  - DoubleValueQuest — Вопрос с числом с плавающей точкой;
  - IntegerIntegerValueQuest — Вопрос с двумя целыми числами;
  - IntegerValueQuest — Вопрос с целым числом;
  - StringQuest — Вопрос со строкой;
  - YesNoQuest — Вопрос вида "да/нет";
  - RangeQuest — Вопрос с диапозоном значений.

Ниже рассмотрим некоторые элементы подробнее.

Главное меню (MainForm) является основной формой приложения, в данную форму можно вернуться из запущенных модулей и отсюда начинается путешествие по обучающей программе. Список на форме содержит набор реализованных

модулей, основанных на объекте Task. Про Модули будет подробнее написано ниже. Для добавления нового модуля нужно создать модуль (см. пункт Модули), а затем добавить их к списку, всё остальное программа будет делать сама. Выбрав алгоритм и нажав кнопку на форме, пользователь перейдёт непосредственно к обучению соответствующему алгоритму.

В программе предусмотрено использование разных языков (необходимо лишь добавлять соответствующие варианты текста). При этом новые текстовые данные будет загружены из локальной базы данных, либо из соответствующего текстового файла.

$IntroForm$ является своеобразным приветствием и предоставляет пользователю выбор: пройти обучение или сразу перейти к практическому заданию.

$InformationForm$ предоставляет пользователю информацию по данному алгоритму. Для изучения доступно 3 уровня подсказок:

1. Основная информация
2. Подробная информация
3. Разбор примера

В случае успешного усвоения информации можно перейти к прохождению теста.

На данном этапе мы переходим к форме, связанной с конкретным обучающим модулем. Данные для алгоритма генерируются случайным образом.

Тест заключается в вопросах 2-х уровней сложности:

1. Вопрос на выполнимость алгоритма
2. Группа вопросов по мере выполнения алгоритма

По сути, уровни вопросов - это шаги алгоритма.

Если пользователь неправильно отвечает на заданные вопросы, то ему выводится поясняющая информация, например выделяет нужные строки-столбцы матриц, выводит формулы и т.д. Если же пользователь совершает ошибки слишком часто, то программа предлагает пользователю вернуться в главное меню. Там он может либо выбрать другой алгоритм, либо заново пройти обучение.

При последовательных правильных ответах программа начинает задавать вопросы следующего уровня, пока не удостоверится, что пользователь со всем справился.

При расширении программы можно добавлять новые виды заданий. Для этого нужно использовать интерфейс IQuest и добавлять нужные контролы на основе реализованного задания.

Ниже будут подробнее разобраны сами модули, как их создавать и использовать, а также будет подробнее описаны логические компоненты программы.

3.2 Модули

Модулем в нашем случае является набор классов, созданных для конкретного алгоритма.

Модуль содержит в себе обучающую и тестирующую Функциональность. Модуль состоит из модифицированного алгоритма решения конкретной задачи и формы.

Алгоритм должен быть разбит на отдельные фукнции, выполняющие какую-то логически завершённую часть алгоритма.

Основой модуля объект Task, включающий в себя:

- Name - имя модуля
- _formType - тип формы для отображения тестирования алгоритма
- _introText - приветствие и общая информация по данному алгоритму, используется в IntroForm
- _help - экземпляр класса TaskHelp, собственно обучающая часть программы

Для добавления нового модуля необходимо выполнить следующие действия:

- Добавить вводную информацию/приветствие;
- Создать необходимый TaskHelp с обучающей информацией по данному алгоритму;
- Если нужно, то добавить собственные виды вопросов, при этом нужно реализовать IQuest и создать соответствующий элемент управления;
- Добавьте форму для тестирования вашего алгоритма, в обработчике которой выделите уровни вопросов. Форма создаётся на основе примеров. Для вопросов используйте имеющиеся Quest классы. Для создания формы можно использовать подготовленные пользовательские элементы управления;
- Добавьте класс с алгоритмом, в котором используйте QuestEvent и FillEvent для взаимодействия с пользователем;
- Создайте экземпляр класса с алгоритмом в форме и добавте нужные вызовы функций. В случае работы с циклами используйте отдельный поток;
- Добавте обработчики QuestEvent и FillEvent;

- Создаёте Task и добавьте его в список модулей на главной форме.

