**Разработка игры-тренажера по химии на языке программирования Delphi**

**Выполнила**:

Бутурлина Любовь Юрьевна

учитель химии и информатики

МОУ Индустринской ООШ

Коломна 2012

Содержание

Введение .……………………………......................................................... 3

Глава 1. Дидактические игры ….……………………….………………...6

1.1. Понятие и принцип дидактических игр..…………………….6

1.2. Примеры дидактических игр по химии.………..……………8

Глава 2. Среда программирования Delphi ……………..…….…….…... 16

2.1. Пользовательский интерфейс………………..…………….....16

2.2. Компоненты и палитра компонентов...….………….………..21

2.3. Структура кода модуля..…………………………………........22

2.4. Проект приложения. Файлы Delphi для приложения……….24

2.5. Обзор палитры компонент Standard и Additional

в Delphi………………………………………………………………..…....25

Глава 3. Создание игры-тренажера «Ядерные реакции» на языке программирования Delphi……………………………………………………….32

3.1. Оформление главного диалогового окна…………………….32

3.2. Оформление диалогового окна «Ядерные реакции»………..34

3.3. Оформление диалогового окна

«Инструкция к выполнению»…………………………………………….35

3.4. Обработка событий и написание программы………………..35

3.5. Создание проекта……………………………………………....37

3.6. Анализ и тестирование………………………………………...37

Заключение .……………………...…………………………………….......42

Список использованной литературы……..……..…………………….... ..44

Приложение…………………………………………………………………45

**Введение**

Современное состояние развития общества и социально-экономические преобразования в стране вызывают необходимость не только реформировать систему образования, но и совершенствовать методы, средства и приемы обучения, результаты которых, с одной стороны, удовлетворяли бы требованиям общества, а, с другой стороны, соответствовали бы образовательным запросам учащихся, их склонностям, интересам и возможностям.

Возрастающий объем информации, непрерывное пополнение и обновление содержания школьных предметов, перегруженность некоторых новых учебников и программ теоретическим материалом требуют от школьников усвоения все большего количества теорий, законов и понятий. В то же время многочисленные наблюдения и исследования показывают, что интерес учащихся к изучаемым предметам, в частности к химии, падает.

Таким образом, возникает противоречие между возрастающими требованиями общества к уровню знаний выпускников общеобразовательных школ и возможностью, а порой и желанием учащихся следовать этим требованиям. Основная тенденция развития этого противоречия заключается в том, что требования общества к процессу обучения постоянно расширяются, а результаты обучения отстают от этих требований. В определенной степени это противоречие вызвано тем, что зачастую ученики не проявляют интереса, желания, стремления к учению и познанию чего-то нового. У многих школьников отсутствуют мотив обучения, потребность к самосовершенствованию, слабо сформированы компоненты учебной деятельности.

В связи с этим в педагогике и частных методиках обучения возникает проблема разработки методов, средств и приемов более эффективного и плодотворного учебно-воспитательного процесса, одновременно не допускающих перегрузок учащихся, которые не только вызывают отрицательное отношение к учебе, но и пагубно сказываются на здоровье детей.

Эту проблему отчасти можно решить, используя в процессе обучения, наряду с традиционными методами и приемами, такое средство обучения, как дидактическая игра. Однако, если игра в учебно-воспитательном процессе детей дошкольного и младшего школьного возраста применятся довольно широко и изучена в значительной степени, то потенциал дидактической игры в обучении и воспитании школьников среднего и старшего возраста мало изучен, и поэтому используется ограниченно.

Как показали исследования, игровой метод на уроках химии используется учителями лишь эпизодически. Более часто игры применяются при проведении внеклассных мероприятий. Это вызвано отсутствием в методической литературе разработанной системы составления и применения дидактических игр, включающей:

а) алгоритм составления дидактических игр;

б) методические рекомендации по их организации и проведению;

в) комплект дидактических игр, соответствующий современному построению и содержанию школьных учебников и программ, позволяющий сделать учителю оптимальный выбор игры в зависимости от темы и задач урока.

