Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАИП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**ИГРОВОЕ ПРОГРАММНОЕ СРЕДСТВО “БУНКЕР”**

БГУИР КП 6-05-0612-01 029 ПЗ

Студент гр. 451004 Редько А.М.

Руководитель Данилова Г.В.

Минск 2025

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc134529284)

[1 Анализ предметной области 7](#_Toc134529285)

[1.1 Обзор аналогов 7](#_Toc134529286)

[1.2 Постановка задачи 9](#_Toc134529287)

[2 Проектирование программного средства 10](#_Toc134529288)

[2.1 Структура программы 10](#_Toc134529289)

[2.2 Проектирование интерфейса программного средства 10](#_Toc134529290)

[2.3 Проектирование функционала программного средства 11](#_Toc134529291)

[3 Разработка программного средства 16](#_Toc134529292)

[3.1 Структура заданий 16](#_Toc134529293)

[3.2 Создание структуры хранения ответов 17](#_Toc134529294)

[3.3 Сбор результатов 18](#_Toc134529295)

[4 Тестирование программного средства 21](#_Toc134529296)

[5 Руководство пользователя 22](#_Toc134529297)

[5.1 Интерфейс программного средства 22](#_Toc134529298)

[5.2 Управление программным средством 25](#_Toc134529299)

[Заключение 26](#_Toc134529300)

[Список использованных источников 27](#_Toc134529301)

[Приложение А. Исходный код 28](#_Toc134529302)

# ВВЕДЕНИЕ

С момента появления первых компьютерных игр разработчики стремились создавать не только развлекательные, но и интеллектуально насыщенные продукты, которые бы стимулировали мышление, логику и стратегическое планирование. История игровой индустрии богата примерами простых, но увлекательных игр, которые стали классикой благодаря своей доступности и уникальным механикам. Такие игры, как “Tetris”, “Pac-man”, “Змейка” и другие, не только развлекали пользователей, но и способствовали формированию определенного типа мышления, связанного с решением задач и анализом ситуаций. В этом контексте особое место занимают настольные игры, которые, будучи перенесенными в цифровой вид, предлагают новые возможности.

Одной из таких игр является “Бункер” – настольная игра, которая сочетает в себе элементы стратегии, логики и социального взаимодействия. Игра “Бункер” представляет собой дискуссионную карточную игру с постапокалиптическим сюжетом, где игрокам необходимо принимать решения, основываясь на своих ролях, характеристиках и ограниченных ресурсах. Уникальность игры заключается в ее социальной составляющей: игроки должны не только анализировать ситуацию, но и взаимодействовать друг с другом, договариваться, убеждать и даже обманывать, чтобы достичь своей цели. Эта игра стала популярной благодаря своей простоте, глубине и возможности играть как в небольшой компании, так и в больших группах.

Цифровая версия игры “Бункер” открывает новые горизонты для ее развития. Во-первых, она позволяет игрокам со всего мира взаимодействовать друг с другом, не ограничиваясь физическим присутствием. Во-вторых, цифровой формат дает возможность автоматизировать многие процессы, например, распределение ролей, что делает игру более удобной. В-третьих, разработка игры в цифровом формате позволяет экспериментировать с новыми механиками, визуальным оформлением, что может сделать игровой процесс еще более увлекательным.

Целью данного курсового проекта является разработка цифровой версии игры “Бункер”, которая сохранит все ключевые элементы оригинальной настольной игры, но при этом будет адаптирована для современных платформ. В процессе разработки планируется уделить особое внимание пользовательскому интерфейсу, чтобы сделать игру максимально интуитивно-понятной и удобной для игроков. Кроме того, важной задачей является реализация многопользовательского режима, который позволит игрокам взаимодействовать друг с другом в реальном времени.

Актуальность данной работы обусловлена нестихающим интересом к настольным играм в цифровом формате, особенно в условиях, когда удаленное взаимодействие становится все более востребованным. Игра “Бункер” не только предоставляет возможность для развлечения, но и способствует развитию навыков коммуникации, дискуссии и стратегического мышления.

Таким образом, разработка цифровой версии игры «Бункер» представляет собой не только техническую задачу, но и возможность создать продукт, который будет способствовать развитию социальных и когнитивных навыков у игроков. В процессе работы будут рассмотрены основные аспекты разработки игр, включая проектирование игрового процесса, создание пользовательского интерфейса, реализацию многопользовательского режима и тестирование. Результатом проекта станет готовое игровое программное средство, которое сможет стать достойной альтернативой оригинальной настольной игре и привлечь внимание как любителей настольных игр, так и новых пользователей.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **Обзор аналогов**

В настоящее время существует множество цифровых адаптаций настольных игр, каждая из которых имеет свои особенности реализации, преимущества и недостатки. Популярность таких игр объясняется удобством онлайн-взаимодействия, автоматизацией игровых процессов и возможностью играть с людьми по всему миру.

Одним из наиболее распространенных способов переноса настольных игр в цифровой формат является использование платформ для создания многопользовательских игр. Эти платформы предоставляют инструменты для эмуляции физических игровых процессов, включая перемещение карт, броски кубиков и взаимодействие между игроками. Например, в “Tabletop Simulator” игроки могут создавать собственные модификации настольных игр, что делает его универсальным решением для любителей жанра. Однако такие платформы требуют от пользователей определенных технических навыков, а также не всегда обеспечивают удобный интерфейс для конкретных игр, что может усложнять процесс игры.

Данное программное средство представлено на рисунке 1.1.

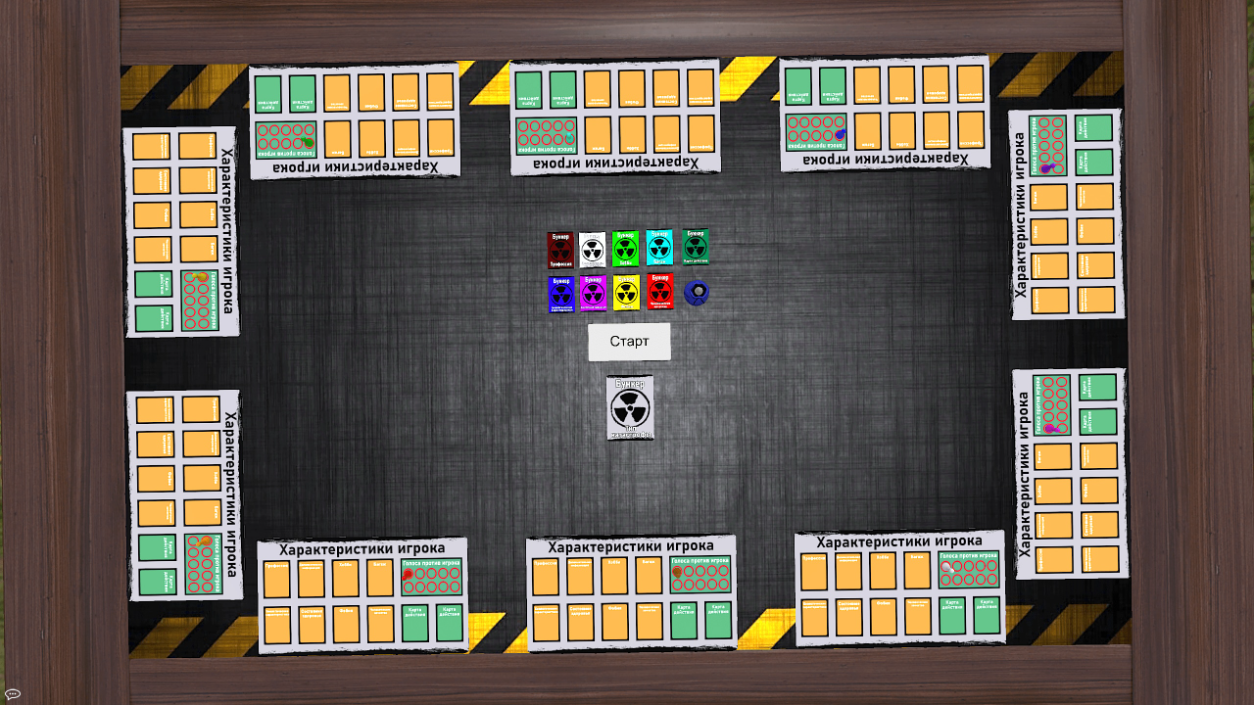


Рисунок 1.1 – Компьютерная игра “Tabletop Simulator”

В настоящее время цифровые адаптации настольных игр активно развиваются, предлагая игрокам новые форматы взаимодействия. Одним из наиболее известных аналогов является игра "Resistance: Avalon", которая, подобно "Бункеру", строится на механике социальной дедукции и коллективного принятия решений. Эта реализация демонстрирует удачный пример адаптации настольной игры в цифровой формат, сохраняя при этом ключевые игровые механики.

Данное программное средство представлено на рисунке 1.2.



Рисунок 1.2 – Компьютерная игра “Resistance: Avalon”

Главным преимуществом цифровой версии "Resistance: Avalon" стала автоматизация процессов, которые в настольном варианте требуют участия ведущего. Программа самостоятельно распределяет роли, контролирует соблюдение правил и подсчитывает результаты голосований, что значительно ускоряет игровой процесс. Особого внимания заслуживает реализация сетевого мультиплеера, позволяющая игрокам со всего мира участвовать в совместных партиях через интернет. Встроенные инструменты коммуникации, включая текстовый и голосовой чат, помогают сохранить элемент социального взаимодействия, который является важной составляющей подобных игр.

Главным недостатком существующих аналогов является отсутствие специализированных решений для игры “Бункер”. Большинство платформ либо слишком универсальны, либо не учитывают специфику этой игры, включая ее механики взаимодействия, систему ролей и процесс голосования.

## **Постановка задачи**

В рамках данной курсовой работы будет разработана цифровая версия настольной игры “Бункер”. Основной целью проекта будет сохранение всех ключевых механик оригинальной игры, обеспечивая при этом удобный интерфейс и автоматизацию игровых процессов.

В программном средстве будут реализованы следующие функции:

– создание новой игровой сессии с настройкой количества игроков;

– генерация уникальных характеристик для каждого игрока;

– загрузка и сохранение текущего состояния игры;

– просмотр карт всех участников с их характеристиками;

– пошаговое раскрытие информации о персонажах;

– система исключения игроков;

– обработка решений по ситуациям в бункере;

– автоматический расчет шансов на выживание;

– анимированное отображение результатов игры;

– просмотр правил и подсказок в процессе игры.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 12.0 Community Edition.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **Структура программы**

При разработке приложения будет использовано семь модулей:

* StartUnit – модуль, отвечающий за отображение начального меню приложения.
* RuleUnit – модуль, содержащий полное описание игрового процесса, механик, правил и особенностей игры.
* SeedEnterUnit – модуль, отвечающий за систему генерации игровых сессий через уникальные ключи.
* PreparationUnit – модуль, являющийся настройками игровой сессии.
* MainUnit – модуль основного игрового процесса. Он отвечает за отображение игрового поля, карт персонажей и управление игровым процессом.
* FinalUnit – модуль, отвечающий за расчет и отображение результатов, анимацию победы/поражения.
* ConnectUnit – модуль, выступающий промежуточным звеном между этапом подготовки и основным игровым процессом.

## **Проектирование интерфейса программного средства**

Дизайн программных продуктов является ключевым фактором, влияющим на удобство взаимодействия пользователей и общее качество решения. Для достижения этих целей интерфейс должен соответствовать следующим базовым принципам:

* оптимизация рабочего пространства;
* интуитивная понятность.

**2.2.1** Начальное окно

Начальное окно приложения состоит из четырех компонентов TLabel и двух кнопок компонента TMainMenu, с помощью которых пользователь может создать новую игру, подключиться по ключу игры, ознакомиться с правилами, ознакомиться с разработчиком, загрузить сохраненную игру из файла, либо осуществить выход:

* нажатие на надпись “Новая игра” открывает модуль PreparationUnit, где пользователь начинает настройку игровой сессии;
* нажатие на надпись “Подключиться” открывает модуль SeedEnterUnit, где пользователь может ввести уникальный ключ;
* нажатие на надпись “Правила” отобразит модуль RuleUnit, где пользователь может ознакомиться с правилами;
* нажатие на надпись “Выход” закрывает приложение.

Макет начального окна представлен на рисунке 2.1.