Task имеет несколько основный методов:

- ShowIntro() — открывает вступительную форму
- ShowInfo() — открывает обучающую форму с прекреплённой информацией
- ShowTest() — открывает тестирующую форму

При реализации IQuest нужно реализовать представление имени вопроса, сам текст вопроса и правильный ответ, а также реализовать функцию проверки ответа.

Для взаимодействия с пользователем нужно использовать событие questEvent(). Аргументом questEvent является QuestEventArgs(IQuest quest, Coordinate coord), при этом Coorditane — это класс координат вопроса, позволяющий указывать, на какую часть теста он ссылается. тест.

Для удобства разработки можно использовать приложенные шаблоны модулей с примерами.

Далее приводится пример модуля на основе алгоритма перемножения матриц.

4 Обучение алгоритмам средствами пакета прикладных программ

4.1 Алгоритм перемножения матриц

Рассмотрим работу нашей системы на алгоритме перемножения матриц.

В начале появляется окно выбора языка. Выбираем язык и переходим в главное меню.

Для того, чтобы начать обучение данному алгоритму, в главном меню в списке задач нужно выбрать задачу "Умножение матриц"(см. рис. 2).

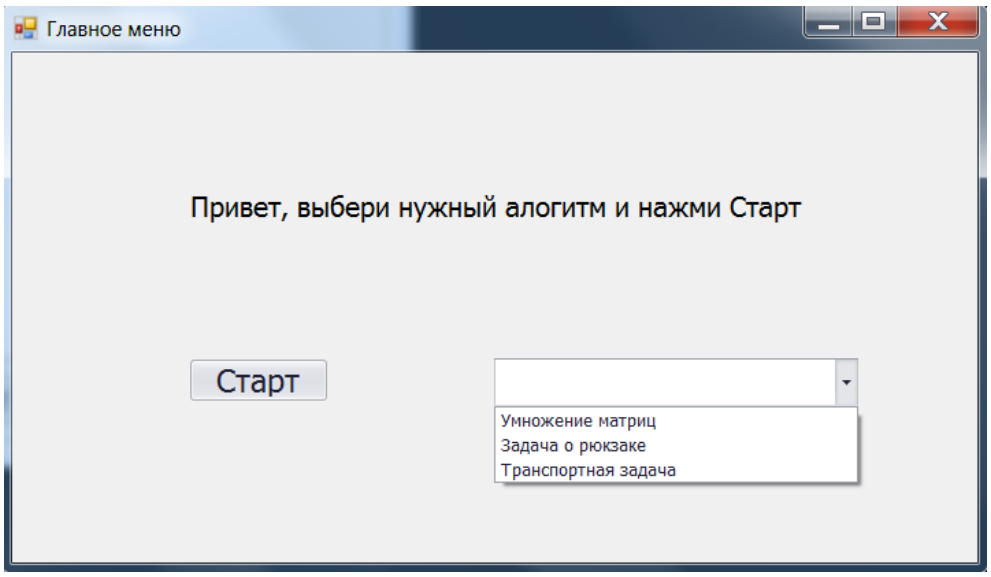

Рисунок 2 — Выбор задачи.

После того, как будет нажата кнопка "Старт произойдет переход на другую форму — форму приветствия, где обучающая система узнает о ваших желаниях по поводу обучения данному алгоритму (см. рис. 3).