Интерес к проблеме повышения эффективности обучения средствами дидактической игры вызван также противоречиями, которые присущи сегодня процессу обучения химии в средней школе: между современными психолого-педагогическими учениями и теориями развивающего обучения и слабой разработанностью методических основ их реализации через школьные предметы, в частности в обучении химии; между необходимостью реализации учебно-воспитательных задач и одновременным увеличением содержательной базы курса химии средней школы и сокращением учебного времени для их решения.

Вышеуказанные факторы обусловили ***актуальность***данной темы, так как, на мой взгляд, средствами дидактической игры действие этих противоречий можно ослабить, а эффективность обучения - повысить.

***Цель исследования*** - разработать дидактическую компьютерную игру по химии.

***Объектом исследования*** является программирование на языке Delphi.

***Предмет исследования*** – разработка программ создания мини-игр на языке программирования Delphi.

***Задачи исследования:***

1. Изучение роли игры в процессе обучения.
2. Рассмотрение необходимости применения дидактических игр на уроках для устранения психологических барьеров.
3. Выявление форм и способов игровой деятельности, влияющих на тот или иной аспект процесса познания.

4) Написать программу создания игры-тренажера «Ядерные реакции» на языке программирования Delphi, которая, во-первых, генерирует ячейки ядерной реакции с некоторыми пустыми клетками; во-вторых, проверяет, является ли написанная реакция ядерной.

**Глава 1. ДИДАКТИЧЕСКИЕ ИГРЫ**

**1.1. Понятие и принцип дидактических игр**

Особое место среди многообразия различных игр занимает дидактическая игра как специальный метод обучения, методический прием или средство обучения в виде игровой ситуации, направленные на достижение определенных дидактических целей в процессе обучения. Дидактические игры возбуждают интерес учащихся к химии, активизируют их на протяжении всего урока, развивают мышление, речь, память, формируют познавательные интересы, оказывают положительное воспитывающее действие. Дидактическая игра является действенным методическим приемом, позволяющим включать ученика в активную творческую учебную деятельность. Дидактические игры можно подразделить на обучающие, развивающие, воспитывающие и комбинированные, позволяющие решать одновременно несколько дидактических задач. Более широкое применение в процессе обучения имеют комбинированные игры, помогающих решению одновременно нескольких дидактических задач урока. Применение дидактических игр должно сопровождаться соблюдением следующих правил:

1. Дидактические игры следует использовать с целью получения определенного результата в процессе обучения и воспитания. К дидактическим играм следует готовить учеников заранее, учитывая, психолого-педагогические аспекты, время проведения, атрибуты и правила.

2. В названии, правилах и сюжете игры должны преобладать элементы химической занимательности и соревновательности.

3. Дидактический фактор игры должен обеспечить достаточно широкую, но доступную для учеников творческую и мыслительную деятельность в области химии и смежных дисциплин, соответствовать возрастным особенностям, степени подготовки и развитию кругозора учащихся.

4. Атрибуты игры должны быть красивыми, яркими, загадочными, необыкновенными. Они должны иметь характер многоразового использования.

5. Правила игры должны быть простыми и однозначно сформулированными.

6. Во время игры необходимы дисциплина и порядок. При необходимости определяют специальное жюри или помощников-тьютеров.

7. Игры лучше всего проводить в быстром темпе, чтобы каждый ученик с максимальной активностью мог участвовать в игре.

8. Учителю следует быть активным со-участником игры, не допуская при этом снижения интереса, субъективности и нечестности. Можно использовать поощрительные и штрафные баллы.

9. Подведение результатов обеспечивает участникам получения морального удовлетворения.

10. Для каждой игры должны быть определены красочные и эффектные призы.

11. Дидактический эффект усиливается, если ученики принимают участие в подготовке атрибутов и призов, а также в разработке правил игры.

