Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационная безопасность»

К защите

Руководитель проекта

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ**

по дисциплине

«Модели безопасности компьютерных систем»

по теме

«Разработка модели безопасности HRU»

Выполнил студент группы 342

Маркина Полина Павловна дата сдачи на проверку, подпись

Руководитель

Фомина Ксения Юрьевна оценка дата защиты, подпись

Рязань 2016

**Содержание**

[**1. Введение** 4](#_Toc470826641)

[**2. Техническое задание** 5](#_Toc470826642)

[**2.1. Основания для разработки программ** 5](#_Toc470826643)

[**2.2. Назначение разработки** 5](#_Toc470826644)

[**2.3. Требования к программе** 5](#_Toc470826645)

[**2.4. Требования к надёжности** 5](#_Toc470826646)

[**2.5. Требования к программной документации** 5](#_Toc470826647)

[**2.6. Тексты заданий по варианту** 5](#_Toc470826648)

[**3. Описание разработанной программы** 6](#_Toc470826649)

[**3.1. Общие сведения** 6](#_Toc470826650)

[**3.2. Функциональное назначение** 6](#_Toc470826651)

[**3.3. Описание логической структуры** 6](#_Toc470826652)

[**3.3.1. Условности в построении модели** 6](#_Toc470826653)

[**3.3.2. Описание команд** 8](#_Toc470826654)

[**3.3.3. Алгоритм проверки безопасности** 10](#_Toc470826655)

[**3.3.4. Выявление небезопасного состояния** 12](#_Toc470826656)

[**3.4. Описание работы программы с её интерфейсом** 13](#_Toc470826657)

[**3.5. Используемые технические средства** 15](#_Toc470826658)

[**3.6. Вызов и загрузка** 15](#_Toc470826659)

[**3.7. Входные данные** 15](#_Toc470826660)

[**3.8. Выходные данные** 15](#_Toc470826661)

[**4. Программа и методика испытаний** 16](#_Toc470826662)

[**4.1. Объект испытаний** 16](#_Toc470826663)

[**4.2. Цель испытаний** 16](#_Toc470826664)

[**4.3. Требования к программе** 16](#_Toc470826665)

[**4.4. Требования к программной документации** 16](#_Toc470826666)

[**4.5. Средства и порядок испытаний** 16](#_Toc470826667)

[**4.6. Методы испытаний** 17](#_Toc470826668)

[**5. Эксплуатационные документы** 19](#_Toc470826669)

[**5.1. Руководство программиста** 19](#_Toc470826670)

[**5.1.1. Общие сведения о программе** 19](#_Toc470826671)

[**5.1.2. Структура программы** 19](#_Toc470826672)

[**5.1.3. Настройка программы** 19](#_Toc470826673)

[**5.1.4. Проверка программы** 19](#_Toc470826674)

[**5.1.5. Дополнительные возможности программы** 19](#_Toc470826675)

[**5.1.6. Сообщение системному программисту** 19](#_Toc470826676)

[**5.2. Руководство оператора (пользователя)** 21](#_Toc470826677)

[**5.2.1. Назначение и условия применения программы** 21](#_Toc470826678)

[**5.2.2. Обращение к программе для запуска** 21](#_Toc470826679)

[**5.2.3. Входные и выходные данные** 21](#_Toc470826680)

[**5.2.4. Сообщения оператору** 21](#_Toc470826681)

[**7. Приложение** 23](#_Toc470826682)

[**8.** **Список литературы** 31](#_Toc470826683)

1. **Введение**

Необходимо разработать программу, представляющую модель безопасности Харрисона-Руззо-Ульмана(HRU) для анализа начального состояния системы защиты, реализующую дискреционную политику безопасности.

1. **Техническое задание**
   1. **Основания для разработки программ**

Основанием для разработки является задание для курсовой работы по дисциплине «Модели безопасности компьютерных систем».

* 1. **Назначение разработки**

Программа предназначена для моделирования работы модели безопасности HRU.

Программа используется на компьютерах, оснащённых операционной системой Windows 7/8/10.

* 1. **Требования к программе**

Программа должна задавать дискреционную модель разграничения доступа, иметь возможность выполнения команд и автоматически их анализировать на предмет безопасности (утечки прав).

* 1. **Требования к надёжности**

Программа должна исправно функционировать при вводе текущих доступов в системе и исполняемых команд, а также проверки безопасности команд.

* 1. **Требования к программной документации**

Документация к программе представлена следующими разделами.

1. Введение.

2. Техническое задание. Раздел оформлен по ГОСТ 19.201-78.

3. Описание разработанной программы. Раздел оформлен по ГОСТ 19.402-78.

4. Программа и методика испытаний. Раздел оформлен по ГОСТ 19.301-79.