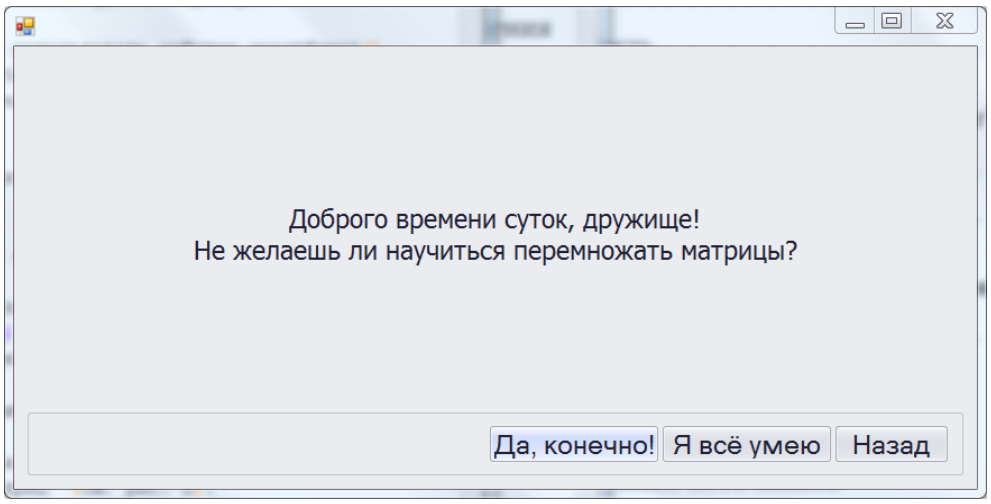

Рисунок 3 — Форма приветствия.

Если вы нажимаете на кнопку "Я всё умею то будете "переброшены"на форму перемножения матриц, где и начнете проходить тесты для подтверждения

ваших знаний, но об этом попозже. Если же вы только хотите овладеть данным алгоритмом, то выбираете кнопку "Да, конечно! после нажатия на которую вы перейдёте на форму с информацией о алгоритме перемножения матриц (см. рис. 4).

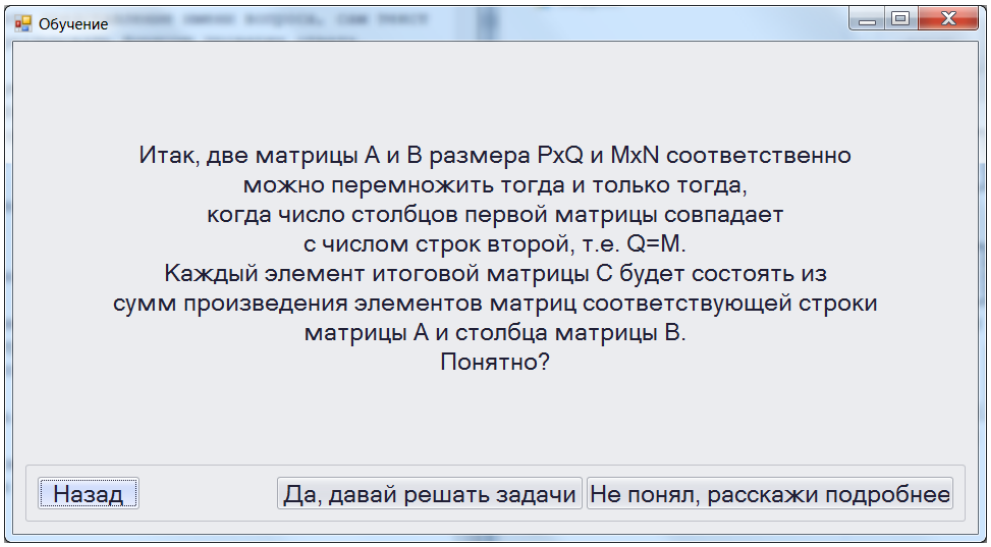

Рисунок 4 — Начальное обучение задаче.

Если вам будет не достаточно этого определения, то вы сможете получить об этом алгоритме дополнительную информацию, нажав на кнопку "Не понял, расскажи подробнее". Тогда на форме будет предложен другой текст, но уже с более подробной информацией. Если же и этого будет не достаточно для понимая данного алгоритма, то вам будет предложено пройти обучение заново. Однако, если вам удалось овладеть данным материалом, то у вас будет возможность перейти к тестам, нажав на кнопку "Давай решать задачи". Переходим на форму перемножения матриц, где нам предложены две матриц и задается вопрос — можно ли их перемножить (см. рис. 5)?