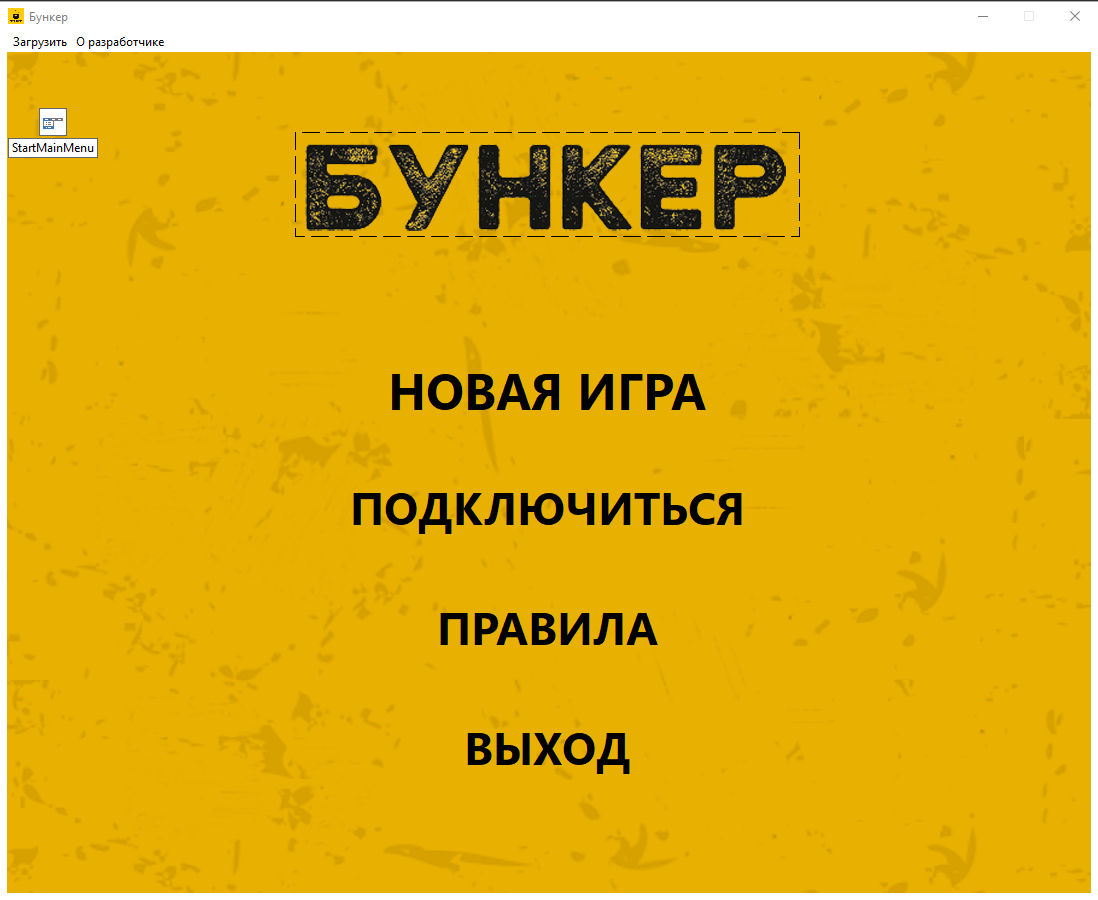


Рисунок 2.1 – Макет начального окна

**2.2.2** Окно с правилами

Окно с правилами состоит из трех компонентов TImage. При нажатии на любой из них компонент скрывается и отображает следующую страницу правил.

Макет окна с правилами представлен на рисунке 2.2.

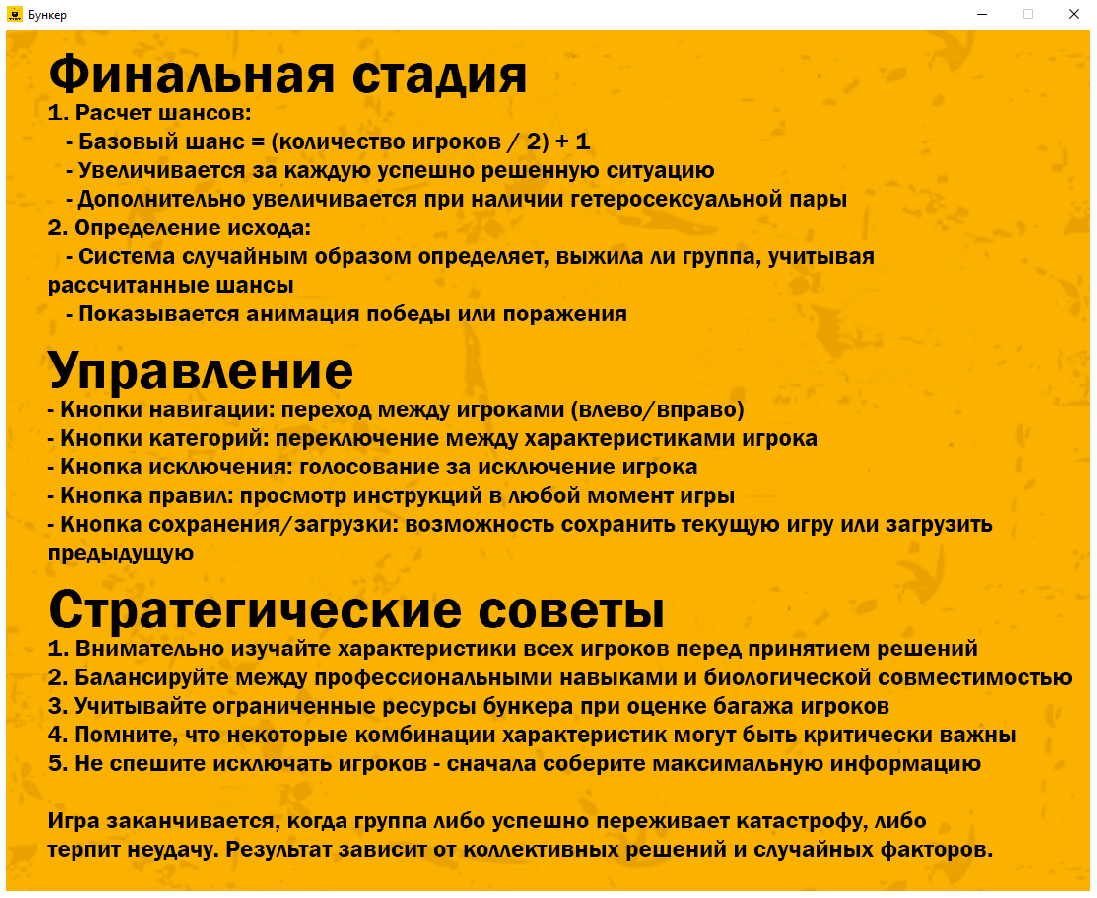


Рисунок 2.2 – Макет окна с правилами

**2.2.3** Окно ввода ключа

Окно ввода ключа состоит из пятнадцати компонентов TImage и двух компонентов TLabel. При нажатии на цифры вводится уникальный ключ, а при нажатии на кнопку "Стереть" удаляется последняя введённая цифра. Кнопка "Назад" возвращает в главное меню, а кнопка "Далее" открывает окно выбора персонажа.

Макет окна ввода ключа представлен на рисунке 2.3.



Рисунок 2.3 – Макет окна ввода ключа

**2.2.4** Окно подготовки

Окно подготовки состоит из трех компонентов TImage и трех компонентов TLabel. При нажатии кнопок "Влево" и "Вправо" количество игроков уменьшается или увеличивается соответственно.

Макет окна подготовки представлен на рисунке 2.4.



Рисунок 2.4 – Макет окна подготовки

**2.2.5** Главное окно

Главное окно состоит из десяти компонентов TImage, шести компонентов TLabel, двух компонентов TTimer, трех кнопок компонента TMainMenu, двух компонентов TVirtualImageList и двух компонентов TImageCollection.

Главное окно приложения представляет собой основной игровой интерфейс, отображаемый после начала игровой сессии. В центре окна расположена карточка игрока с текстовым описанием. Слева и справа от карточки находятся кнопки переключения между категориями информации.

В верхней части интерфейса расположен элемент TLabel, отображающий номер текущего игрока. Для навигации между игроками используются кнопки-стрелки. Снизу находится кнопка исключения игрока, а в левом нижнем углу - кнопка возврата в главное меню.

Кнопка справки активирует отображение таблицы правил. Кнопка перехода к финалу появляется автоматически при выполнении условий завершения игры.

Реализована система сохранения и загрузки игрового прогресса через стандартное меню "Файл".

Макет главного окна представлен на рисунке 2.5.

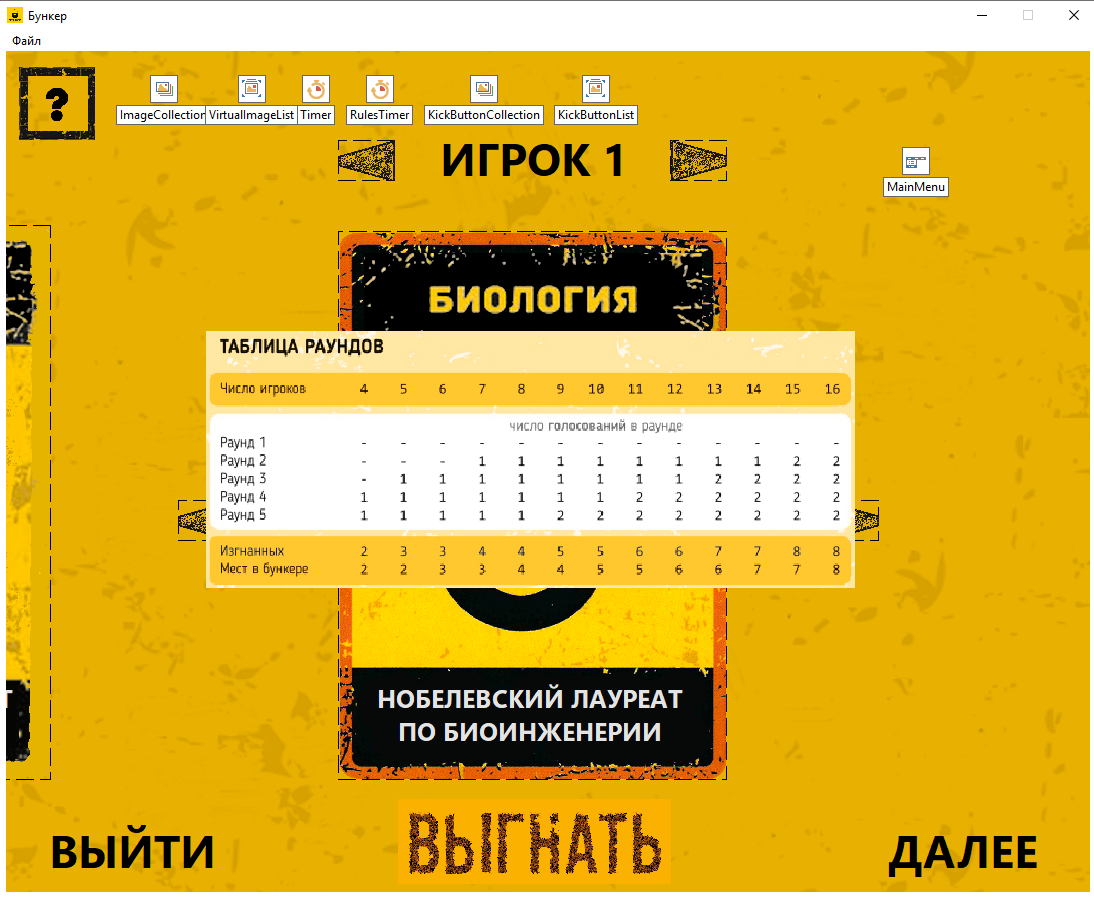


Рисунок 2.5 – Макет главного окна

**2.2.6** Финальное окно

Финальное окно состоит из восьми компонентов TImage, четырех компонентов TLabel и компонента TTimer.

Функциональность финального окна реализована через анимационную последовательность, которая включает несколько этапов:

* появление вопроса с анимированными кнопками ответа;
* отображение информационного сообщения с эффектом постепенного набора текста;
* последовательный показ игровых ситуаций с возможностью выбора ответов;
* расчет и отображение шансов на победу;
* финальную анимацию с затемнением и показом результата.

Макет главного окна представлен на рисунке 2.6.