5. Эксплуатационные документы. Раздел оформлен по ГОСТ 19.504-79 и   
ГОСТ 19.503-79.

6. Приложение.

7. Список литературы.

* 1. **Тексты заданий по варианту**

Разработать программу, представляющую модель безопасности Харрисона-Руззо-Ульмана(HRU) для анализа начального состояния системы защиты, реализующую дискреционную политику безопасности.

1. **Описание разработанной программы**
   1. **Общие сведения**

Данная программа реализует модель безопасности Харрисона-Руззо-Ульмана(HRU) для анализа системы защиты.

Программа разработана в среде программирования Windows Microsoft Visual Studio 2013 . Для функционирования программы требуется операционная система Windows 7/8/10.

* 1. **Функциональное назначение**

Программа выполняет проверку вводимых команд на предмет их безопасности для системы.

* 1. **Описание логической структуры**

Дискреционная политика безопасности – политика безопасности, основанная на дискреционном управлении доступом, которое определяется двумя свойствами: все субъекты и объекты идентифицированы; права доступа субъектов на объекты системы определяются на основании некоторого внешнего по отношению к системе правила. Основным элементом систем дискреционного разграничения доступа является матрица доступов.

Модель HRU используется для анализа систем защиты, реализующих дискреционную политику безопасности.

* + 1. **Условности в построении модели**

В рамках программной реализации такой модели примем следующие необходимые условности.

Элементами модели будут:

- S – множество возможных субъектов,

- O – множество возможных объектов (напомним, что S ⊂ O);

- R={r,w,rw,wr, α } – конечное множество прав доступа

- M – матрица прав доступа, описывающая текущие права доступа субъектов к

объектам;

Функционирование системы рассматривается только с точки зрения изменений в матрице доступов.

Поведение системы во времени моделируется переходами между различными её состояниями. Переходы осуществляются путём внесения изменений в матрицу М с использованием команд следующего вида:

command ( ) 1,..., k α x x

if r1 in M[xs1, xo1] and

r2 in M[xs2, xo2] and

…

rm in M[xsm, xom]

then

op1,

op2,

…

opn,

end

Здесь α - имя команды; xi – параметры команды, представляющие собой идентификаторы субъектов и объектов, opi – элементарные операции.

Элементарные операции op1…opn будут выполнены в том случае, если выполняются все без исключения условия из блока if … then.

При описании элементарных операций мы будем полагать, что в результате выполнения операции система переходит из состояния Q=(S, O, M) в состояние Q’=(S’, O’, M’).

Начальное состояние q0 считается безопасным по отношению к праву , если невозможен переход в такое состояние q, в котором возможна утечка права r.

* + 1. **Описание команд**

1. Внести право r в M[s,o] ( s∈S,o∈O ) – добавление субъекту s права r по отношению к объекту o. В результате выполнения команды происходят следующие изменения в состоянии системы:

- S’=S,

- O’=O,

- M’[xs, xo]=M[xs, x0], если (xs, xo) ≠ (s,o),

- M’[s, o]=M[s, o] ∪ {r}.

Заметим, что содержимое ячейки таблицы рассматривается как множество. Это, в частности, означает, что если добавляемый элемент уже присутствовал в ячейке, то её содержимое не изменяется.

1. Удалить право r из M[s,o] ( s∈S,o∈O ) – удаление у субъекта s права r по отношению к объекту o. Изменения в состоянии системы:

- S’=S,

- O’=O,

- M’[xs, xo]=M[xs, x0], если (xs, xo) ≠ (s,o),

- M’[s,o]=M[s.o] \ {r}.

Если удаляемое право отсутствовало в ячейке, то состояние системы в результате выполнения данной команды никак не изменяется.

1. Создать субъект s (s∉S ) – создание нового субъекта s. Изменения в состоянии системы:

- O’=O∪ {s},

- S’=S ∪ {s},

- M’[xs, xo]=M[xs, xo] для ∀ (xs, xo)∈S×O,

- M’[s, xo]=Ø для ∀ x O' o ∈

- M’[s, xs]=Ø для ∀ x S' s ∈

При создании субъекта в матрицу M добавляются строка и столбец.

4. Уничтожить субъект s (s∈S) – удаление существующего субъекта s.

Изменения в состоянии системы:

- S’=S \ {s},

- O’=O \ {s},

- M’[xs, xo]=M[xs, xo] для ∀ (xs, xo)∈S’×O’.

5. Создать объект o (o∉O) – создание нового объекта o.

Изменения в состоянии системы:

- O’=O∪ {o},

- S’=S,

- M’[xs, xo]=M[xs, xo] для ∀ (xs, xo)∈S×O,

- M’[xs, xo]= Ø для ∀ x S' s ∈

При добавлении объекта в матрице доступа создаётся новый столбец.