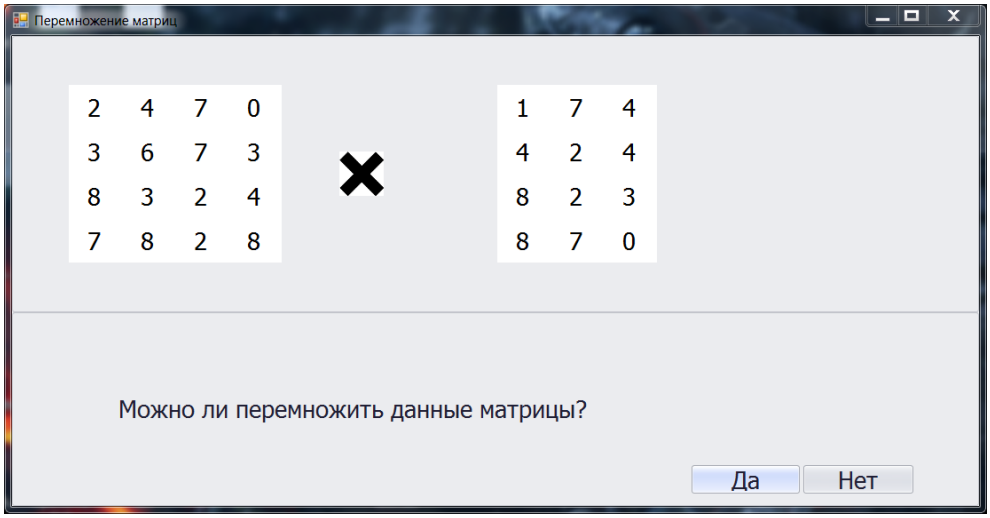

Рисунок 5 — Уровень 1.

Если мы отвечаем неправильно, то нам будет предложен тот же самый вопрос, но уже с другими матрицами. Но, ответив правильно, мы переходим на второй уровень, где нам опять предложены две матрицы, но в качестве ответа требуется размерность матрицы, которая получится в результате перемножения данных двух (см. рис. 6).

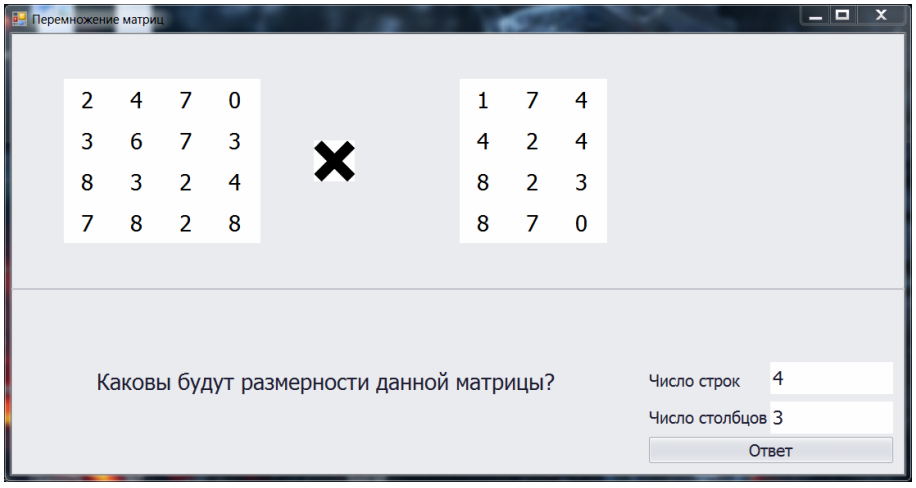

Рисунок 6 — Уровень 2.

Правильно ответив на вопрос 2-го уровня (см. рис. 7), мы переходим на третий уровень.

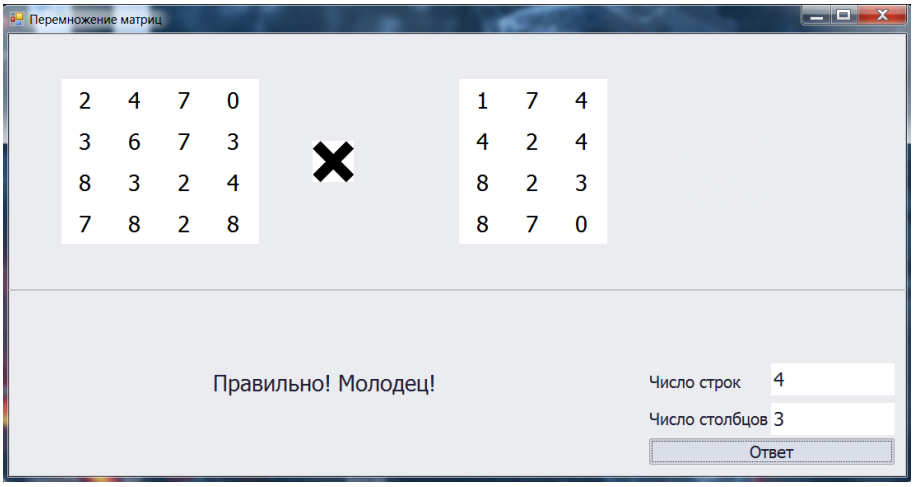

Рисунок 7 — Правильный ответ на вопрос 2-го уровня.