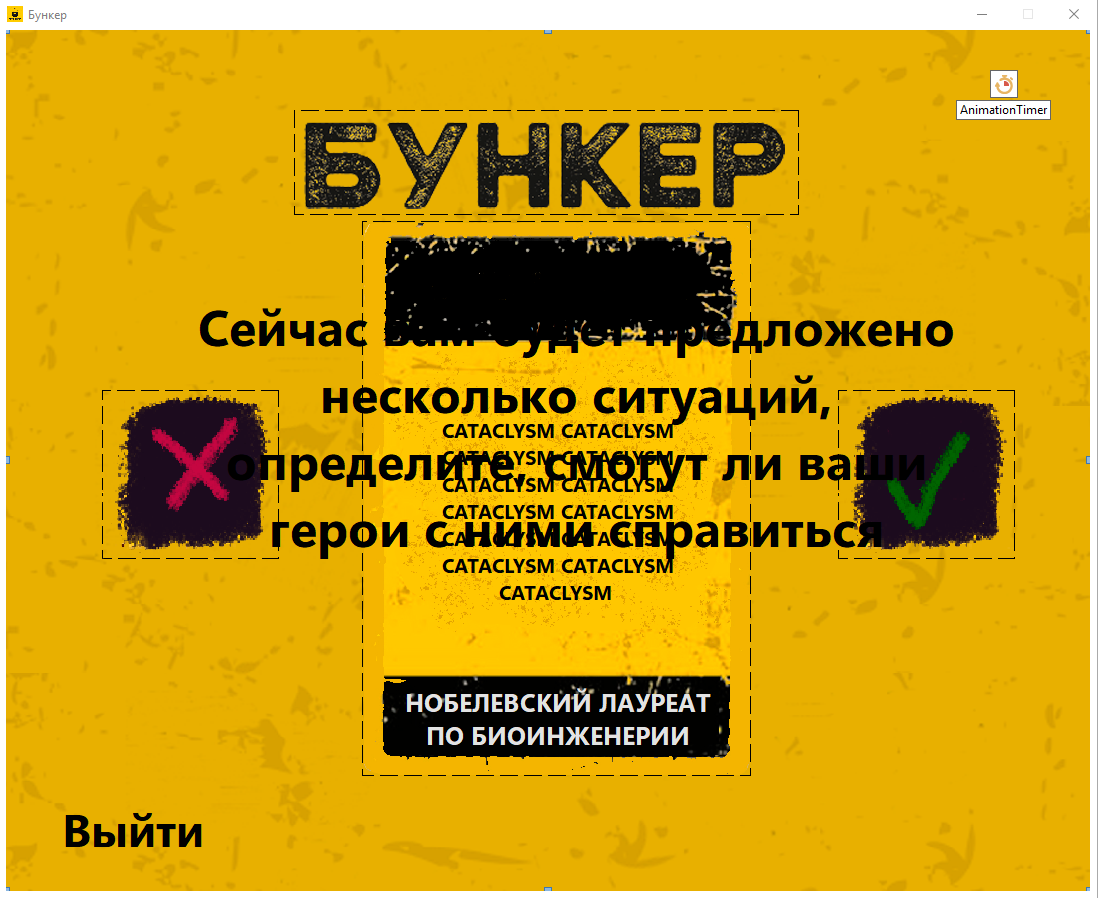


Рисунок 2.6 – Макет финального окна

**2.2.7** Окно подключения

Окно подключения состоит из пяти компонентов TImage и шести компонентов TLabel. В окне отображается ключ игры для подключения, а также происходит выбор активного игрока.

Макет главного окна представлен на рисунке 2.7.



Рисунок 2.7 – Макет окна подключения

## **Проектирование функционала программного средства**

Программное средство представляет собой интерактивную игру, которая моделирует процесс выживания группы людей после глобальной катастрофы. Основные функциональные модули включают управление игровыми картами, обработку игрового процесса и работу с сохранениями.

**2.3.1** Управление игровыми картами

В модуле CardSystem реализована генерация и управление игровыми картами различных категорий: профессии, биологические характеристики, состояние здоровья, хобби, багаж, особые факты, катастрофы и ситуации.

Генерация карт происходит при инициализации игры через процедуру GenerateGameSetup. Ниже приведена блок-схема процедуры GenerateGameSetup на рисунке 2.8.



Рисунок 2.8 – Блок-схема процедуры GenerateGameSetup

**2.3.2** Загрузка ситуаций бункера

Загрузка ситуаций бункера происходит из основного игрового набора FGameSetup в связный список для последующего использования в финальной части игры. Ниже приведена блок-схема процедуры LoadSituationsFromGameSetup на рисунке 2.9.



Рисунок 2.9 – Блок-схема процедуры LoadSituationsFromGameSetup

**2.3.3** Сохранение игры

Система сохранения игры реализована через cтруктуру данных TGameSave, содержащую текущее состояние игры. Блок-схема процедуры сохранения сессии в файл отображена на рисунке 2.10.



Рисунок 2.10 – Блок-схема процедуры SaveDataFile

Эти процедуры реализуют все основные задачи игрового программного средства.

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## **Структура карточной игры**

Целью программы является создание интерактивной карточной игры "Бункер", где игроки принимают решения в постапокалиптическом сценарии. Основной акцент сделан на взаимодействии между игроками и обработке их решений.

Процедура инициализации карт игроков представлена ниже:

// Инициализируются карты игроков

Procedure InitializePlayersCards;

Var

I: Integer;

Cat: TCardCategory;

Begin

// Выделяется память под массив карт игроков

SetLength(FPlayersCards, GameSetup.Players);

// Перебираются все игроки

For I := 0 To GameSetup.Players - 1 Do

Begin

// Устанавливается, что карта не раскрывается

FPlayersCards[I].Revealed := False;

// Устанавливается, что игрок не выбывает

FPlayersCards[I].Kicked := False;

// Очищаются тексты карт для всех категорий

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[I].CardText[Cat] := '';

End;

End;

// Проверяется, есть ли игроки

If GameSetup.Players > 0 Then

Begin

// Устанавливается, что карта текущего игрока раскрывается

FPlayersCards[CurrentPlayerIndex].Revealed := True;

// Заполняются тексты карт для всех категорий текущего игрока

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[CurrentPlayerIndex].CardText[Cat] :=

GameSetup.Cards[GetCardTypeFromCategory(Cat)

][CurrentPlayerIndex].Description;

End;

End;

End;

Данная процедура автоматизирует процесс раздачи карт. После базовой инициализации система заполняет карты текущего игрока реальными значениями из игрового набора. Это позволяет сразу отобразить информацию для первого игрока, в то время как карты остальных остаются скрытыми.

## **Визуальные эффекты и анимация**

Для улучшения пользовательского опыта в игре реализована система плавных переходов между состояниями. Основные визуальные эффекты включают: плавное появление/исчезание элементов, анимацию изменения размеров карт, затемнение исключенных игроков. Система анимации построена на основе таймера и управляется через параметры прозрачности. Ниже представлен код эффекта перелистывания карт:

Procedure StartAnimation(TargetIndex: Integer);

Begin

FTargetIndex := TargetIndex; // Целевое состояние

PrepareBitmaps; // Подготовка изображений

FAlpha := 0; // Начальное значение прозрачности

Timer.Enabled := True; // Запуск анимации

// Эффект уменьшения карты перед переходом

Card.Width := Card.Width Div 2;

Card.Left := Card.Left + Card.Width Div 2;

End;

Каждый кадр анимации пересчитывает параметры отображения и применяет их к графическим элементам. Это создает эффект плавного перехода между различными состояниями интерфейса.

## **Игровая логика и механики**

Основной игровой процесс строится вокруг нескольких ключевых механик.

**3.3.1** Просмотр характеристик игроков

Игроки могут поочередно открывать карты друг друга, чтобы получить информацию для принятия решений. Реализация включает проверку состояния карты перед отображением. Ниже представлена процедура открытия карт:

Procedure ToggleCardReveal;

Var

PlayerIndex: Integer;

Begin

PlayerIndex := GetPlayerIndex;

// Открывается карта только если она была скрыта

If Not FPlayersCards[PlayerIndex].Revealed Then

Begin

FPlayersCards[PlayerIndex].Revealed := True;

// Загружается описание из игрового набора

FPlayersCards[PlayerIndex].CardText[FCurrentCategory] :=

GameSetup.Cards[GetCardTypeFromCategory(FCurrentCategory)

][PlayerIndex].Description;

ShowCurrentCard; // Обновляется отображение

End;

End;

**3.3.2** Исключение игроков

Важнейший элемент стратегии – возможность исключать игроков из бункера. Этот процесс включает: подтверждение действия через диалоговое окно, визуальное обозначение исключенного игрока и обновление интерфейса. Ниже представлена процедура исключения игрока:

Procedure KickButtonClick(Sender: TObject);

Var

PlayerIndex: Integer;

Begin

PlayerIndex := GetPlayerIndex;

// Запрос подтверждения действия

If MessageBox(Handle, 'Вы действительно хотите выгнать игрока?',

'Подтверждение', MB\_YESNO Or MB\_ICONQUESTION) = IDYES Then

Begin

FPlayersCards[PlayerIndex].Kicked := True; // Помечается как исключенный

UpdateKickButton; // Обновляется кнопка

KickCardReveal; // Показываются все карты исключенного

End;

End;

Эти процедуры реализуют логику и механику игрового программного средства.

## **Финальная стадия игры**

Когда остается половина игроков, система переходит к финальной стадии, где оцениваются шансы на выживание. Анализируются характеристики оставшихся игроков, проверяется наличие гетеросексуальной пары и генерируется случайное событие с учетом рассчитанных вероятностей.

Ниже представлена процедура подсчета шансов:

Procedure CountTheChance;

Begin

// Учет гетеросексуальной пары

If HasPair Then

Inc(Chance);

// Расчет процентного соотношения

Percent := Round((Chance / MaxChance) \* 100);

// Запуск анимации отображения результата

AnimationStep := 200;

AnimationTimer.Enabled := True;

End;

Визуальное представление результата включает анимированное появление текста и финального изображения (победа/поражение). Ниже представлена процедура отвечающая за финальную анимацию:

Procedure ShowNextImage;

Var

FadeAlpha: Integer;

FadeBitmap: TBitmap;

MaxChance: Integer;

TheEnd: Integer;

Begin

// Обрабатывается текущий шаг анимации

Case AnimationStep Of

0:

Begin

FadeInImage(QuestionImage);

AnimationTimer.Interval := 600;

Inc(AnimationStep);

End;

1:

Begin

// Проверяется наличие пары

If HasPair Then

Begin

// Устанавливается позиция кнопки "Да"

YesButton.Left := 464;

FadeInImage(YesButton);

End

Else

Begin

// Устанавливается позиция кнопки "Нет"

NoButton.Left := 464;

FadeInImage(NoButton);

End;

AnimationTimer.Interval := 200;

Inc(AnimationStep);

End;

2 .. 19:

Begin

// Мигает соответствующая кнопка в зависимости от наличия пары

If HasPair Then

BlinkImage(YesButton)

Else

BlinkImage(NoButton);

Inc(AnimationStep);

End;

20:

Begin

// Скрываются элементы интерфейса

QuestionImage.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Caption := '';

InfoLabel.Visible := True;

// Устанавливается полное сообщение

FullMessage :=

'Сейчас вам будет предложено несколько ситуаций,

определите, смогут ли ваши герои с ними справиться';

CurrentTextPos := 0;

// Устанавливается интервал таймера

AnimationTimer.Interval := 50;

// Увеличивается шаг анимации

Inc(AnimationStep);

End;

21 .. 120:

Begin

// Отображается текст посимвольно

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 120 Then

Begin

// Устанавливается задержка после полного отображения текста

AnimationTimer.Interval := 4000;

End;

Inc(AnimationStep);

End;

121:

Begin

// Обновляются элементы интерфейса для отображения ситуации

InfoLabel.Visible := False;

BunkerSituationImage.Visible := True;

BunkerSituation.Visible := True;

SituationTitle.Visible := True;

// Загружаются ситуации из настроек игры

LoadSituationsFromGameSetup;

// Инициализируется первая ситуация

InitializeSituation(FSituations.First);

// Отображаются кнопки ответа

YesButton.Visible := True;

NoButton.Visible := True;

YesButton.Top := 360;

YesButton.Left := 832;

NoButton.Top := 360;

NoButton.Left := 96;

YesButton.OnClick := YesButtonClick;

NoButton.OnClick := NoButtonClick;

Inc(AnimationStep);

End;

200:

Begin

// Подготавливается метка для отображения шансов

InfoLabel.Caption := '';

InfoLabel.Visible := True;

MaxChance := (FGameSetup.Players Div 2) + 1;

If MaxChance = 0 Then

MaxChance := 1;

// Формируется сообщение о шансах

FullMessage := 'Ваши шансы победить: ' +

IntToStr(Percent) + '%';

CurrentTextPos := 0;

AnimationTimer.Interval := 20;

Inc(AnimationStep);

End;

201 .. 250:

Begin

// Отображается текст посимвольно

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 250 Then

Begin

AnimationTimer.Interval := 2000;

End;

Inc(AnimationStep);

End;

251:

Begin

// Скрываются элементы интерфейса

InfoLabel.Visible := False;

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

QuestionImage.Visible := False;

Zagolovok.Visible := False;

BackButtonLabel.Visible := False;

AnimationTimer.Interval := 150;

Inc(AnimationStep);

End;

252 .. 280:

Begin

// Рассчитывается прозрачность для эффекта затухания

FadeAlpha := (AnimationStep - 251) \* 10;

If FadeAlpha > 255 Then

FadeAlpha := 255;

// В конце анимации отображается результат

If AnimationStep = 280 Then

Begin

// Определяется исход игры случайным образом с учетом шансов

If Random(100) < Percent Then

TheEnd := 1

Else

TheEnd := 0;

// Отображается экран победы или поражения

If TheEnd = 1 Then

Win.Visible := True

Else

Lose.Visible := True;

// Скрываются оставшиеся элементы интерфейса

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Visible := False;

// Останавливается таймер анимации

AnimationTimer.Enabled := False;

End;

Inc(AnimationStep);

End;

End;

End;

Эти процедуры реализуют финальную часть программного средства.

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Основной проблемой являлись ошибки отображения карт игрока. При переключении категорий отображался некорректный текст или несоответствующая изображению картинка. Проблема была вызвана тем, что процедуры обработки изменяемой информации получали неверные индексы. Проблема представлена на рисунке 4.1.