6. Уничтожить объект o (o∈O \ S) – удаление существующего объекта o.

Изменения в состоянии системы:

- O’=O \ {o},

- S’=S,

- M’[xs, xo]=M[xs, xo] для ∀ (xs, xo)∈S’×O’.

* + 1. **Алгоритм проверки безопасности**

Утечка права (получение права) может произойти только в следующем случае выполнения команды передачи права доступа.

1. Система находится в начальном состоянии.
2. Первая команда Q0 создает 3 субъекта с названиями S1,S2,S3,S4

Q0:CS[s1];CS[s2];CS[s3]; CS[s4];

Рисунок 1 – Команда, включающая 3 примитивных оператора «Создать субъект»,CS[S]- примитивный оператор create subject.

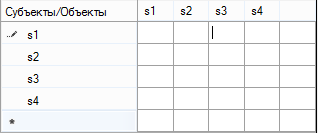


Рисунок 2 – Состояние системы после выполнения команды Q0.

1. Вторая команда Q1 вносит право w субъекта S1 на субъект S2

Q1:ER[w;s1;s3];

Рисунок 3 – Команда, включающая один примитивный оператор «Внести право»,ER [w;s1,s3]- примитивный оператор enter rule.

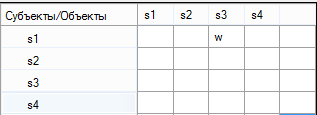


Рисунок 4 – Состояние системы после выполнения команды Q1.

4)Третья команда Q2 вносит право субъекта S1 на субъект S4

Q2:ER[a;s1;s4];

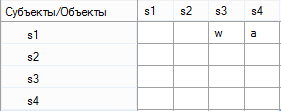


Рисунок 5 – Состояние системы после выполнения команды Q2.

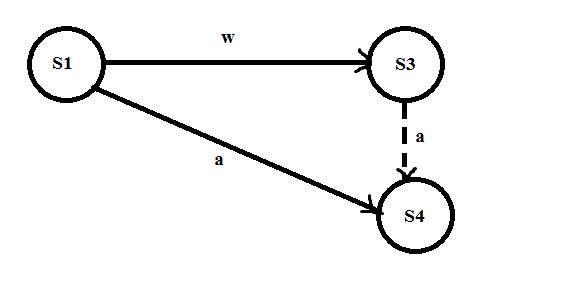


Рисунок 5 –Граф состояния системы.

Построив граф, можно увидеть, что субъект S3 через право w может получить право, а на субъект S4.

Вывод:конечное состояние системы не безопасно, возможна утечка права а .

* + 1. **Выявление небезопасного состояния**

Согласно разделу 3.3.1, начальное состояние q0 считается безопасным по отношению к праву , если невозможен переход в такое состояние q, в котором возможна утечка права r.

Утечка может произойти при передаче права доступа.

Команда разрешает передачу произвольного права доступа *x* от любого субъекта *p* любому субъекту, по отношению к которому исходный субъект *s* обладает правом записи.

command Передатьx(s, o, p)

если w в M[s, o] и

x в M[s, p]

тогда

Внести x в M[o, p],

End

Такая команда пройдет проверку по следующим условиям:

Если выполняется команда «Получить», то состояние небезопасно.

Команда «Получить» разрешает получение произвольного права доступа *x* от любого субъекта *o*, по отношению к которому исходный субъект *s* имеет право чтения.

command Получитьx(s, o, p)

если r в M[s, o] и

x в M[o, p]

тогда

внести x в M[s, p],

end.

* 1. **Описание работы программы с её интерфейсом**

Далее представлено главное окно программы.

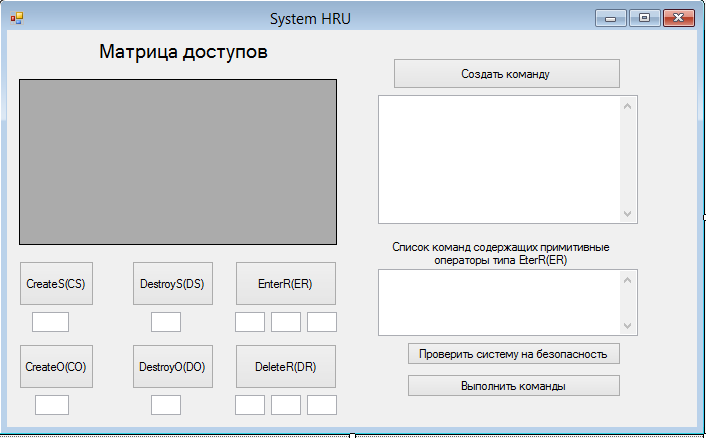


Рисунок 6– Главное окно программы.

1. Поле матрицы доступов предназначено для просмотра, текущего или исходного состояния системы, в зависимости от выбора пользователя.