На третем уровне программа будет выборочно спрашивать о результате перемножения для каких-нибудь случайно выбранных элементав результирующей матрицы (см. рис. 8). Остальные элементы программа заполнит сама.

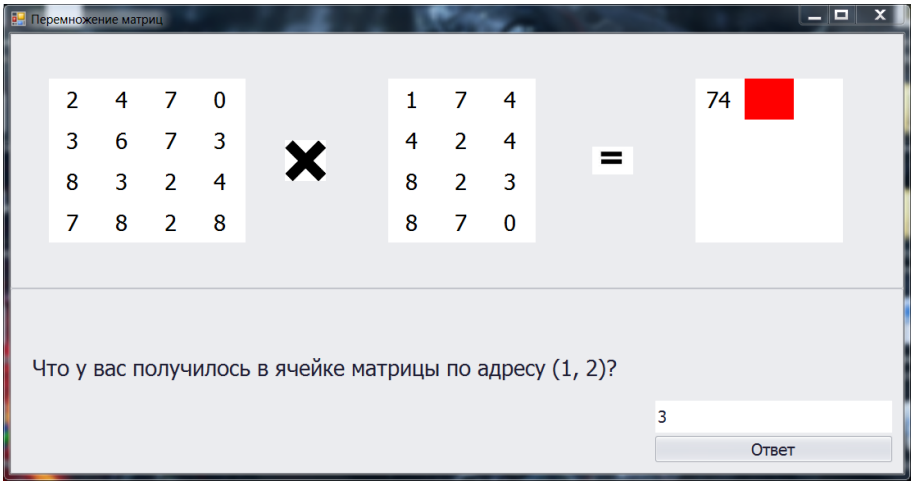

Рисунок 8 — Вопрос 3-го уровня.

В процессе, если пользователь отвечает не правильно, программа ему подсказывает. Сначала выделяет строку и столбец, которые используются для вычисления текущего элемента(см. рис. 9).

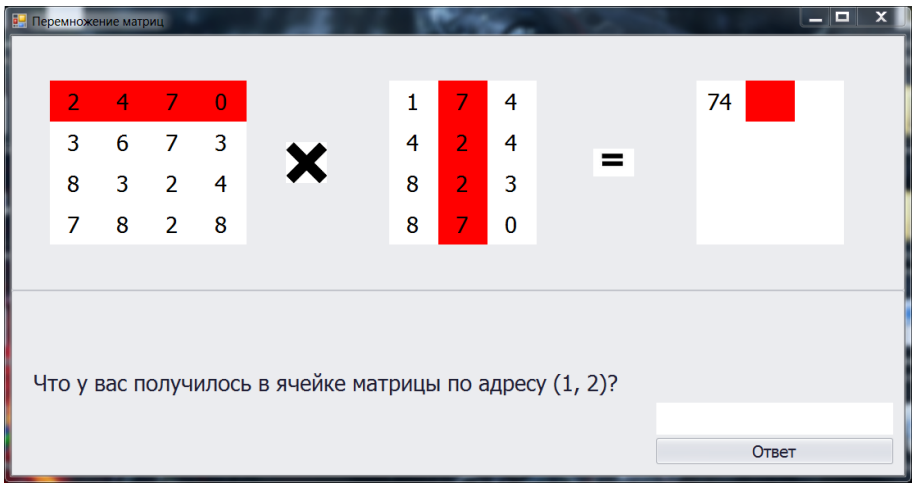

Рисунок 9 — Пример подсказки 1-го уровня.

Если и после этого пользователь неправильно отвечает, то выводится формула перемножения (см. рис. 10).