**Последовательность составления дидактической игры**

|  |  |
| --- | --- |
| 1) Определение дидактической цели игры. | *- Вытекает из содержания изучаемой темы.* |
| 2) Выбор формы и названия дидактической игры, разработка ее содержания и правил. | *- Определяется возрастным фактором учащихся, их интересами на момент игры.* |
| 3) Изготовление атрибутики, подготовка вспомогательных средств игры. | *- Зависит от содержания игры и подготовленности учащихся.* |
| 4) Уточнение оптимального соотношения дидактических функций игры и ее занимательности. | *- Элементы занимательности не должны снижать дидактическую цель.* |
| 5) Уточнение правил и окончательной модели игры от начала и до конца. | *- Определяется временем, отводимым на игру, целями игры, контингентом учащихся.* |
| 6) Разработка системы подведения итогов игры. | *- Вытекают из содержания игры, материальной базы, возраста.* |

В свою очередь обучающие игры по химии можно подразделить на: игры, формирующие новые понятия; игры, систематизирующие изучаемый материал; игры, используемые при контроле и учете знаний. Среди таких игр следует выделить: игры, развивающие свойства личности (память, внимание, мышление и др.); игры, формирующие практические умения и навыки. Воспитывающие игры чаще всего находят место при контроле домашнего задания, закреплении изученного материала, а также при проведении практических и лабораторных занятий, но особый результат с их помощью достигается в процессе подготовки и проведения уроков-семинаров, зачетных занятий, при проведении внеклассных соревновательных мероприятий по предмету.

**1.2. Примеры дидактических игр по химии**

Известно, что учащимся на занятиях приходится давать большое количество информации при ограниченном количестве времени. При этом нередко требуется особое внимание уделять межпредметным связям химии с предметами специального цикла, что, в свою очередь, ещё больше увеличивает их учебную нагрузку.

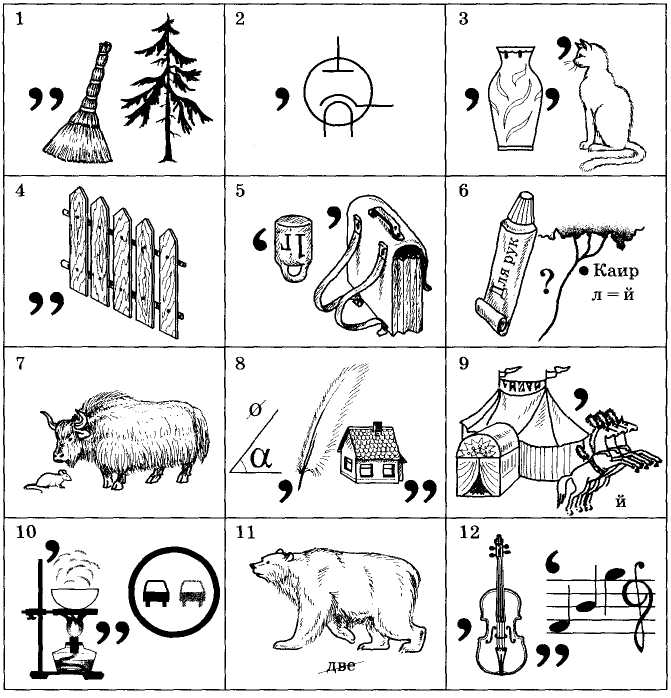
Снижению перегрузки обучаемых, снятию утомления и психологического дискомфорта, повышению эмоционального настроя на дальнейшее восприятие материала помогает игровая деятельность. Игра используется как форма проведения урока или его отдельных частей (при закреплении материала даются дополнительные упражнения). По отдельным темам учащимся предлагаются кроссворды, ребусы, головоломки. При их решении они реализуют свои знания о межпредметных связях, экологии и др. Эти игры развивают внимание, тренируют память, способствуют более глубокому усвоению и закреплению полученных знаний, регулируют смену вида деятельности учащихся, вызывают интерес их к изучаемой проблеме.

В данном разделе представлены некоторые игровые формы, применяемые на уроках химии. Игровая форма урока сочетает в себе обучающие, развивающие и воспитательные элементы образовательного процесса.

*1. «Металлы и неметаллы»*

В этих маленьких ребусах зашифрованы названия двенадцати химических элементов. Расшифруйте их и установите, какие из этих элементов образуют простые вещества в виде металлов, а какие - в виде неметаллов. Есть ли среди данных элементов такие, которые образуют аллотропные видоизменения? Какие из этих веществ находятся при обычных условиях в газообразном состоянии?

Какой из этих элементов был известен человечеству с давних времен, а какой из них был открыт последним?