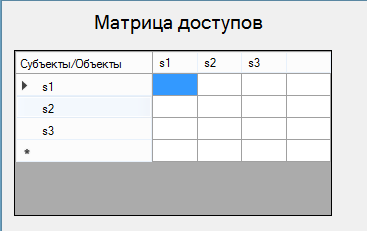


Рисунок 7 – Поле матрицы доступов.

2)Поля и кнопки для ввода команд позволяют пользователю вводить необходимые команды для работы с системой.

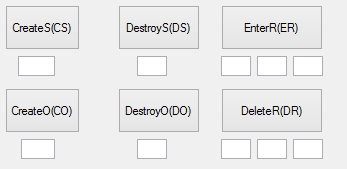


Рисунок 8 – Поля и кнопки ввода команд

3)Окно просмотра введенных команд. Позволяет узнать, какие команды уже выполнены системой.

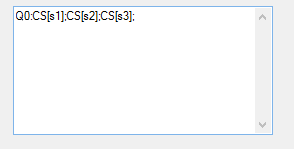


Рисунок 9 – Окно просмотра команд

4)Окно просмотра введенных команд, содержащих примитивные операторы «внести право».

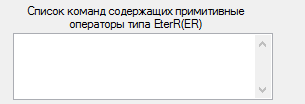


Рисунок 10 – Окно просмотра команд, содержащих примитивные операторы «внести право».

5)Кнопка, предназначенная для создания команды, включающая примитивные операторы.



Рисунок 11 – Кнопка создания команды.

6)Кнопки, предназначенные для выполнения заданных команд и проверки системы на безопасность.

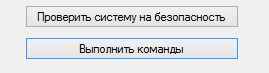


Рисунок 12 – Кнопки проверки на безопасность и выполнения команд.

* 1. **Используемые технические средства**

Для работы программы необходим персональный компьютер с установленной операционной системой Windows 7/8/10, монитор, клавиатура и мышь.

* 1. **Вызов и загрузка**

Программа может быть загружена запуском для исполнения файла «Курсовой.exe», находящегося в том месте, куда он был скопирован.

* 1. **Входные данные**

При работе, в качестве входных данных выступают:

1. матрица текущих доступов в системе, которая может быть получена способами, описанными в разделе 3.4;
2. список команд для исполнения, который может быть получен способами, описанными в разделе 3.4.
   1. **Выходные данные**

В качестве выходных данных выступают:

1. сообщения, характеризующие начальное состояние системы как безопасное или небезопасное.
2. **Программа и методика испытаний**
   1. **Объект испытаний**

Объектом испытаний является программа, представляющую модель безопасности Харрисона-Руззо-Ульмана(HRU) для анализа систем защиты, реализующих дискреционную политику безопасности.

* 1. **Цель испытаний**

Целью испытаний является выявление возможных функциональных ошибок и сбоев в программе при выполнении процедур анализа на предмет безопасности начального состояния системы.

* 1. **Требования к программе**

Программа должна реализовывать заданную модель безопасности, а именно в случае утечки прав доступа указать начальное состояние как небезопасное.

* 1. **Требования к программной документации**

В программной документации должны присутствовать следующие разделы:

1. Введение;
2. Техническое задание;
3. Описание разработанной программы;
4. Программа и методика испытаний;
5. Эксплуатационные документы;
6. Приложение (листинг программы);
7. Список литературы.
   1. **Средства и порядок испытаний**

В процессе проведения испытаний использовалась конфигурация аппаратных средств, соответствующая разделу 3.5. На момент проведения испытаний использовалась следующая конфигурация аппаратных средств:

1. ПК с процессором Intel(R) Core(TM) i5-3340M CPU @ 3.1 GHz;
2. объем ОЗУ – 8 Гбайт;
3. видеоадаптер NVidia GeForce GTX 760
4. мышь.

Из программных средств используется 64-разрядная операционная система   
Windows 8, полностью совместимая с указанной конфигурацией, а также система программирования Windows Microsoft Visual Studio 2013.

Порядок испытаний:

1. проверить правильность работы алгоритма проверки безопасности на контрольных примерах;
2. дополнительно проверить работоспособность элементов окон программы.
   1. **Методы испытаний**

Испытание правильности функционирования модели произведём с помощью контрольных примеров команд и матриц доступов. Результаты тестирований приведены в таблице 1.