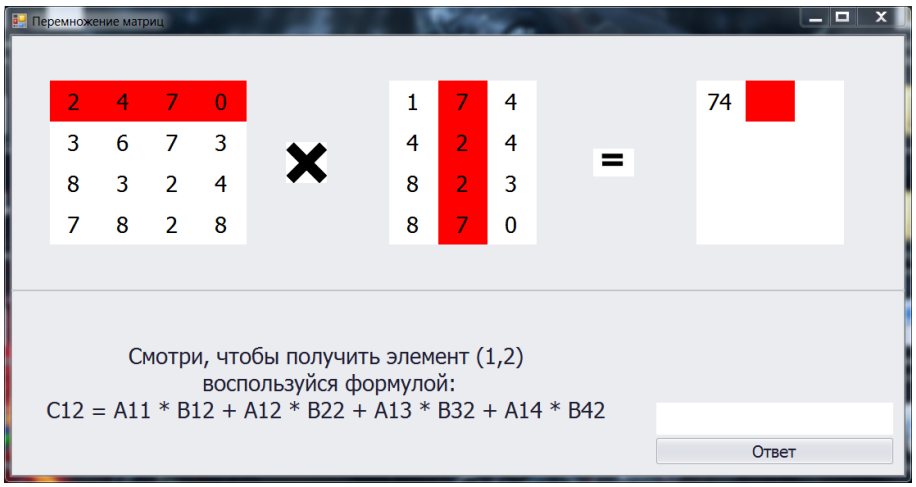

Рисунок 10 — Пример подсказки 2-го уровня.

На третий неправильный ответ, программа подставляет в формулу данные из матриц (см. рис. 11) и если после этого пользователь не может правильно ответить, программа предлагает пройти обучение ещё раз.

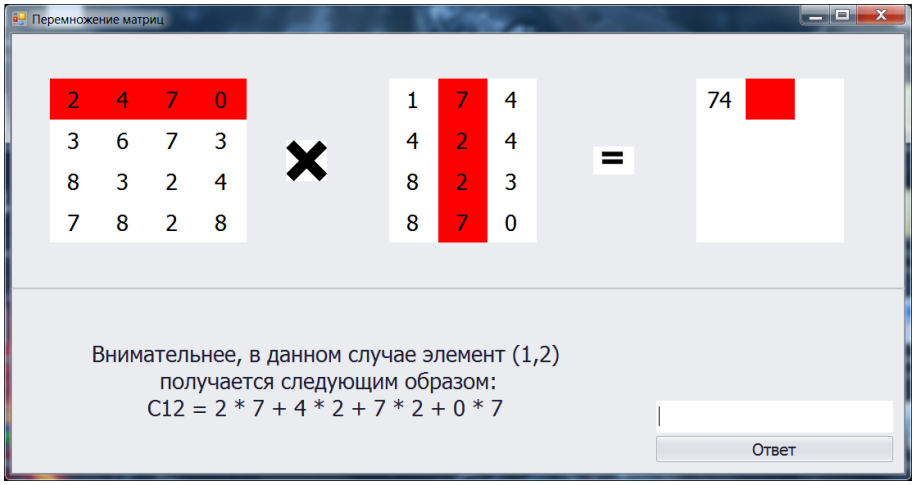

Рисунок 11 — Пример подсказки 3-го уровня.

По окончанию теста программа хвалит пользователя за успешное прохождение (см. рис. 12).

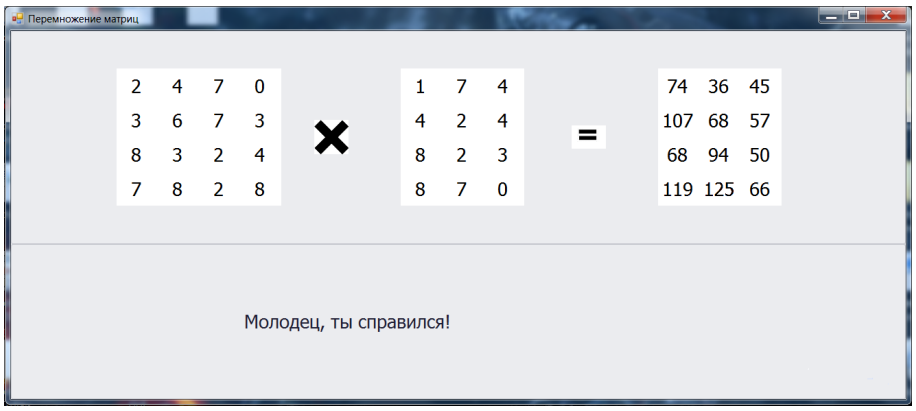

Рисунок 12 — Завершение теста.