Рисунок 4.1 – Проблема с фильтрацией информации

Проблема была решена изменением процедуры отображения карт, которая имеет вид:

Procedure ShowCurrentCard;

Var

PlayerIndex: Integer;

CardTextEmpty: Boolean;

BlendFunc: TBlendFunction;

TempBmp: TBitmap;

Begin

// Получение индекса текущего игрока и проверка, пуст ли текст карты

PlayerIndex := GetPlayerIndex;

CardTextEmpty := (FPlayersCards[PlayerIndex].CardText[FCurrentCategory] = '');

// Обрабатка различных случаев отображения в зависимости от состояния

карты и категории

If (FCurrentCategory = CcProfession) And Not FPlayersCards[PlayerIndex].Revealed

And Not Timer.Enabled Then

Begin

// Случай 1: Категория "Профессия", карта не раскрыта, анимация не активна

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else If (Not FPlayersCards[PlayerIndex].Revealed Or CardTextEmpty)

And Not Timer.Enabled Then

Begin

// Случай 2: Карта не раскрыта или текст пуст, анимация не активна

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else

Begin

// Стандартный случай: Показ карты с соответствующей анимацией и текстом

StartAnimation(GetCategoryIndex(FCurrentCategory));

ProfLabel.Caption := FPlayersCards[PlayerIndex].CardText[FCurrentCategory];

End;

End;

Большинство проблем возникло при работе с TVirtualImageList, на стадии тестирования все проблемы были исправлены.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## **Интерфейс программного средства**

**5.1.1** Начальное окно

Начальное окно программы, изображенное на рисунке 5.1, содержит следующие кнопки:

* “Новая игра”;
* “Подключиться”;
* “Правила”;
* “Выход”.

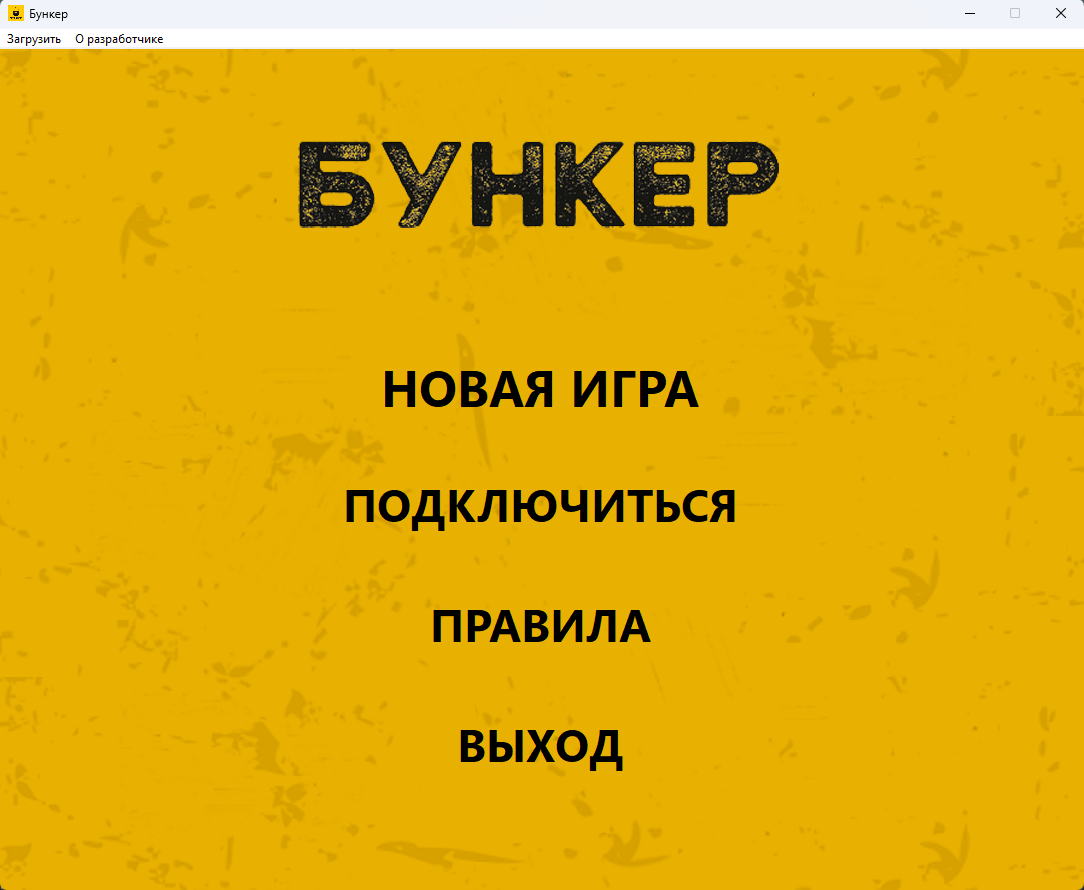


Рисунок 5.1 – Внешний вид начального окна

* + 1. Окна подготовки к игровой сессии

При нажатии на кнопку “Новая игра” пользователю предоставляется окно выбора количества игроков. При нажатии на кнопку “Далее” пользователю предоставляется окно выбора активного игрока. Окна подготовки представлены на рисунке 5.2.

Рисунок 5.2 – Окна подготовки

**5.1.3** Основное окно

После подготовки пользователю представляется окно основной игры, где изображены карты игроков и карта катастрофы. В данном окне осуществляется основной игровой процесс, где пользователи раскрывают карты и исключают игроков. Вид основного окна представлен на рисунке 5.3.

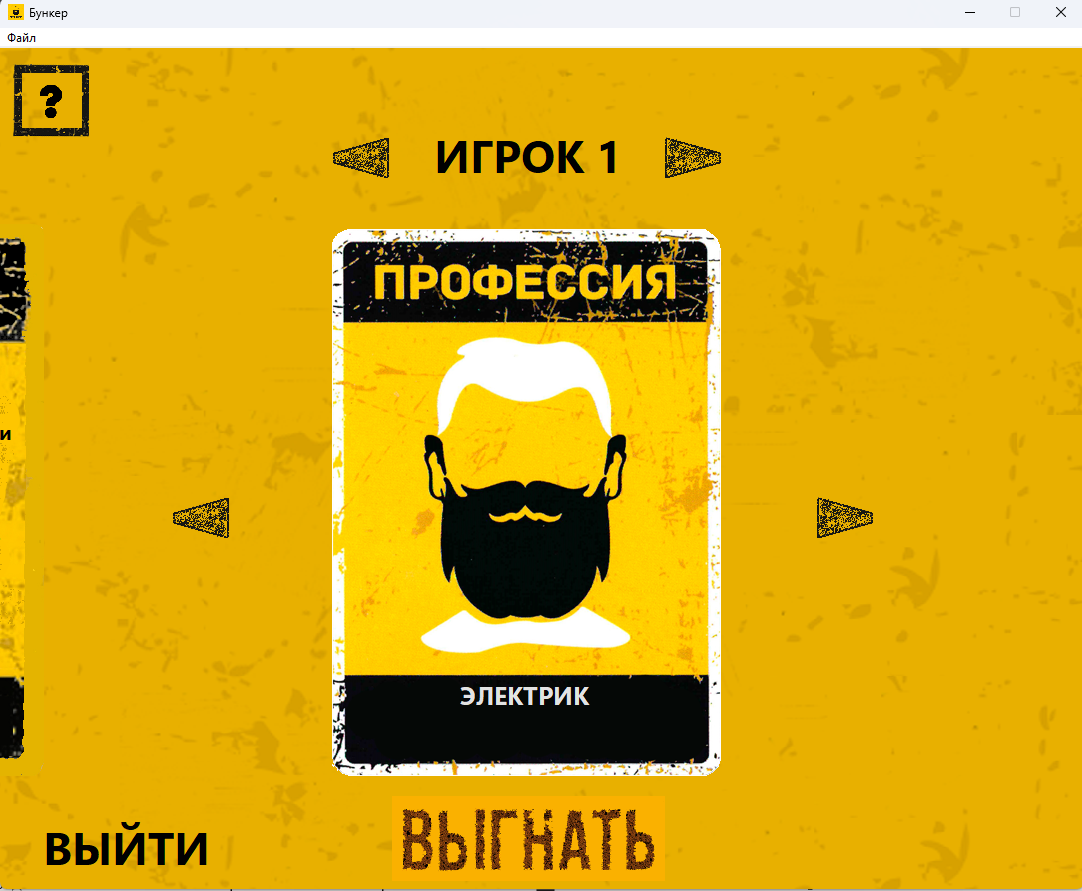


Рисунок 5.3 – Основное окно

**5.1.4** Финальное окно

Когда исключена половина игроков, активируется кнопка "Далее". Ее нажатие запускает финальную стадию игры, включающую анимацию ключевых моментов игры и демонстрацию карт ситуаций. После ответа на все предложенные ситуации система отображает финальный результат – выжили игроки или нет. Интерфейс финального окна представлен на рисунке 5.4.

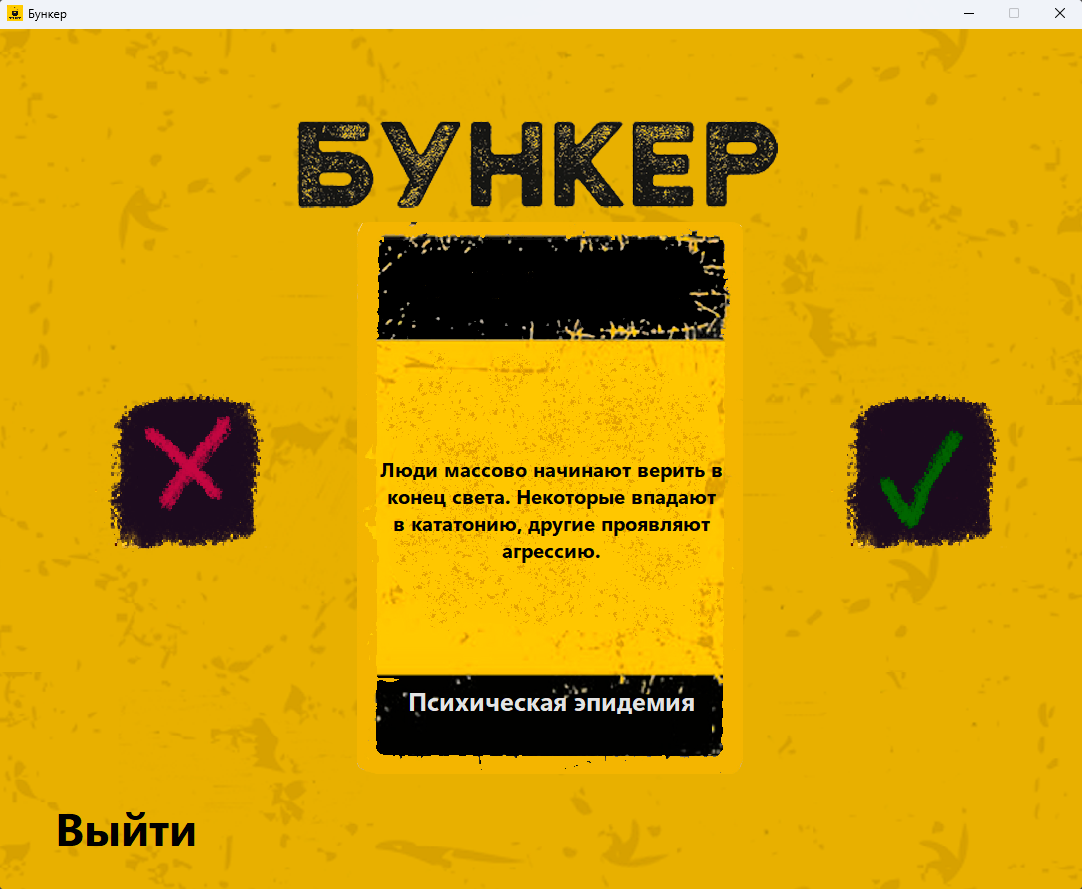


Рисунок 5.4 – Вид финального окна

После анимаций и результатов, пользователь вернется на начальное окно.

## **Управление программным средством**

После запуска программы пользователь может выбрать один из трёх вариантов начала работы: загрузить ранее сохранённую игру из файла, начать новую игру с произвольными параметрами или возобновить игру с заранее известными параметрами по ключу.

Для того чтобы загрузить данные из файла, нужно нажать кнопку “Файл” – “Открыть” в меню сверху экрана.

Кнопки “Влево” и “Вправо” сверху экрана позволяют циклически переключаться между игроками. Текущий активный игрок отображается в центральной верхней части экрана “ИГРОК N”.

.

Просмотр категорий карт осуществляется кнопками “Влево” и “Вправо” по бокам от карт игрока. Для открытия карты нужно кликнуть по центральному изображению карты. Название текущей категории отображается под картой.

Для исключения текущего игрока нажмите кнопку “Выгнать” внизу экрана. Перед исключением система запросит подтверждение действия.

Чтобы переключить страницу правил, нажмите на любое место экрана в окне с правилами.

Чтобы получить подсказку во время игры, нажмите на значок “?” в левом верхнем углу экрана.

Интерфейс разработан с учётом эргономических требований и обеспечивает комфортный игровой процесс как для новичков, так и для опытных пользователей.