Таблица 1 – Результаты тестирования на контрольных наборах

|  |  |  |  |
| --- | --- | --- | --- |
| Исходная матрица доступов | Команда | Ожидаемый результат | Результат |
|  | Создать субъект s4 | Создастся субъект s4 |  |
|  | Создать объект o1 | Создастся субъект o1 |  |
|  | Внести право r субъекту s1 на cсубъект s2 | Право удачно внесено. |  |
|  | Внести право a субъекту s1 на объект o1. | Право будет успешно внесено, но произойдет утечка права. |  |
|  | Выполнить проверку безопасности | Утечка права a |  |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Исходная матрица доступов | Команда | Ожидаемый результат | Результат |
|  | Уничтожить субъект s2 | Субъект s2 будет удалён |  |
|  | Удалить право a у субъекта s1 на объект o1. | Право успешно удалено |  |

С большой долей вероятности можно утверждать, что алгоритм работы автомата верно обрабатывает поступающие команды в систему. Несоответствий между ожиданием и итоговыми результатами не наблюдается.

Дополнительно проведено тестирование визуальных элементов окна приложения. Все действия, выполняемые этими элементами, отвечают их назначению.

1. **Эксплуатационные документы**
   1. **Руководство программиста**
      1. **Общие сведения о программе**

Программа предназначена для моделирования работы модели безопасности Харрисона-Руззо-Ульмана(HRU) для анализа систем защиты, реализующих дискреционную политику безопасности.

Программа позволяет пользователю вводить данные через программный интерфейс.

Программа проста в использовании и рассчитана на средний уровень квалификации пользователя.

Программа разработана в среде программирования Delphi 10 Seattle.

Программа используется на компьютерах, оснащённых операционной системой Windows 7/8/10.

Программа так же может быть использована как демонстрационная модель.

* + 1. **Структура программы**

Рабочая версия программы представляет собой файл «Курсовой.exe».

* + 1. **Настройка программы**

Программы не нуждаются в какой-либо дополнительной настройке или обслуживании. Для корректной работы желательно запускать её от имени администратора.

* + 1. **Проверка программы**

Результаты тестирования программы приведены в разделе 4.6. При проведении дополнительных тестирований следует опираться на результаты этого раздела.

* + 1. **Дополнительные возможности программы**

Незаявленных возможностей программа не имеет.

* + 1. **Сообщение системному программисту**

В ходе работы могут возникнуть следущие сообщения напротив выполняемых команд.

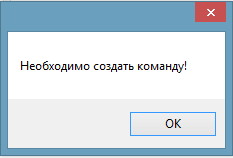


Рисунок 13 – Сообщение при создании субъекта\объекта, данные примитивные операторы не могут быть выполнены, так как не задана команда.

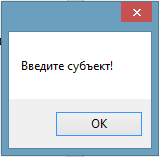


Рисунок 14 – Сообщение об отсутствии имени создаваемого субъекта в поле ввода(аналогичные сообщения появляются всегда, когда одно из обязательных полей не заполнено)

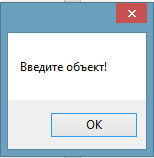


Рисунок 15 – Сообщение об отсутствии имени создаваемого объекта в поле ввода(аналогичные сообщения появляются всегда, когда одно из обязательных полей не заполнено)

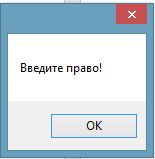


Рисунок 14 – Сообщение об отсутствии имени задаваемого права.

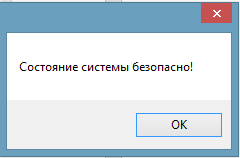


Рисунок 15 – Сообщение может появиться при проверке безопасности.

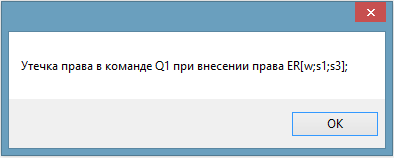


Рисунок 16-Сообщение может появиться при проверки безопасности, если произошла утечка права

* 1. **Руководство оператора (пользователя)**
     1. **Назначение и условия применения программы**

Программа предназначена для моделирования работы модели безопасности Харрисона-Руззо-Ульмана(HRU) для анализа систем защиты, реализующих дискреционную политику безопасности.

Программа используется на компьютерах, оснащённых операционной системой Windows 7/8/10.

* + 1. **Обращение к программе для запуска**

Программа может быть загружена запуском для исполнения файла «Курсовой.exe», находящегося в том месте, куда он был скопирован.

* + 1. **Входные и выходные данные**

При работе, в качестве входных данных выступают:

1. матрица текущих доступов в системе, которая может быть получена способами, описанными в разделе 3.4;
2. список команд для исполнения, который может быть получен способами, описанными в разделе 3.4.

В качестве выходных данных выступают:

1. сообщения, характеризующие начальное состояние системы как безопасное или небезопасное.
   * 1. **Сообщения оператору**
2. В ходе работы могут возникнуть следущие сообщения напротив выполняемых команд.