*2) Краткий "энциклопедический" словарь*

Наш словарик необычный. Определяемые слова зашифрованы в виде буквенных ребусов. Однако и буквы здесь не обычные, а знаки химических элементов.

**1,Э,2,~~3~~,4**

**Ar, ,At -** летательный аппарат, наполняемый газом легче воздуха. (Аэростат)

**B Ne, -** остров в Малайском архипелаге. (Борнео).

**,,Au -** настольная игра, в которой особыми фишками закрываются карточки с номерами или картинками. (Лото)

**M=В**

**Cu, Cu -** крупное хищное животное класса млекопитающих. (Медведь)

**As,,** -  животное из отряда грызунов. (Мышь)

**6,5 7,2**

**Ge Se Ge** - жители страны, местности, города. (Население)

**a=o**

**Pt -** гидротехническое сооружение на реке или другом водоеме, предназначенное для подъема уровня воды. (Плотина)

**Rn; Ar; -** радиолокационная установка. (Радар)

**,,Ag -** отрезок, лежащий на пересечении граней геометрического тела. (Ребро)

**,Xe, -** корм для домашних животных в виде высушенной травы. (Сено)

**1=C**

**Sn -** структурная единица языка, служащая для обозначения понятий, предметов, действий. (Слово)

**1,2,4**

**U, Hf -** ветер разрушительной силы. (Ураган)

*3) Игра "Отгадай задуманный элемент"*

Цель. Заинтересовать учащихся, повысить уровень их внимания к изучаемому материалу. Развить внимание, наблюдательность и умение ориентироваться в периодической системе химических элементов Д.И. Менделеева.

Атрибуты. Периодическая система Д.И. Менделеева и калькуляторы.

Описание игры. Ведущий просит одного из учеников задумать любой химический элемент по периодической системе. После этого ведущий предлагает провести с номером этого элемента следующие вычисления без сообщения промежуточных результатов.

1) Номер элемента удвоить.

2) К произведению прибавить 5.

3) Сумму умножить на 5.

Последний результат сообщается ведущему, который тотчас объяв­ляет задуманный играющим элемент.

Объяснение игры. Разгадка заключается в следующем. Пусть задуман элемент №25 /марганец/. Проведем с числом 25 соответствующие математические действия:

25 ⋅ 2=50; 50+5=55; 55 ⋅ 5 = 275.

Число 275 сообщается ведущему, который в уме отбрасывает послед­нюю цифру /получается 27/ и отнимает от полученного числа число 2 - получается 25. Это и есть номер задуманного элемента. После этого ведущему остается только назвать этот элемент - марганец.

4) *«Химия – это страна чудес»*

Для этого Фокуса потребуется тридцать небольших карточек из плотной бумаги. На одной стороне этих картотек пишутся знаки и названия различных химических элементов, например, начиная от водорода до цинка по периодической системе элементов. Ведущий просит одного из зрителей назвать любое число от 1 до 10. Предположим, что названо число 4. Ведущий убирает из стопки любые четыре карточки, оставшиеся отдает зрителю и просит их пересчитать и определить в полученном числе сумму цифр. Далее ведущий предлагает зрителю запомнить ту карточку снизу, номер которой равен этой сумме. Затем он забирает все карточки и начинает откладывать их сверху по одной, произнося при этом по буквам фразу: «Х-и-м-и-я — э-т-о с-т-р-а-н-а ч-у-д-е-с» так, чтобы на каждую произнесенную букву откладывалась одна карточка. Карточка, которая выпадает при произнесении последней буквы, переворачивается надписью вверх. Это и будет та карточка, которую запомнил зритель.

*5) Игра «Магические элементы»*

Цель. Активизировать процесс запоминания названий химических элементов.

Атрибуты. Периодическая система химических элементов Д.И. Менделеева.

ВАРИАНТ I. Ведущий называет порядковый номер какого-либо химического элемента, в названии которого отсутствовали бы буквы Ь, Й, Я и т.д. /нет названий химических элементов, которые начинались бы с этих букв/.

Задание. Играющие записывают название данного химического элемента по вертикальной линии, а рядом - названия химических элементов, начинающиеся с соответствующей буквы из первого слова. Выигрывает тот, кто первым справит­ся с заданием.