Заключение

Обоснование целесообразности выбора .Net: В данной работе была поставлена цель написать библиотеку для создание обучающей программы, реализация которой может быть с помощью различных языков программирования. Был выбран язык программирования C#, входящий в платформу .Net по ряду следующих причин.

Во-первых, платформа .Net и язык программирования C# является широко распространёными и входят в учебную программу, а это значит, что большее количество людей смогут разобрается в написанной программе с целью добавления новых алгоритмов.

Во-вторых, язык C# был спроектирован таким образом, что C,C++,Java, JavaScript программисты легко могут на него перейти, что ещё больше расширяет аудиторию.

В-третьих, платформа .Net обладает обилием стандартных и сторонних средств для создания качественных объектно ориентированных приложений, что нужно при создании обучающей программы.

В-четвёртых, синтаксис C# проще синтаксиса C++, следовательно облегчается написание программ.

В-пятых, огромный вклад в выбор данной среды заключается в безопасности работы с типами, что являтся важным фактором избежания ошибок.

Как видно из вышеприведённых пунктов выбранный язык программирования C# показался наиболее удобным для выполнения поздавленных задач.

Текущее состояние: В ходе работы были изучена основная информация по созданию обучающих программ. На данный момент реализована прототип библиотеки: вывод обучающей информации, несколько уравней тестирования, вывод подсказок при тестировании и несколько других специальных возможностей. Создано средство для внедрения вопросов в алгоритм и механизмы взаимодействия

с пользователем. Созданы примеры обучающих программ, на основе алгоритмов перемножения матриц и решения транспортной задачи.

Перспективы: В дальнейшем планируется добавить несколько новых алгоритмов и добавить средства для работы с графами.

Проблемы: Основная проблема связана с визуализацией работы с графами. Стандартные средства .Net не дают такой функциональности и предётся разрабатывать собственные элементы управления.

Список использованных источников

[1] Дидактическая целесообразность применения компьютерных учебников и обучающих систем [Электронный ресурс].

URL:http://www.inoteach.ru/conf/ped/archive/143/6199/

[2] Принципы построения обучающих программ и их классификация [Электронный ресурс].

URL:http://scholar.urc.ac.ru/ped_journal/numero4/pedag/tsit3.html.ru

[3] Актуальность применения компьютерных обучающих программ на примеры английского языка [Электронный ресурс].

URL:http://cyberleninka.ru/article/n/aktualnost-primeneniya-kompyuternyh-obuchayuschih-programm-v-obuchenii-angliyskomu-yazyku

[4] Использование компьютерных обучающих программ в процессе преподавания курса высшей алгебры [Электронный ресурс].

URL:http://www.dissercat.com/content/ispolzovanie-kompyuternykh-obuchayushchikh-programm-v-protsesse-prepodavaniya-kursa-vysshei-#ixzz3Ly3KWLxE

[5] Компьютерные обучающие системы [Электронный ресурс].

URL:http://intsys.msu.ru/magazine/archive/v8(1-4)/strogalov-005-044.pdf

[6] Проектирование компьютерных обучающих программ по социально-гуманитарным дисциплинам [Электронный ресурс].

URL:http://old.kpfu.ru/gum_konf/ot2r.htm

[7] Методология создания обучающих компьютерных программ в высшем медицинском образовании [Электронный ресурс].

URL:http://www.ssmu.ru/bull/02/2/794.pdf

[8] Руководство по программированию на C# [Электронный ресурс].

URL:http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx

[9] Теория и технологии компьютерного обучения и тестирования [Электронный ресурс]. URL:http://ito.osu.ru/files/monograf.pdf

10) Доржиев Ц.Ц. Разработка и методические рекомендации по применению автоматизированной обучающей системы по начертательной геометрии в учебном процессе. - Улан-Удэ: Изд-во ВСГТУ, 2004. - 72 с.