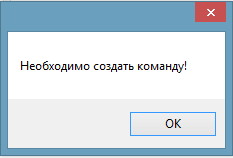


Рисунок 17 – Сообщение при создании субъекта\объекта, данные примитивные операторы не могут быть выполнены, так как не задана команда.

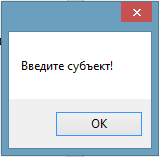


Рисунок 18 – Сообщение об отсутствии имени создаваемого субъекта в поле ввода(аналогичные сообщения появляются всегда, когда одно из обязательных полей не заполнено)

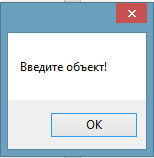


Рисунок 19 – Сообщение об отсутствии имени создаваемого объекта в поле ввода(аналогичные сообщения появляются всегда, когда одно из обязательных полей не заполнено)

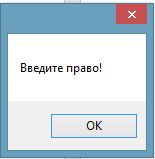


Рисунок 20 – Сообщение об отсутствии имени задаваемого права.

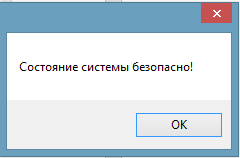


Рисунок 21 – Сообщение может появиться при проверке безопасности.

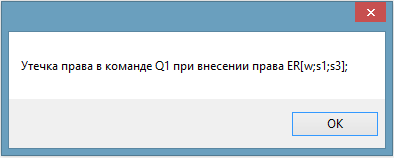


Рисунок 22-Сообщение может появиться при проверки безопасности, если произошла утечка права