*Пример.* ЭЛЕМЕНТ НОМЕР 86

**Р** *А Д И Й*

**А** *С Т А Т*

**Д** *И С П Р О З И Й*

**О** *Л О В О*

**Н** *А Т Р И Й*

ВАРИАНТ II. Ведущий называет номер химического элемента или его индивидуальные признаки. Играющие записывают его название в таблицу так, чтобы в каждой верхней графе находилось по одной букве, ниже записывают столько слов, сколько их в исходном слове. Эти слова должны начинаться с соответствующей буквы и быть назва­ниями химических элементов. Если окажется, что на данную букву нет достаточного числа названий химических элементов, то их заменяют другими химическими терминами.

**Пример.** ЭЛЕМЕНТ НОМЕР 22

**Т И Т А Н**

тантал индий тербий аргон натрий

технеций иридий торий астат никель

таллий иод тигель азот неон

теллур иттербий теплота актиний неодим

туллий иттрий торф алюминий ниобий

*6) Игра "Химическая тайнопись"*

Цель. Облегчить запоминание названий химических элементов и их символов.

Атрибуты игры. Карточки с названиями химических элементов и их символами.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  |  |  |  |  |  |  |
| **НАТРИЙ** | |  | | **I** | |  | |  |  | **ЛИТИЙ** |  | **Ag** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |
| **МАРГАНЕЦ** | |  | | **Na** | |  | |  |  | **БРОМ** |  | **Li** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |
| **ИОД** | |  | | **Cu** | |  | |  |  | **ХРОМ** |  | **S** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |
| **СВИНЕЦ** | |  | | **Au** | |  | |  |  | **СЕРЕБРО** |  | **Cr** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |
| **ЗОЛОТО** | |  | | **Pb** | |  | |  |  | **СЕРА** |  | **Zn** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |
| **МЕДЬ** | |  | | **Mn** | |  | |  |  | **ЦИНК** |  | **Br** |  |
|  | |  | |  | |  | |  |  |  |  |  |  |

Задание. Как можно быстрее соедините линией прямоугольники /название элемента/ с соответствующими прямоугольниками /химический знак/.

Чтобы обеспечить многократное использование карточек, можно линии проводить на листочках полиэтиленовой пленки соответствую­щего размера.

*7) Игра «Узнай-ка!»*

Цель. Развить внимание, наблюдательность, зрительную память и закрепить знания учащихся о лабораторной посуде.

Атрибуты. Пластмассовая воронка; стеклянная палочка (с оплавленными концами); тигельные щипцы; фарфоровая чашка; фарфоровый пестик; держатель для пробирок; спиртовка, не наполненная спиртом; штатив для пробирок; другие безопасные предметы из числа химического оборудования; платок для завязывания глаз.

Задание. На столе раскладываются предметы, указанные выше. Игрок в течение 5 секунд запоминает их, и затем ему завязывают глаза. Ведущий быстро меняет местами предметы. Участник игры с завязанными глазами поворачивается по команде и берет любой из предметов, лежащих на столе. Его задача – узнать предмет по его внешним признакам, дать название и рассказать для чего он используется в химии.

**Глава 2. СРЕДА ПРОГРАММИРОВАНИЯ DELPHI 7**

**2.1.Пользовательский интерфейс**

Для запуска среды *Delphi* выполните следующую команду:

*Пуск → Все программы → Borland Delphi 7 → Delphi 7*

После загрузки интерфейс *Delphi 7* имеет вид, показанный на *рис. 1*, и имеет 5 основных окон:

1. главное окно (Delphi 7 - Project 1);
2. окно редактора свойств объектов (Object Inspector)**;**
3. окно формы (Form 1);
4. окно редактора кода (Unit1.pas);
5. окно дерева объектов (Object TreeView).

Окно редактора кода почти полностью закрыто окном стартовой формы. Для переключения между окном формы и окном редактора кода нажмите клавишу F12.