# ЗАКЛЮЧЕНИЕ

Результатом выполнения данной курсовой работы стало разработанное программное средство – интерактивная карточная игра “Бункер”, созданная в интегрированной среде разработки Delphi 12.0. В процессе реализации проекта удалось достичь поставленных целей: создать увлекательную игру с удобным интерфейсом, реалистичной механикой принятия решений и комплексной системой оценки шансов на выживание.

В ходе выполнения проекта была проведена значительная работа:

* анализ существующих аналогов и механик настольных игр;
* проектирование гибкой архитектуры с возможностью расширения;
* реализация сложной логики взаимодействия игровых элементов;
* тестирование баланса и пользовательского опыта.

Особое внимание было уделено: пользовательскому интерфейсу, реалистичности механик, визуальному оформлению.

Перспективы развития проекта включают: добавление новых типов катастроф и игровых ситуаций, реализацию сетевого режима для многопользовательской игры, углубление системы характеристик игроков, разработку мобильной версии приложения, введение системы достижений и рейтингов.

Таким образом, данный курсовой проект представляет собой законченное игровое приложение, сочетающее в себе: увлекательный геймплей, глубокую механику принятия решений, качественную визуальную реализацию, гибкую систему настройки параметров игры.

Проект может быть использован как для развлекательных целей, так и в образовательном процессе для развития навыков стратегического мышления и анализа сложных ситуаций. Дальнейшая разработка позволит расширить аудиторию и игровые возможности приложения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Григорьев А.Б. О чем не пишут в книгах по Delphi [Текст]. – СПб.: БХВ-Петербург, 2010. – 576 с. – ISBN 978-5-699-40703-3.
2. Культин Н.Б. Delphi в задачах и примерах [Текст]. – СПб.: БХВ-Петербург, 2012. – 288 с. – ISBN 978-5-9775-0811-7.
3. Осипов Д.Л. Алгоритмы и структуры данных в Delphi [Текст]. – СПб.: БХВ-Петербург, 2011. – 752 с. – ISBN 978-5-9775-0659-5.
4. Фаронов В. Delphi. Программирование на языке высокого уровня [Текст]: Учебник для вузов. – СПб.: Питер, 2009. – 640 с. – ISBN 978-5-9775-0657-1.
5. Фаулер М. Предметно-ориентированные языки программирования / [Текст]. – М.: Вильямс, 2011. – 576 с. – ISBN 978-5-8459-1738-6.
6. Фленов М.Е. Библия Delphi [Текст]. 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 800 с. – ISBN 978-5-397-01304-8.
7. Чиртик А. Программирование в Delphi. Трюки и эффекты [Текст]. – СПб.: Питер, 2011. – 400 с. – ISBN 978-5-8046-0008-3.
8. Официальная документация Embarcadero Delphi [Электронный ресурс]. URL: https://docwiki.embarcadero.com/.
9. Осипов Д. Л. Базы данных и Delphi. Теория и практика. – СПб.: БХВ-Петербург, 2011. – 752 с. – ISBN 978-5-9775-0659-5.

# ПРИЛОЖЕНИЕ А

**ИСХОДНЫЙ КОД**

Unit UnitMain;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.Imaging.Jpeg,

Vcl.ExtCtrls,

Vcl.StdCtrls,

UnitCategory,

UnitResult,

Instruction,

UnitDeveloper,

UnitLight,

UnitCompereQuestions,

Vcl.Imaging.Pngimage,

UnitResultsFromFile;

Type

TForm1 = Class(TForm)

Image1: TImage;

Label1: TLabel;

Categories: TLabel;

Results: TLabel;

Instruction: TLabel;

Label5: TLabel;

Author: TLabel;

LabelExit: TLabel;

Procedure CategoriesClick(Sender: TObject);

Procedure ResultsClick(Sender: TObject);

Procedure InstructionClick(Sender: TObject);

Procedure CategoriesMouseEnter(Sender: TObject);

Procedure CategoriesMouseLeave(Sender: TObject);

Procedure AuthorClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure LabelExitMouseEnter(Sender: TObject);

Procedure LabelExitMouseLeave(Sender: TObject);

Procedure LabelExitClick(Sender: TObject);

Procedure FormClose(Sender: TObject; Var Action: TCloseAction);

Procedure SaveResultToFile(Const FileName: String; Last: String);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

Form1: TForm1;

Const

FileName = 'GenResults.txt';

Implementation

{$R \*.dfm}

Procedure TForm1.AuthorClick(Sender: TObject);

Begin

Application.CreateForm(TFormDeveloper, FormDeveloper);

FormDeveloper.ShowModal;

FormDeveloper.Destroy();

End;

Procedure TForm1.CategoriesClick(Sender: TObject);

Begin

Application.CreateForm(TCategoryForm, CategoryForm);

CategoryForm.ShowModal;

CategoryForm.Destroy();

End;

Procedure TForm1.CategoriesMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

Light(Sender As TLabel);

End;

Procedure TForm1.CategoriesMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Function ResultAll: String;

Var

Correct, UnCorrect, Procent, All, NotAns: Integer;

Begin

Correct := 0;

UnCorrect := 0;

Procent := 0;

All := 0;

NotAns := 0;

For Var J := 1 To 3 Do

Begin

All := All + Length(Table1[J].Questions);

For Var I := 0 To Length(Table1[J].Questions) - 1 Do

Begin

If Table1[J].Questions[I].Answer = Table1[J].Questions[I].Useranswer Then

Begin

Inc(Correct);

End

Else

If Table1[J].Questions[I].Useranswer = 'none' Then

Inc(NotAns)

Else

Begin

Inc(UnCorrect);

End;

End;

End;

Result := IntToSTR((Correct \* 100) Div All);

End;

Procedure TForm1.SaveResultToFile(Const FileName: String; Last: String);

Var

F: TextFile;

Num: Integer;

Date: Tdatetime;

Begin

If TryStrToInt(Last, Num) And (Num <> 0) Then

Begin

AssignFile(F, FileName);

If FileExists(FileName) Then

Append(F)

Else

Rewrite(F);

Date := Now;

WriteLn(F, Num, ' ', DateToStr(Date));

CloseFile(F);

End;

End;

Procedure TForm1.FormClose(Sender: TObject; Var Action: TCloseAction);

Var

Dialog: TForm;

DialogResult: Integer;

Begin

Dialog := CreateMessageDialog(' Are you sure??', MtConfirmation, [MbYes, MbNo], MbNo, ['Yes', 'No']);

Dialog.Caption := 'Exit';

If Dialog.ShowModal = MrYes Then

Begin

SaveResultToFile(FileName, ResultAll);

FreeList;

Action := CaFree;

End

Else

Begin

Action := CaNone;

End;

End;

Procedure TForm1.FormCreate(Sender: TObject);

Begin

CreateTableResult;

End;

Procedure TForm1.ResultsClick(Sender: TObject);

Begin

Application.CreateForm(TResultForm, ResultForm);

ResultForm.ShowModal;

ResultForm.Destroy();

End;

Procedure TForm1.InstructionClick(Sender: TObject);

Begin

Application.CreateForm(TFormInstruction, FormInstruction);

FormInstruction.ShowModal;

FormInstruction.Destroy();

End;

Procedure TForm1.LabelExitClick(Sender: TObject);

Var

Dialog: TForm;

DialogResult: Integer;

Begin

Close;

End;

Procedure TForm1.LabelExitMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

Light(Sender As TLabel);

End;

Procedure TForm1.LabelExitMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

End.

Unit UnitAudition;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.Imaging.Jpeg,

Vcl.ExtCtrls,

Vcl.MPlayer,

UnitQuestions,

Vcl.StdCtrls,

UnitCompereQuestions,

UnitLight;

Type

TFormAudition = Class(TForm)

ImageBack: TImage;

MediaPlayerTask1: TMediaPlayer;

MediaPlayerTask2: TMediaPlayer;

MediaPlayerTask3: TMediaPlayer;

MediaPlayerTask4: TMediaPlayer;

MediaPlayerTask5: TMediaPlayer;

MediaPlayerTask6: TMediaPlayer;

MediaPlayerTask7: TMediaPlayer;

MediaPlayerTask8: TMediaPlayer;

LabelTask1: TLabel;

LabelTask2: TLabel;

LabelTask3: TLabel;

LabelTask4: TLabel;

LabelTask5: TLabel;

LabelTask6: TLabel;

LabelTask7: TLabel;

LabelTask8: TLabel;

ComboBoxAnswer1: TComboBox;

ComboBoxAnswer2: TComboBox;

ComboBoxAnswer3: TComboBox;

ComboBoxAnswer4: TComboBox;

ComboBoxAnswer5: TComboBox;

ComboBoxAnswer6: TComboBox;

ComboBoxAnswer7: TComboBox;

ComboBoxAnswer8: TComboBox;

LabelTask: TLabel;

LabelEnd: TLabel;

Procedure FormCreate(Sender: TObject);

Procedure AddListenQuestions;

Procedure OpenTestQuestionsCom(FName: String; Var QuestionsWord: TQueueList);

Procedure FixUserAnswer;

Procedure LabelMenuClick(Sender: TObject);

Procedure LabelEndClick(Sender: TObject);

Procedure LabelEndMouseEnter(Sender: TObject);

Procedure LabelEndMouseLeave(Sender: TObject);

Procedure InitMediaPlayers;

Procedure StopPleer();

Procedure MediaPlayerTask1Click(Sender: TObject; Button: TMPBtnType; Var DoDefault: Boolean);

Private

{ Private declarations }

Public

Pleer: TMediaPlayer;

{ Public declarations }

End;

Var

FormAudition: TFormAudition;

ListenQuestionsList: TQueueList;

Const

FileQuestions = 'AuditionQuestions.txt';

Implementation

{$R \*.dfm}

Procedure TFormAudition.FormCreate(Sender: TObject);

Var

I: Integer;

Begin

OpenTestQuestionsCom(FileQuestions, ListenQuestionsList);

AddListenQuestions;

InitMediaPlayers;

End;

Procedure TFormAudition.InitMediaPlayers;

Var

I: Integer;

MediaPlayer: TMediaPlayer;

FilePath: String;

Begin

For I := 1 To 8 Do

Begin

MediaPlayer := TMediaPlayer(FindComponent('MediaPlayerTask' + IntToStr(I)));

If Assigned(MediaPlayer) Then

Begin

FilePath := 'audio' + IntToStr(I) + '.wav';

MediaPlayer.FileName := FilePath;

MediaPlayer.Open;

End;

End;

End;

Procedure TFormAudition.StopPleer();

Var

I: Integer;

MediaPlayer: TMediaPlayer;

Begin

For I := 1 To 8 Do

Begin

MediaPlayer := TMediaPlayer(FindComponent('MediaPlayerTask' + IntToStr(I)));

If Assigned(MediaPlayer) Then

Begin

If Pleer <> MediaPlayer Then

Begin

If MediaPlayer.Mode = MpPlaying Then

MediaPlayer.Pause;

End

Else

Begin

If MediaPlayer.Mode = MpStopped Then

Begin

Begin

MediaPlayer.Open;

End;

End;

End;

End;

End;

End;

Procedure TFormAudition.LabelEndClick(Sender: TObject);

Begin

FixUserAnswer;

Close;

End;

Procedure TFormAudition.LabelEndMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

LightGrey(Sender As TLabel);

End;

Procedure TFormAudition.LabelEndMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Procedure TFormAudition.LabelMenuClick(Sender: TObject);

Begin

Close;

End;

Procedure TFormAudition.MediaPlayerTask1Click(Sender: TObject; Button: TMPBtnType; Var DoDefault: Boolean);

Begin

Pleer := TMediaPlayer(Sender);

StopPleer;

End;

Procedure TFormAudition.AddListenQuestions;

Var

I, J, K: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

T := ListenQuestionsList.PHead;

J := 0;

For I := 1 To ListenQuestionsList.CountQ Do

Begin

C := FindComponent('LabelTask' + IntToStr(I));

M := FindComponent('ComboBoxAnswer' + IntToStr(I));

If (C Is TLabel) Then

Begin

TLabel(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

AnswerF := T^.AnswerList.PHead;

While AnswerF <> Nil Do

Begin

If (M Is TComboBox) Then

Begin

TComboBox(M).ItemIndex := -1;

TComboBox(M).Items.Add(AnswerF^.Answer);

AnswerF := AnswerF^.PNext;

End;

End;

If Table1[3].Questions[J].Useranswer <> 'none' Then

Begin

For K := 0 To TComboBox(M).Items.Count Do

If TComboBox(M).Items[K] = Table1[3].Questions[J].Useranswer Then

TComboBox(M).ItemIndex := K;

End;

Inc(J);

End;

T := T^.PNext;

End;

End;

Procedure TFormAudition.OpenTestQuestionsCom(FName: String; Var QuestionsWord: TQueueList);

Var

F: TextFile;

MainQuestion, Ans, Current: String;

ColAnswer, I: Integer;

TmpAnswerList: TQueueAnswerList;

Begin

AssignFile(F, FName);

Reset(F);

Try

While Not EOF(F) Do

Begin

Readln(F, MainQuestion);

Readln(F, ColAnswer);

TmpAnswerList.PHead := Nil;

TmpAnswerList.PTail := Nil;

TmpAnswerList.Count := 0;

For I := 1 To ColAnswer Do

Begin

Readln(F, Ans);

AddAnswer(TmpAnswerList, Ans);

End;

Readln(F, Current);

AddAllAnswer(QuestionsWord, TmpAnswerList, Current, MainQuestion);

End;

Except

End;

CloseFile(F);

End;

Procedure TFormAudition.FixUserAnswer;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := ListenQuestionsList.PHead;

CurrentIndex := 0;

For I := 1 To ListenQuestionsList.CountQ Do

Begin

C := FindComponent('LabelTask' + IntToStr(I));

M := FindComponent('ComboBoxAnswer' + IntToStr(I));

If (C Is TLabel) Then

Begin

Table1[3].Questions[CurrentIndex].Question := T^.Questions;

Table1[3].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TComboBox(M).Text <> '' Then

Table1[3].Questions[CurrentIndex].Useranswer := TComboBox(M).Text;

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

End.