1. **Приложение**
2. using System;
3. using System.Collections.Generic;
4. using System.ComponentModel;
5. using System.Data;
6. using System.Drawing;
7. using System.Linq;
8. using System.Text;
9. using System.Threading.Tasks;
10. using System.Windows.Forms;
11. namespace WindowsFormsApplication1
12. {
13. public partial class Form1 : Form
14. {
15. int rownum= 1;
16. int columnnum = 1;
17. int countQ=0;
18. string pust = null;
19. bool flag = true; //флаг колличества команд
20. bool flag2 = false;
21. string yQ;
22. string[] pravo = new string[10] ;
23. int z = 0;
24. bool ytechka = false;
25. Dictionary<int,string> subjects = new Dictionary<int,string>(); // ключ номер строки таблицы
26. Dictionary<int, string> objects = new Dictionary<int, string>(); // ключ номер столбца таблицы
27. public Form1()
28. {
29. InitializeComponent();
30. }
31. private void Form1\_Load(object sender, EventArgs e)
32. {
34. dataGridView1.Rows.Clear();
35. dataGridView1.Columns.Clear();
36. dataGridView1.TopLeftHeaderCell.Value = "Субъекты/Объекты";
37. dataGridView1.Visible =false ;
38. }
39. private void button1\_Click(object sender, EventArgs e)
40. {
41. if(textBox1.Text=="")
42. {
43. MessageBox.Show ("Введите субъект!");
44. return;
45. }
46. if (flag2 == false)
47. {
48. MessageBox.Show("Необходимо создать команду!");
49. return;
50. }
51. subjects.Add(rownum,textBox1.Text);
52. dataGridView1.RowCount = rownum + 1;
53. dataGridView1.Rows[rownum-1].HeaderCell.Value = textBox1.Text;
54. rownum = rownum + 1;
55. textBox9.AppendText("CS[" + textBox1.Text + "];");
56. objects.Add(columnnum, textBox1.Text);
57. dataGridView1.ColumnCount = columnnum + 1;
58. dataGridView1.Columns[columnnum - 1].HeaderText = textBox1.Text;
59. columnnum = columnnum + 1;
60. dataGridView1.RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders;
61. }
62. private void button4\_Click(object sender, EventArgs e)
63. {
64. if (textBox2.Text == "")
65. {
66. MessageBox.Show("Введите объект!");
67. return;
68. }
69. if (flag2 == false)
70. {
71. MessageBox.Show("Необходимо создать команду!");
72. return;
73. }
74. objects.Add(columnnum,textBox2.Text);
75. dataGridView1.ColumnCount = columnnum + 1;
76. dataGridView1.Columns[columnnum - 1].HeaderText = textBox2.Text;
77. columnnum = columnnum + 1;
78. dataGridView1.RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders;
79. textBox9.AppendText("CO[" + textBox2.Text + "];");
80. }
81. private void button8\_Click(object sender, EventArgs e)
82. {
83. if (countQ == 0)
84. {
85. MessageBox.Show("Не задано ни одной команды!");
86. return;
87. }
88. dataGridView1.Visible = true;
89. }
90. private void button7\_Click(object sender, EventArgs e)
91. {
92. flag2 = true;
93. if (rownum==1) {
94. textBox9.Text = textBox9.Text + 'Q' + countQ+':';
95. countQ = countQ + 1;
96. }
97. else
98. {
99. textBox9.Text = textBox9.Text + Environment.NewLine;
100. textBox9.Text = textBox9.Text + 'Q' + countQ + ':';
101. countQ = countQ + 1;
102. }
103. flag = true;
104. }
105. private void button3\_Click(object sender, EventArgs e)
106. {
107. if (textBox11.Text == "")
108. {
109. MessageBox.Show("Введите право!");
110. return;
111. }
112. if (textBox6.Text == "")
113. {
114. MessageBox.Show("Введите субъект!");
115. return;
116. }
117. if (textBox5.Text == "")
118. {
119. MessageBox.Show("Введите объект!");
120. return;
121. }
122. int cl = 0, rw = 0;
123. string rule = null;
124. if (flag2 == false)
125. {
126. MessageBox.Show("Необходимо создать команду!");
127. return;
128. }
129. if (flag == true)
130. {
131. textBox10.AppendText('Q' + (countQ-1).ToString() + ";");
132. flag = false;
133. }
134. ICollection<int> okeys = objects.Keys;
135. ICollection<int> skeys = subjects.Keys;
136. foreach (int j in skeys)
137. {
138. if (subjects[j] == textBox6.Text)
139. {
140. rw = j - 1;
141. break;
142. }
143. }
144. foreach (int j in okeys)
145. {
146. if (objects[j] == textBox5.Text)
147. {
148. cl = j - 1;
149. break;
150. }
151. }
152. rule = ' ' + rule + dataGridView1[cl, rw].Value + textBox11.Text + ' ';
153. dataGridView1[cl, rw].Value = rule;
154. textBox9.AppendText("ER[" + textBox11.Text + ';' + textBox6.Text + ';' + textBox5.Text + "];");
155. if (ytechka == false)
156. {
157. int colcount, rowcount;
158. colcount = Int32.Parse(dataGridView1.ColumnCount.ToString());
159. rowcount = Int32.Parse(dataGridView1.RowCount.ToString());
160. for (int i = 0; i <= rowcount - 1; i++)
161. {
162. for (int j = 0; j <= colcount - 1; j++)
163. {
164. if (dataGridView1[j, i].Value != null)
165. {
166. string prov = dataGridView1[j, i].Value.ToString();
167. if (prov == " w " || prov == " r " || prov == " w r " || prov == " r w ")
168. {
169. for (int m = 0; m < colcount - 1; m++)
170. {
171. if (dataGridView1[m, j].Value != null)
172. {
173. if (dataGridView1[m, i].Value == null || dataGridView1[m, j].Value.ToString() != dataGridView1[m, i].Value.ToString()) //объекты не сравнивают
174. {
176. yQ = countQ.ToString();
177. pravo[z]="ER[" + textBox11.Text + ';' + textBox6.Text + ';' + textBox5.Text + "];";
178. ytechka = true;
179. return;
181. }
182. }
183. }
184. for (int m = 0; m < colcount - 1; m++)
185. {
186. if (dataGridView1[m, i].Value != null & m != j)
187. {
188. if (dataGridView1[m, j].Value == null || dataGridView1[m, j].Value.ToString() != dataGridView1[m, i].Value.ToString()) //объекты не сравнивают
189. {
191. yQ = countQ.ToString();
192. pravo[z]="ER[" + textBox11.Text + ';' + textBox6.Text + ';' + textBox5.Text + "];";
193. ytechka = true;
194. return;
196. }
197. }
198. }
199. }
200. }
201. }
202. }
203. }
204. }
205. private void button6\_Click(object sender, EventArgs e)
206. {
207. if (textBox12.Text == "")
208. {
209. MessageBox.Show("Введите право!");
210. return;
211. }
212. if (textBox7.Text == "")
213. {
214. MessageBox.Show("Введите субъект!");
215. return;
216. }
217. if (textBox8.Text == "")
218. {
219. MessageBox.Show("Введите объект!");
220. return;
221. }
222. string value= null;
223. if (flag2 == false)
224. {
225. MessageBox.Show("Необходимо создать команду!");
226. return;
227. }