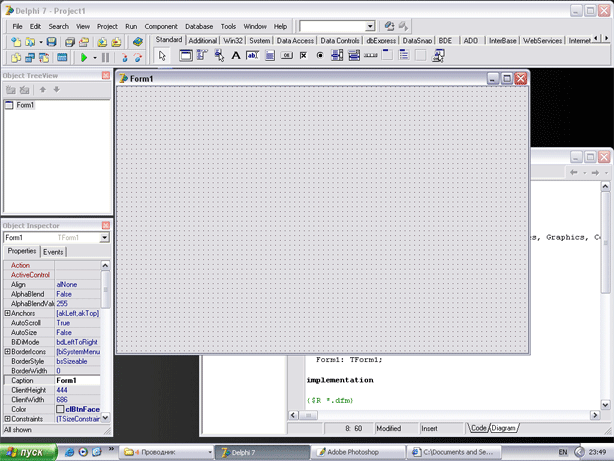


Рис. 1. Интерфейс Delphi 7

**Главное окно** находится в верхней части экрана. В нём расположены:

- строка заголовка;

- строка меню;

- панель инструментов;

- панель палитры компонентов.

Вид главного окна представлен на рис. 2.

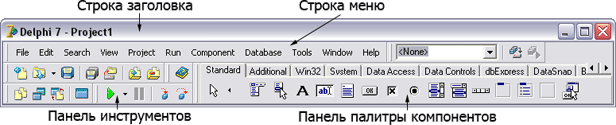


Рис. 2. Главное окно

**Object Inspector –** окно редактора свойств объектов или инспектор объектов (рис. 3).

Любой размещаемый на форме компонент характеризуется некоторым набором параметров: положением, размером, цветом и т. д. Часть этих параметров, например, положение и размеры компонента, можно изменять, манипулируя с компонентом в окне формы. Для изменения других параметров предназначено окно редактора свойств объектов (или, что то же самое, Инспектора объектов). Это окно содержит две вкладки – *Properties* (Свойства) и *Events* (События). Вкладка properties служит для установки нужных свойств компонента, вкладка Events позволяет определить реакцию компонента на то или иное событие. Совокупность cвойств отображает видимую сторону компонента: положение относительно левого верхнего угла рабочей области формы, его размеры и цвет, шрифт и текст надписи на нем и т. п.; совокупность событий – его поведенческую сторону: будет ли компонент реагировать на щелчок мыши или на нажатие клавиш, как он будет вести себя в момент появления на экране или в момент изменения размеров окна и т. п.

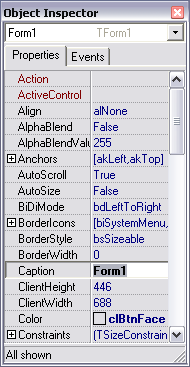


Рис. 3. Окно редактора свойств объектов (инспектор объектов)

**Object TreeView** – окно дерева объектов (рис. 4) предназначено для наглядного отображения связей между отдельными компонентами, размещенными на активной форме или в активном модуле данных. Щелчок по любому компоненту в этом окне активизирует соответствующий компонент в окне формы и отображает свойства этого компонента в окне Инспектора объектов.

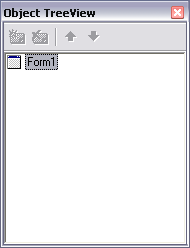


Рис. 4. Окно дерева объектов

**Окно кода программы** представлено на рис. 5.

Окно редактора кода, можно увидеть, отодвинув в сторону окно формы либо, как уже говорилось выше, нажать клавишу F12. В него следует набирать текст программы. В начале работы над новым проектом это окно редактора кода содержит сформированный Delphi шаблон программы.

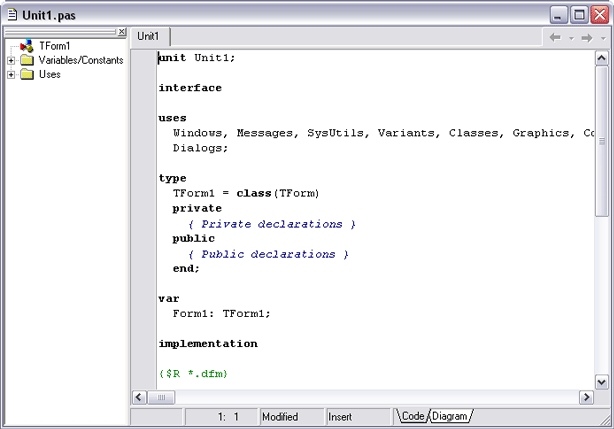


Рис. 5. Окно кода программы

**Окно формы** представляет собой проект Windows-окна будущей программы. Вначале это окно содержит стандартные для Windows интерфейсные элементы - кнопки вызова системного меню, максимизации, минимизации и закрытия окна, строку заголовка и очерчивающую рамку. Вся рабочая область окна заполнена точками координатной сетки, служащей для упорядочения размещаемых на форме компонентов.