Unit UnitCategory;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.ExtCtrls,

Vcl.Imaging.Jpeg,

Vcl.StdCtrls,

Vcl.Imaging.Pngimage,

UnitSpeaking,

UnitLight,

UnitGrammer,

UnitAudition;

Type

TCategoryForm = Class(TForm)

Image1: TImage;

Label1: TLabel;

LabelCommunicative: TLabel;

LabelGrammar: TLabel;

LabelAudition: TLabel;

LabelBack: TLabel;

Procedure LabelCommunicativeClick(Sender: TObject);

Procedure LabelAuditionMouseEnter(Sender: TObject);

Procedure LabelAuditionMouseLeave(Sender: TObject);

Procedure LabelGrammarClick(Sender: TObject);

Procedure LabelAuditionClick(Sender: TObject);

Procedure LabelBackClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

CategoryForm: TCategoryForm;

Implementation

{$R \*.dfm}

Procedure TCategoryForm.LabelCommunicativeClick(Sender: TObject);

Begin

CategoryForm.Hide;

Application.CreateForm(TForm\_Speaking, Form\_Speaking);

Form\_Speaking.CurrentAnwerTest := 1;

Form\_Speaking.ShowModal;

Form\_Speaking.Destroy();

Close;

End;

Procedure TCategoryForm.LabelGrammarClick(Sender: TObject);

Begin

CategoryForm.Hide;

Application.CreateForm(TFormGrammer, FormGrammer);

FormGrammer.ShowModal;

FormGrammer.Destroy();

Close;

End;

Procedure TCategoryForm.LabelAuditionClick(Sender: TObject);

Begin

CategoryForm.Hide;

Application.CreateForm(TFormAudition, FormAudition);

FormAudition.ShowModal;

FormAudition.Destroy();

Close;

End;

Procedure TCategoryForm.LabelAuditionMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

LightGrey(Sender As TLabel);

End;

Procedure TCategoryForm.LabelAuditionMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Procedure TCategoryForm.LabelBackClick(Sender: TObject);

Begin

Close;

End;

End.

Unit UnitCompereQuestions;

Interface

Type

TQuestionRec = Record

Question: String;

Answer: String;

Useranswer: String;

End;

TTask = Record

Questions: Array Of TQuestionRec;

End;

TTable = Array [1 .. 3] Of TTask;

Var

Table1: TTable;

Procedure CreateTableResult;

Implementation

Procedure CreateTableResult;

Begin

Setlength(Table1[1].Questions, 25);

Setlength(Table1[2].Questions, 31);

Setlength(Table1[3].Questions, 8);

For Var I := 1 To 3 Do

Begin

For Var J := 0 To Length(Table1[I].Questions) - 1 Do

Begin

Table1[I].Questions[J].Useranswer := 'none';

End;

End;

End;

End.

Unit UnitGrammer;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.ComCtrls,

Vcl.Imaging.Jpeg,

Vcl.ExtCtrls,

Vcl.StdCtrls,

Vcl.Imaging.Pngimage,

UnitQuestions,

Vcl.Menus,

UnitLight,

UnitCompereQuestions;

Type

TFormGrammer = Class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

ImageBack1: TImage;

LabelNext: TLabel;

ImageBack2: TImage;

ImageBack3: TImage;

LabelNext2: TLabel;

LabelFinish: TLabel;

LbWriteWord1: TLabel;

LabelMain: TLabel;

LabelTask: TLabel;

Image1: TImage;

LabelTask1: TLabel;

LabelTask2: TLabel;

LabelTask3: TLabel;

LabelTask4: TLabel;

LabelTask5: TLabel;

LabelTask6: TLabel;

LabelTask7: TLabel;

LabelTask8: TLabel;

LabelTask9: TLabel;

LabelTask10: TLabel;

ComboBoxWord1: TComboBox;

ComboBoxWord2: TComboBox;

ComboBoxWord3: TComboBox;

ComboBoxWord4: TComboBox;

ComboBoxWord5: TComboBox;

ComboBoxWord6: TComboBox;

ComboBoxWord7: TComboBox;

ComboBoxWord8: TComboBox;

ComboBoxWord9: TComboBox;

ComboBoxWord10: TComboBox;

LabelTask11: TLabel;

ComboBoxWord11: TComboBox;

LabelBack: TLabel;

LabelTaskForm: TLabel;

LbQuestionForm1: TLabel;

LbQuestionForm2: TLabel;

LbQuestionForm3: TLabel;

LbQuestionForm4: TLabel;

LbQuestionForm5: TLabel;

LbQuestionForm6: TLabel;

LbQuestionForm7: TLabel;

LbQuestionForm8: TLabel;

LbQuestionForm9: TLabel;

LbQuestionForm10: TLabel;

CBQuestionsForm1: TComboBox;

CBQuestionsForm2: TComboBox;

CBQuestionsForm3: TComboBox;

CBQuestionsForm4: TComboBox;

CBQuestionsForm5: TComboBox;

CBQuestionsForm6: TComboBox;

CBQuestionsForm7: TComboBox;

CBQuestionsForm8: TComboBox;

CBQuestionsForm9: TComboBox;

CBQuestionsForm10: TComboBox;

LbWriteWord2: TLabel;

LbWriteWord3: TLabel;

LbWriteWord4: TLabel;

LbWriteWord5: TLabel;

LbWriteWord6: TLabel;

LbWriteWord7: TLabel;

LbWriteWord8: TLabel;

LbWriteWord9: TLabel;

LbWriteWord10: TLabel;

EditWord1: TEdit;

EditWord2: TEdit;

EditWord3: TEdit;

EditWord4: TEdit;

EditWord5: TEdit;

EditWord6: TEdit;

EditWord7: TEdit;

EditWord8: TEdit;

EditWord9: TEdit;

EditWord10: TEdit;

PopupMenuStop: TPopupMenu;

LabelBack2: TLabel;

LabelTaskEnd3: TLabel;

Procedure FormCreate(Sender: TObject);

Procedure LabelMainClick(Sender: TObject);

Procedure LabelNext2Click(Sender: TObject);

Procedure LabelNextClick(Sender: TObject);

Procedure LabelFinishClick(Sender: TObject);

Procedure OpenTestWordCom1(FName: String; Var QuestionsWord1: TQueueList);

Procedure AddTestWords;

Procedure LabelBackClick(Sender: TObject);

Procedure AddTestWords2;

Procedure OpenTestWriteWord(FName: String; Var QuestionsWord3: TQueueList);

Procedure AddTestWords3;

Procedure EditWord1KeyPress(Sender: TObject; Var Key: Char);

Procedure LabelBack2MouseEnter(Sender: TObject);

Procedure LabelBack2MouseLeave(Sender: TObject);

Procedure FixUsersAnswer1;

Procedure FixUsersAnswer2;

Procedure FixUsersAnswer3;

Private

{ Private declarations }

Public

{ Public declarations }

QuestionsWord1: TQueueList;

QuestionsWord2: TQueueList;

QuestionsWord3: TQueueList;

End;

Const

File1 = 'Grammer1.txt';

File2 = 'Grammer2.txt';

File3 = 'Grammer3.txt';

Var

FormGrammer: TFormGrammer;

Implementation

{$R \*.dfm}

Procedure TFormGrammer.FixUsersAnswer3;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := QuestionsWord3.PHead;

CurrentIndex := 21;

For I := 1 To QuestionsWord3.CountQ Do

Begin

C := FindComponent('LbWriteWord' + IntToStr(I));

M := FindComponent('EditWord' + IntToStr(I));

If (C Is TLabel) Then

Begin

Table1[2].Questions[CurrentIndex].Question := T^.Questions;

Table1[2].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TEdit(M).Text <> '' Then

Table1[2].Questions[CurrentIndex].Useranswer := TEdit(M).Text;

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

Procedure TFormGrammer.FixUsersAnswer1;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := QuestionsWord1.PHead;

CurrentIndex := 0;

For I := 1 To QuestionsWord1.CountQ Do

Begin

C := FindComponent('LabelTask' + IntToStr(I));

M := FindComponent('ComboBoxWord' + IntToStr(I));

If (C Is TLabel) Then

Begin

Table1[2].Questions[CurrentIndex].Question := T^.Questions;

Table1[2].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TComboBox(M).Text <> '' Then

Table1[2].Questions[CurrentIndex].Useranswer := TComboBox(M).Text;

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

Procedure TFormGrammer.FixUsersAnswer2;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := QuestionsWord2.PHead;

CurrentIndex := 11;

For I := 1 To QuestionsWord2.CountQ Do

Begin

C := FindComponent('LbQuestionForm' + IntToStr(I));

M := FindComponent('CBQuestionsForm' + IntToStr(I));

If (C Is TLabel) Then

Begin

Table1[2].Questions[CurrentIndex].Question := T^.Questions;

Table1[2].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TComboBox(M).Text <> '' Then

Table1[2].Questions[CurrentIndex].Useranswer := TComboBox(M).Text;

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

Procedure TFormGrammer.OpenTestWordCom1(FName: String; Var QuestionsWord1: TQueueList);

Var

F: TextFile;

MainQuestion, Ans, Current: String;

ColAnswer, I: Integer;

TmpAnswerList: TQueueAnswerList;

Begin

Try

AssignFile(F, FName);

Reset(F);

Try

While Not EOF(F) Do

Begin

Readln(F, MainQuestion);

Readln(F, ColAnswer);

TmpAnswerList.PHead := Nil;

TmpAnswerList.PTail := Nil;

TmpAnswerList.Count := 0;

For I := 1 To ColAnswer Do

Begin

Readln(F, Ans);

AddAnswer(TmpAnswerList, Ans);

End;

Readln(F, Current);

AddAllAnswer(QuestionsWord1, TmpAnswerList, Current, MainQuestion);

End;

Except

End;

CloseFile(F);

Except

Showmessage('file is empty(');

Close;

End;

End;

Procedure TFormGrammer.AddTestWords;

Var

I, J, K: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

T := QuestionsWord1.PHead;

J := 0;

For I := 1 To QuestionsWord1.CountQ Do

Begin

C := FindComponent('LabelTask' + IntToStr(I));

M := FindComponent('ComboBoxWord' + IntToStr(I));

If (C Is TLabel) Then

Begin

TLabel(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

AnswerF := T^.AnswerList.PHead;

While AnswerF <> Nil Do

Begin

If (M Is TComboBox) Then

BEGIN

TComboBox(M).ItemIndex := -1;

TComboBox(M).Items.Add(AnswerF^.Answer);

AnswerF := AnswerF^.PNext;

END;

End;

If Table1[2].Questions[J].Useranswer <> 'none' Then

Begin

For K := 0 To TComboBox(M).Items.Count Do

If TComboBox(M).Items[K] = Table1[2].Questions[J].Useranswer Then

TComboBox(M).ItemIndex := K;

End;

Inc(J);

End;

T := T^.PNext;

End;

End;

Procedure TFormGrammer.AddTestWords2;

Var

I, J, K: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

T := QuestionsWord2.PHead;

J := 11;

For I := 1 To QuestionsWord2.CountQ Do

Begin

C := FindComponent('LbQuestionForm' + IntToStr(I));

M := FindComponent('CBQuestionsForm' + IntToStr(I));

If (C Is TLabel) Then

Begin

TLabel(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

AnswerF := T^.AnswerList.PHead;

While AnswerF <> Nil Do

Begin

If (M Is TComboBox) Then

BEGIN

TComboBox(M).ItemIndex := -1;

TComboBox(M).Items.Add(AnswerF^.Answer);

AnswerF := AnswerF^.PNext;

END;

End;

If Table1[2].Questions[J].Useranswer <> 'none' Then

Begin

For K := 0 To TComboBox(M).Items.Count Do

If TComboBox(M).Items[K] = Table1[2].Questions[J].Useranswer Then

TComboBox(M).ItemIndex := K;

End;

Inc(J);

End;

T := T^.PNext;

End;

End;

Procedure TFormGrammer.OpenTestWriteWord(FName: String; Var QuestionsWord3: TQueueList);

Var

F: TextFile;

Question, CorrectAnswer: String;

TmpAnswerList: TQueueAnswerList;