230. int cl = 0, rw = 0;
232. ICollection<int> okeys = objects.Keys;
233. ICollection<int> skeys = subjects.Keys;
234. foreach (int j in skeys)
235. {
236. if (subjects[j] == textBox7.Text)
237. {
238. rw = j-1;
239. break;
240. }
241. }
242. foreach (int j in okeys)
243. {
244. if (objects[j] == textBox8.Text)
245. {
246. cl = j-1;
247. break;
248. }
249. }
250. if (dataGridView1[cl, rw].Value != null)
251. {
252. value = value + dataGridView1[cl, rw].Value;
253. dataGridView1[cl, rw].Value = pust;
254. value = value.Remove(value.IndexOf(textBox12.Text), 1);
255. dataGridView1[cl, rw].Value = value;
256. textBox9.AppendText("DR[" + textBox12.Text + ';' + textBox7.Text + ';' + textBox8.Text + "];");
257. }
258. else
259. {
260. MessageBox.Show("Заданная ячейка пуста!");
261. }
262. }
263. private void button9\_Click(object sender, EventArgs e)
264. {
265. if (Convert.ToInt32(yQ) == 0)
266. {
267. MessageBox.Show("Состояние системы безопасно!");
268. return;
269. }
270. textBox10.Text = textBox10.Text.Substring(textBox10.Text.IndexOf(yQ) + 2);
271. MessageBox.Show("Утечка права в команде Q" +(Convert.ToInt32(yQ)-1).ToString() + " при внесении права " + pravo[z]);
272. return;
274. }
275. private void button5\_Click(object sender, EventArgs e)
276. {
277. if (textBox4.Text == "")
278. {
279. MessageBox.Show("Введите объект!");
280. return;
281. }
282. if (flag2 == false)
283. {
284. MessageBox.Show("Необходимо создать команду!");
285. return;
286. }
287. int delcol = 0, colcount;
288. bool nashel = false;
289. if (flag2 == false)
290. {
291. MessageBox.Show("Необходимо создать команду!");
292. return;
293. }
294. colcount = Int32.Parse(dataGridView1.ColumnCount.ToString());
295. ICollection<int> okeys = objects.Keys;
296. foreach (int j in okeys)
297. {
298. if (objects[j] == textBox4.Text)
299. {
300. delcol = j - 1;
301. nashel = true;
302. break;
303. }
304. }
305. if (nashel == false)
306. {
307. MessageBox.Show("Введенный объект " + textBox4.Text + " не существует !");
308. return;
309. }
310. dataGridView1.Columns.RemoveAt(delcol);
311. delcol = delcol + 1;
312. objects.Remove(delcol);
313. for (int m = 1; m < colcount; m++)
314. {
315. if (m == delcol + 1)
316. {
317. string value = objects[m];
318. objects.Remove(m);
319. objects.Add(m - 1, value);
320. delcol = delcol + 1;
321. }
322. }
323. columnnum = columnnum - 1;
324. }
325. private void button2\_Click(object sender, EventArgs e)
326. {
327. if (textBox3.Text == "")
328. {
329. MessageBox.Show("Введите субъект!");
330. return;
331. }
332. int delrow=0 ,rowcount;
333. bool nashel = false;
334. if (flag2 == false)
335. {
336. MessageBox.Show("Необходимо создать команду!");
337. return;
338. }
339. rowcount = Int32.Parse(dataGridView1.RowCount.ToString());
340. ICollection<int> skeys = subjects.Keys;
341. foreach (int j in skeys)
342. {
343. if (subjects[j] == textBox3.Text)
344. {
345. delrow = j-1;
346. nashel = true;
347. break;
348. }
349. }
350. if (nashel == false)
351. {
352. MessageBox.Show("Введенный субъект " + textBox3.Text + " не существует !");
353. return;
354. }
355. dataGridView1.Rows.RemoveAt(delrow);
356. delrow=delrow+1;
357. subjects.Remove(delrow);
358. for (int m = 1; m < rowcount; m++)
359. {
360. if (m == delrow + 1)
361. {
362. string value = subjects[m];
363. subjects.Remove(m);
364. subjects.Add(m - 1, value);
365. delrow = delrow + 1;
367. }
368. }
369. rownum = rownum - 1;
370. int delcol = 0, colcount;
372. colcount = Int32.Parse(dataGridView1.ColumnCount.ToString());
373. ICollection<int> okeys = objects.Keys;
374. foreach (int j in okeys)
375. {
376. if (objects[j] == textBox3.Text)
377. {
378. delcol = j - 1;
379. break;
380. }
381. }
382. dataGridView1.Columns.RemoveAt(delcol);
383. delcol = delcol + 1;
384. objects.Remove(delcol);
385. for (int m = 1; m < colcount; m++)
386. {
387. if (m == delcol + 1)
388. {
389. string value = objects[m];
390. objects.Remove(m);
391. objects.Add(m - 1, value);
392. delcol = delcol + 1;
393. }
394. }
395. columnnum = columnnum - 1;
397. }
398. }
399. }
400. **Список литературы**
401. Модели безопасности компьютерных систем: курс лекций/ Рязанский государственный радиотехнический университет; составитель – Фомина К.Ю.
402. Девянин П.Н. «Модели безопасности компьютерных систем: Учебное пособия для студентов высших учебных заведений» / Издательский центр «Академия» 2005.
403. Цирилов В.Л. «Основы информационной безопасности автоматизированных систем» / Издательство «Феникс» 2008.