Begin

Try

AssignFile(F, FName);

Reset(F);

Try

While Not EOF(F) Do

Begin

Readln(F, Question);

Readln(F, CorrectAnswer);

TmpAnswerList.PHead := Nil;

TmpAnswerList.PTail := Nil;

AddAllAnswer(QuestionsWord3, TmpAnswerList, CorrectAnswer, Question);

End;

Except

End;

CloseFile(F);

Except

Showmessage('file is empty(');

Close;

End;

End;

Procedure TFormGrammer.AddTestWords3;

Var

I, J, K: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

J := 21;

T := QuestionsWord3.PHead;

For I := 1 To QuestionsWord3.CountQ Do

Begin

C := FindComponent('LbWriteWord' + IntToStr(I));

M := FindComponent('EditWord' + IntToStr(I));

If (C Is TLabel) Then

Begin

TLabel(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

End;

If (Table1[2].Questions[J].Useranswer <> 'none') Then

Begin

TEdit(M).Text := Table1[2].Questions[J].Useranswer;

End;

Inc(J);

T := T^.PNext;

End;

End;

Procedure TFormGrammer.EditWord1KeyPress(Sender: TObject; Var Key: Char);

Begin

If Key In ['0' .. '9'] Then

Key := #0;

End;

Procedure TFormGrammer.FormCreate(Sender: TObject);

Begin

PageControl1.ActivePage := TabSheet1;

OpenTestWordCom1(File1, QuestionsWord1);

AddTestWords;

OpenTestWordCom1(File2, QuestionsWord2);

AddTestWords2;

OpenTestWriteWord(File3, QuestionsWord3);

AddTestWords3;

End;

Procedure TFormGrammer.LabelBack2MouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

LightGrey(Sender As TLabel);

End;

Procedure TFormGrammer.LabelBack2MouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Procedure TFormGrammer.LabelBackClick(Sender: TObject);

Begin

FixUsersAnswer3;

FixUsersAnswer2;

FixUsersAnswer1;

PageControl1.ActivePageIndex := PageControl1.ActivePageIndex - 1;

End;

Procedure TFormGrammer.LabelFinishClick(Sender: TObject);

Begin

FixUsersAnswer3;

Close;

End;

Procedure TFormGrammer.LabelMainClick(Sender: TObject);

Begin

FixUsersAnswer1;

FixUsersAnswer2;

FixUsersAnswer3;

Close;

End;

Procedure TFormGrammer.LabelNext2Click(Sender: TObject);

Begin

FixUsersAnswer2;

FixUsersAnswer1;

FixUsersAnswer3;

PageControl1.ActivePageIndex := PageControl1.ActivePageIndex + 1;

End;

Procedure TFormGrammer.LabelNextClick(Sender: TObject);

Begin

FixUsersAnswer1;

PageControl1.ActivePageIndex := PageControl1.ActivePageIndex + 1;

End;

End.

Unit UnitQuestions;

Interface

Type

TPNodeAnswer = ^TNodeListAnswer;

TNodeListAnswer = Record

Answer: String[200];

PNext: TPNodeAnswer;

End;

TQueueAnswerList = Record

PHead, PTail: TPNodeAnswer;

Count: Integer;

End;

Type

TPNodeList = ^TNodeList;

TNodeList = Record

Questions: String[200];

AnswerList: TQueueAnswerList;

CurrentAnswer: String[200];

PNext: TPNodeList;

End;

TQueueList = Record

PHead, PTail: TPNodeList;

CountQ: Integer;

End;

Procedure AddAnswer(Var List: TQueueAnswerList; Answer: String);

Procedure AddAllAnswer(Var List: TQueueList; Answer: TQueueAnswerList; CurrentAnswer: String; Questions: String);

Implementation

Procedure ClearAnswerList(Var Queue: TQueueAnswerList);

Var

Temp, Current: TPNodeAnswer;

Begin

Current := Queue.PHead;

While Current <> Nil Do

Begin

Temp := Current;

Current := Current^.PNext;

Dispose(Temp);

End;

Queue.PHead := Nil;

Queue.PTail := Nil;

Queue.Count := 0;

End;

Procedure ClearQuestionList(Var Queue: TQueueList);

Var

Temp, Current: TPNodeList;

Begin

Current := Queue.PHead;

While Current <> Nil Do

Begin

Temp := Current;

Current := Current^.PNext;

ClearAnswerList(Temp^.AnswerList);

Dispose(Temp);

End;

Queue.PHead := Nil;

Queue.PTail := Nil;

Queue.CountQ := 0;

End;

Procedure AddAllAnswer(Var List: TQueueList; Answer: TQueueAnswerList; CurrentAnswer: String; Questions: String);

Var

AddElem: TPNodeList;

Begin

New(AddElem);

AddElem.PNext := Nil;

AddElem.AnswerList := Answer;

AddElem.CurrentAnswer := CurrentAnswer;

AddElem.Questions := Questions;

If List.PHead = Nil Then

Begin

List.CountQ := 1;

List.PHead := AddElem;

List.PTail := AddElem;

End

Else

Begin

Inc(List.CountQ);

List.PTail.PNext := AddElem;

List.PTail := AddElem;

End;

End;

Procedure AddAnswer(Var List: TQueueAnswerList; Answer: String);

Var

AddElem: TPNodeAnswer;

Begin

New(AddElem);

AddElem.PNext := Nil;

AddElem.Answer := Answer;

If List.PHead = Nil Then

Begin

List.Count := 1;

List.PHead := AddElem;

List.PTail := AddElem;

End

Else

Begin

Inc(List.Count);

List.PTail.PNext := AddElem;

List.PTail := AddElem;

End;

End;

End.

Unit UnitResult;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.ExtCtrls,

Vcl.Imaging.Jpeg,

Vcl.Grids,

Vcl.StdCtrls,

Math,

UnitLight,

UnitCompereQuestions,

UnitResultsFromFile,

Vcl.ComCtrls;

Type

TResultForm = Class(TForm)

Image1: TImage;

LabelMain: TLabel;

LabelDelete: TLabel;

Label3: TLabel;

EditLastResult: TEdit;

Label4: TLabel;

LabelChoose: TLabel;

CBLastCategory: TComboBox;

StringGridBestResults: TStringGrid;

Label6: TLabel;

StringGrid1: TStringGrid;

LabelStatistic: TLabel;

LabelCorrect: TLabel;

LabelCorr: TLabel;

LabelUnCorrCapt: TLabel;

LabelAllCapt: TLabel;

LabelUnCorrect: TLabel;

LabelAll: TLabel;

LabelResultCapt: TLabel;

LabelRes: TLabel;

Panel1: TPanel;

Panel2: TPanel;

LabelC: TLabel;

LabelGenRes: TLabel;

LabelGenAll: TLabel;

LabelGenUnCor: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label11: TLabel;

Procedure LabelMainClick(Sender: TObject);

Procedure LabelMainMouseEnter(Sender: TObject);

Procedure LabelMainMouseLeave(Sender: TObject);

Procedure CBLastCategoryChange(Sender: TObject);

Procedure FirstCResult(IndexCategory: Integer);

Procedure FormCreate(Sender: TObject);

Procedure GenResult;

Procedure StringGrid1DrawCell(Sender: TObject; ACol, ARow: LongInt; Rect:

TRect; State: TGridDrawState);

Procedure DisplayTop5InGrid(Const Arr: Array Of TNode);

Procedure LabelDeleteClick(Sender: TObject);

Procedure Button1Click(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Const

FileResults = 'GenResults.txt';

Var

ResultForm: TResultForm;

Implementation

{$R \*.dfm}

Procedure TResultForm.Button1Click(Sender: TObject);

Begin

Showmessage(Table1[2].Questions[0].Answer);

Showmessage(Table1[2].Questions[0].Useranswer);

End;

Procedure TResultForm.CBLastCategoryChange(Sender: TObject);

Begin

If CBLastCategory.ItemIndex = 3 Then

Begin

GenResult;

StringGrid1.Visible := False;

Panel1.Visible := False;

End

ELSE

If CBLastCategory.ItemIndex <> -1 Then

Begin

StringGrid1.Visible := True;

FirstCResult(CBLastCategory.ItemIndex);

End;

End;

Procedure TResultForm.FirstCResult(IndexCategory: Integer);

Var

Correct, UnCorrect, Procent, All: Integer;

B: String;

Begin

Panel1.Visible := True;

StringGrid1.Visible := True;

Panel2.Visible := False;

Correct := 0;

UnCorrect := 0;

Procent := 0;

StringGrid1.RowCount := Length(Table1[IndexCategory + 1].Questions) + 1;

All := Length(Table1[IndexCategory + 1].Questions);

For Var I := 0 To Length(Table1[IndexCategory + 1].Questions) - 1 Do

Begin

StringGrid1.Cells[0, I + 1] := '№ ' + IntToSTR(I + 1);

If AnsiLowerCase(Table1[IndexCategory + 1].Questions[I].Answer)

= AnsiLowerCase(Table1[IndexCategory + 1].Questions[I].Useranswer) Then

Begin

StringGrid1.Cells[1, I + 1] := 'Correct';

Inc(Correct);

End

Else

If Table1[IndexCategory + 1].Questions[I].Useranswer = 'none' Then

StringGrid1.Cells[1, I + 1] := 'Not answer'

Else

Begin

StringGrid1.Cells[1, I + 1] := 'Uncorrect';

Inc(UnCorrect);

End;

End;

LabelCorrect.Caption := IntToSTR(Correct);

LabelUnCorrect.Caption := IntToSTR(UnCorrect);

LabelAll.Caption := IntToSTR(Length(Table1[IndexCategory + 1].Questions));

LabelRes.Caption := IntToSTR((Correct \* 100) Div All);

End;

Procedure TResultForm.GenResult;

Var

Correct, UnCorrect, Procent, All, NotAns: Integer;

Begin

StringGrid1.Visible := False;

Panel2.Visible := True;

Correct := 0;

UnCorrect := 0;

Procent := 0;

All := 0;

NotAns := 0;

For Var J := 1 To 3 Do

Begin

All := All + Length(Table1[J].Questions);

For Var I := 0 To Length(Table1[J].Questions) - 1 Do

Begin

If AnsiLowerCase(Table1[J].Questions[I].Answer) = AnsiLowerCase(Table1[J].Questions[I].Useranswer) Then

Begin

Inc(Correct);

End

Else

If Table1[J].Questions[I].Useranswer = 'none' Then

Inc(NotAns)

Else

BEGIN

Inc(UnCorrect);

END;

End;

End;

Label8.Caption := IntToSTR(Correct);

Label11.Caption := IntToSTR(UnCorrect);

Label10.Caption := IntToSTR(All);

Label9.Caption := IntToSTR((Correct \* 100) Div All);

End;

Procedure TResultForm.DisplayTop5InGrid(Const Arr: Array Of TNode);

Var

I, CountToShow: Integer;

Begin

StringGridBestResults.ColCount := 2;

CountToShow := Min(5, Length(Arr));

StringGridBestResults.RowCount := CountToShow + 1;

StringGridBestResults.Cells[0, 0] := 'Top Scores';

StringGridBestResults.Cells[1, 0] := 'date';

For I := 0 To CountToShow - 1 Do

Begin

StringGridBestResults.Cells[0, I + 1] := IntToStr(Arr[I].Value);

StringGridBestResults.Cells[1, I + 1] := Arr[I].Date;

End;

End;

Procedure TResultForm.FormCreate(Sender: TObject);

Begin

ReadFromFile;

CountList;

ListToArray;

SortDescending;

If (Head <> Nil) And (LastValue <> 0) Then

Begin

EditLastResult.Text := IntToStr(LastValue);

DisplayTop5InGrid(Arr);

End

Else

Begin

EditLastResult.Text := 'There are no recent results.';

StringGridBestResults.RowCount := 1;

StringGridBestResults.ColCount := 2;

StringGridBestResults.Cells[0, 0] := 'Top Scores';

StringGridBestResults.Cells[1, 0] := 'date';

End;

FreeList;

StringGrid1.Visible := False;

Panel1.Visible := False;

StringGrid1.ColWidths[0] := 35;

StringGrid1.Cells[0, 0] := '№';

StringGrid1.Cells[1, 0] := 'Your answer';

Panel1.Color := ClWhite;

Panel1.ParentBackground := False;

Panel2.Color := ClWhite;

Panel2.ParentBackground := False;

StringGrid1.Visible := False;

GenResult;

End;

Procedure ClearFile(Const FileName: String);

Var

FileHandle: THandle;

Begin

FileHandle := FileCreate(FileName);

If FileHandle <> INVALID\_HANDLE\_VALUE Then

FileClose(FileHandle);

End;

Procedure TResultForm.LabelDeleteClick(Sender: TObject);

Var

Dialog: TForm;

DialogResult: Integer;

Begin

Dialog := CreateMessageDialog(' Are you sure??', MtConfirmation, [MbYes, MbNo], MbNo, ['Yes', 'No']);

Dialog.Caption := 'Delete';

If Dialog.ShowModal = MrYes Then

Begin

ClearFile(FileResults);

End

Else

Begin

End;

End;

Procedure TResultForm.LabelMainClick(Sender: TObject);

Begin

LastResFromForm := Label9.Caption;

Close;

End;

Procedure TResultForm.LabelMainMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

Light(Sender As TLabel);

End;

Procedure TResultForm.LabelMainMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Procedure TResultForm.StringGrid1DrawCell(Sender: TObject; ACol, ARow: LongInt; Rect: TRect; State: TGridDrawState);

Var

CellText: String;

Grid: TStringGrid;

Begin

Grid := StringGrid1;

Grid.Canvas.Brush.Color := ClWhite;

If ARow = 0 Then

Grid.Canvas.Brush.Color := ClBtnFace

Else

Begin

CellText := Grid.Cells[1, ARow];

If (ACol = 1) Then

Begin

If CellText = 'Correct' Then

Grid.Canvas.Brush.Color := ClGreen

Else

If CellText = 'Uncorrect' Then

Grid.Canvas.Brush.Color := ClRed

Else

If CellText = 'Not answer' Then

Grid.Canvas.Brush.Color := ClGray

Else

Grid.Canvas.Brush.Color := ClWhite;

End;

End;

Grid.Canvas.FillRect(Rect);

Grid.Canvas.TextOut(Rect.Left + 4, Rect.Top + 2, Grid.Cells[ACol, ARow]);

End;

End.

Unit UnitResultsFromFile;

Interface

Uses

System.SysUtils,

Vcl.Dialogs;

Type

PNode = ^TNode;

TNode = Record

Value: Integer;

Next: PNode;

Date: String;

End;

Var

Head: PNode = Nil;

Count: Integer;

LastValue: Integer;

LastResFromForm: String;

Arr: Array Of TNode;

Procedure ReadFromFile;

Procedure CountList;

Procedure ListToArray;

Procedure SortDescending;

Procedure FreeList;

Const

FileResults = 'GenResults.txt';

Implementation

Procedure ReadFromFile;

Const

Ch: Array [1 .. 1] Of Char = (' ');

Var

F: TextFile;

Line: String;

Num: Integer;

Tail, NewNode: PNode;

First, Second: String;

Begin

AssignFile(F, FileResults);

Reset(F);

Head := Nil;

Tail := Nil;

LastValue := 0;

While Not Eof(F) Do

Begin

ReadLn(F, Line);

If Length(Line.Split(Ch)) > 0 Then

Begin

First := Line.Split(Ch)[0];

Second := Line.Split(Ch)[1];

If TryStrToInt(Trim(First), Num) Then

Begin

LastValue := Num;

New(NewNode);

NewNode^.Value := Num;

NewNode^.Next := Nil;

NewNode^.Date := Second;

If Head = Nil Then

Begin

Head := NewNode;

Tail := NewNode;

End

Else

Begin

Tail^.Next := NewNode;

Tail := NewNode;

End;

End;

End;

End;

CloseFile(F);

End;

Procedure CountList;

Var

Current: PNode;

Begin

Count := 0;

Current := Head;

While Current <> Nil Do

Begin

Inc(Count);

Current := Current^.Next;

End;

End;

Procedure ListToArray;

Var

Current: PNode;

I: Integer;

Begin

SetLength(Arr, Count);

Current := Head;

I := 0;

While Current <> Nil Do

Begin

Arr[I] := Current^;

Inc(I);

Current := Current^.Next;

End;

End;

Procedure SortDescending;

Var

I, J: Integer;

Temp: TNode;

Begin

For I := 0 To High(Arr) - 1 Do

For J := I + 1 To High(Arr) Do

If Arr[I].Value < Arr[J].Value Then

Begin

Temp := Arr[I];

Arr[I] := Arr[J];

Arr[J] := Temp;

End;

End;

Procedure FreeList;

Var

Temp: PNode;

Begin

While Head <> Nil Do

Begin

Temp := Head;

Head := Head^.Next;

Dispose(Temp);

End;

End;

End.

Unit UnitSpeaking;

Interface

Uses

Winapi.Windows,

Winapi.Messages,

System.SysUtils,

System.Variants,

System.Classes,

Vcl.Graphics,

Vcl.Controls,

Vcl.Forms,

Vcl.Dialogs,

Vcl.ComCtrls,

Vcl.Imaging.Jpeg,

Vcl.ExtCtrls,

Vcl.StdCtrls,

Vcl.Imaging.Pngimage,

UnitQuestions,

UnitLight,

UnitCompereQuestions;

Type

TForm\_Speaking = Class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

Image1: TImage;

LabelNext: TLabel;

LabelMenu: TLabel;

Image2: TImage;

Image3: TImage;

LabelEnd: TLabel;

LabelBack: TLabel;

LabelTask1: TLabel;

RadioGroup1: TRadioGroup;

RadioGroup2: TRadioGroup;

RadioGroup3: TRadioGroup;

RadioGroup4: TRadioGroup;

RadioGroup5: TRadioGroup;

RadioGroup6: TRadioGroup;

RadioGroup7: TRadioGroup;

RadioGroup8: TRadioGroup;

RadioGroup9: TRadioGroup;

RadioGroup10: TRadioGroup;

RadioGroup11: TRadioGroup;

RadioGroup12: TRadioGroup;

RadioGroup13: TRadioGroup;

RadioGroup14: TRadioGroup;

RadioGroup15: TRadioGroup;

LabelTask: TLabel;

ComboBoxTask1: TComboBox;

ComboBoxTask2: TComboBox;

ComboBoxTask3: TComboBox;

ComboBoxTask4: TComboBox;

ComboBoxTask5: TComboBox;

ComboBoxTask6: TComboBox;

ComboBoxTask7: TComboBox;

ComboBoxTask8: TComboBox;

ComboBoxTask9: TComboBox;

ComboBoxTask10: TComboBox;

LabelIdiom1: TLabel;

LabelIdiom2: TLabel;

LabelIdiom3: TLabel;

LabelIdiom6: TLabel;

LabelIdiom8: TLabel;

LabelIdiom7: TLabel;

LabelIdiom5: TLabel;

LabelIdiom4: TLabel;

LabelIdiom9: TLabel;

LabelIdiom10: TLabel;

Procedure LabelMenuClick(Sender: TObject);

Procedure LabelNextClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure AddTestQuestions;

Procedure AddTestQuestions2;

Procedure LabelBackClick(Sender: TObject);

Procedure OpenTestQuestionsCom(FName: String; Var QuestionsWord1:

TQueueList);

Procedure LabelMenuMouseEnter(Sender: TObject);

Procedure LabelMenuMouseLeave(Sender: TObject);

Procedure FixUserAnswer1;

Procedure FixUserAnswer;

Procedure LabelEndClick(Sender: TObject);

Procedure FormClose(Sender: TObject; Var Action: TCloseAction);

Private

{ Private declarations }

Public

Questions: TQueueList;

Questions2: TQueueList;

AnswerFromFile: TQueueAnswerList;

CurrentAnwerTest: Integer;

{ Public declarations }

End;

Var

Form\_Speaking: TForm\_Speaking;

Const

File1 = 'communicative1.txt';

File2 = 'communicativeQuestions2.txt';

Implementation

{$R \*.dfm}

Procedure TForm\_Speaking.AddTestQuestions;

Var

I, J, K: Integer;

C: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

T := Questions.PHead;

J := 0;

For I := 1 To Questions.CountQ Do

Begin

C := FindComponent('RadioGroup' + IntToStr(I));

If (C Is TRadioGroup) Then

Begin

TRadioGroup(C).Items.Clear;

TRadioGroup(C).ItemIndex := -1;

TRadioGroup(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

AnswerF := T^.AnswerList.PHead;

While AnswerF <> Nil Do

Begin

TRadioGroup(C).Items.Add(AnswerF^.Answer);

AnswerF := AnswerF^.PNext;

End;

If Table1[1].Questions[J].Useranswer <> 'none' Then

Begin

For K := 0 To TRadioGroup(C).Items.Count - 1 Do

If TRadioGroup(C).Items[K] = Table1[1].Questions[J].Useranswer

Then

TRadioGroup(C).ItemIndex := K;

End;

Inc(J);

End;

T := T^.PNext;

End;

End;

Procedure TForm\_Speaking.OpenTestQuestionsCom(FName: String; Var QuestionsWord1: TQueueList);

Var

F: TextFile;

MainQuestion, Ans, Current: String;

ColAnswer, I: Integer;

TmpAnswerList: TQueueAnswerList;

Begin

PageControl1.ActivePage := TabSheet1;

Try

AssignFile(F, FName);

Reset(F);

Try

While Not EOF(F) Do

Begin

Readln(F, MainQuestion);

Readln(F, ColAnswer);

TmpAnswerList.PHead := Nil;

TmpAnswerList.PTail := Nil;

TmpAnswerList.Count := 0;

For I := 1 To ColAnswer Do

Begin

Readln(F, Ans);

AddAnswer(TmpAnswerList, Ans);

End;

Readln(F, Current);

AddAllAnswer(QuestionsWord1, TmpAnswerList, Current, MainQuestion);

End;

Except

End;

CloseFile(F);

Except

Showmessage('file is empty or not found');

Close;

End;

End;

Procedure TForm\_Speaking.AddTestQuestions2;

Var

I, J, K: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

Begin

T := Questions2.PHead;

J := 15;

For I := 1 To Questions2.CountQ Do

Begin

C := FindComponent('LabelIdiom' + IntToStr(I));

M := FindComponent('ComboBoxTask' + IntToStr(I));

If (C Is TLabel) Then

Begin

TLabel(C).Caption := IntToStr(J + 1) + ' ' + T^.Questions;

AnswerF := T^.AnswerList.PHead;

While AnswerF <> Nil Do

Begin

If (M Is TComboBox) Then

BEGIN

TComboBox(M).ItemIndex := -1;

TComboBox(M).Items.Add(AnswerF^.Answer);

AnswerF := AnswerF^.PNext;

END;

End;

If Table1[1].Questions[J].Useranswer <> 'none' Then

Begin

For K := 0 To TComboBox(M).Items.Count Do

If TComboBox(M).Items[K] = Table1[1].Questions[J].Useranswer Then

TComboBox(M).ItemIndex := K;

End;

Inc(J);

End;

T := T^.PNext;

End;

End;

Procedure TForm\_Speaking.FormClose(Sender: TObject; Var Action: TCloseAction);

Begin

FixUserAnswer1;

FixUserAnswer;

End;

Procedure TForm\_Speaking.FormCreate(Sender: TObject);

Begin

CurrentAnwerTest := 1;

OpenTestQuestionsCom(File1, Questions);

AddTestQuestions;

OpenTestQuestionsCom(File2, Questions2);

AddTestQuestions2

End;

Procedure TForm\_Speaking.LabelBackClick(Sender: TObject);

Begin

PageControl1.ActivePageIndex := PageControl1.ActivePageIndex - 1;

FixUserAnswer;

End;

Procedure TForm\_Speaking.LabelEndClick(Sender: TObject);

Begin

FixUserAnswer1;

FixUserAnswer;

Close;

End;

Procedure TForm\_Speaking.FixUserAnswer;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := Questions2.PHead;

CurrentIndex := 15;

For I := 1 To Questions2.CountQ Do

Begin

C := FindComponent('LabelIdiom' + IntToStr(I));

M := FindComponent('ComboBoxTask' + IntToStr(I));

If (C Is TLabel) Then

Begin

Table1[1].Questions[CurrentIndex].Question := T^.Questions;

Table1[1].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TComboBox(M).Text <> '' Then

Table1[1].Questions[CurrentIndex].Useranswer := TComboBox(M).Text;

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

Procedure TForm\_Speaking.FixUserAnswer1;

Var

I: Integer;

C, M: TComponent;

T: TPNodeList;

AnswerF: TPNodeAnswer;

CurrentIndex: Integer;

Begin

T := Questions.PHead;

CurrentIndex := 0;

For I := 1 To Questions.CountQ Do

Begin

M := FindComponent('RadioGroup' + IntToStr(I));

If (M Is TRadioGroup) Then

Begin

Table1[1].Questions[CurrentIndex].Question := T^.Questions;

Table1[1].Questions[CurrentIndex].Answer := T.CurrentAnswer;

If TRadioGroup(M).ItemIndex <> -1 Then

Table1[1].Questions[CurrentIndex].Useranswer := TRadioGroup(M).Items[TRadioGroup(M).ItemIndex];

Inc(CurrentIndex);

End;

T := T^.PNext;

End;

End;

Procedure TForm\_Speaking.LabelMenuClick(Sender: TObject);

Begin

FixUserAnswer1;

FixUserAnswer;

Close;

End;

Procedure TForm\_Speaking.LabelMenuMouseEnter(Sender: TObject);

Begin

If Sender Is TLabel Then

LightGrey(Sender As TLabel);

End;

Procedure TForm\_Speaking.LabelMenuMouseLeave(Sender: TObject);

Begin

If Sender Is TLabel Then

Dark(Sender As TLabel);

End;

Procedure TForm\_Speaking.LabelNextClick(Sender: TObject);

Begin

PageControl1.ActivePageIndex := PageControl1.ActivePageIndex + 1;

FixUserAnswer1;

FixUserAnswer;

End;

End.