Свойства формы определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки.

 В ходе работы над новым проектом*,* так в Delphi называется разрабатываемое приложение, программист изменяет значения свойств формы *Form1*и добавляет к форме необходимые компоненты (поля ввода и вывода текста, командные кнопки), при этом он в любой момент времени контролирует содержание окна создаваемой программы и может внести в него необходимые изменения.

**2.2. Компоненты и палитра компонентов**

*Палитра компонентов* – это каталог, состоящий из визуальных и невизуальных компонентов. *Компонент* – это структурная единица Delphi. Основу ООП в Delphi составляет набор компонентов, который позволяет Delphi с помощью компонентов использовать множество возможностей, присущих Windows.

Окно формы – это окно Windows. Всё рабочее пространство окна – это рабочая область, размеченная сеткой для удобства расположения на ней компонентов из Палитры компонентов. Сама форма также является компонентом. Новая форма, которая создаётся при загрузке Delphi или при создании нового проекта, является главной формой приложения.

Палитра компонентов расположена в правой части главного окна и имеет вид многостраничного блокнота, где на каждой странице размещён набор пиктограмм её компонентов. Активизировать группу компонентов требуемой страницы надо щелчком мыши на её закладке.

Для добавления какого-либо компонента в Окно формы надо выбрать его пиктограмму на панели компонентов, щёлкнуть на нём левой кнопкой мыши, а затем возможны два варианта действий:

- если надо точно позиционировать компонент, надо подвести курсор мыши в нужное место Окна формы и щёлкнуть левой кнопкой мыши один раз; левый верхний угол компонента совпадёт при этом с положением конца стрелки курсора мыши;

- если надо поместить компонент в центр Окна формы, используется двойной щелчок на пиктограмме.

**2.3. Структура кода модуля**

 Нажмите клавишу F12 для просмотра кода программы. Можно заметить, что в окне кода программы на вкладке Unit1 уже существует некоторый код, сформированный Delphi. Этот код выглядит следующим образом:

**unit** Unit1;

**interface**  *// раздел интерфейса*

*{ Здесь находятся описания процедур и функций модуля, которые могут использоваться другими модулями}*

*{ Список подключаемых модулей}*

**uses**

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs;

**type** *// раздел объявления типов*

 TForm1 = **class**(TForm)

 private *// Закрытый раздел класса*

*{ Private declarations }*

*{ Сюда могут помещаться объявления переменных, функций и процедур, включаемых в класс формы, но не доступных для других модулей}*

**public***// Открытый раздел класса*

*{ Public declarations }*

*{ Сюда могут помещаться объявления переменных, функций и процедур, включаемых в класс формы и доступных для других модулей}*

**end**;

**var** *// раздел объявления переменных*

*{ Здесь находятся объявления глобальных переменных модуля, которые могут использоваться процедурами и функциями модуля }*

   Form1: TForm1;

**implementation** *// раздел реализации*

*{Здесь находятся описания (текст) процедур и функций модуля. Сюда могут помещаться предложения* ***uses****, объявления типов, констант, переменных, к которым не будет доступа из других модулей. Тут же должны быть реализации всех объявленных в разделе* ***interface*** *функций и процедур, а также могут быть реализации любых дополнительных, не объявленных ранее функций и процедур}*

*{$R \*.dfm}*

**end**.

Модуль состоит из последовательности разделов. Каждый раздел начинается ключевым словом и продолжается до начала следующего раздела.

 Модуль начинается с ключевого слова **unit,** после которого пишется имя модуля. Оно совпадает с именем файла, в которым вы сохранили свой модуль. По умолчанию для первого модуля имя равно **Unit1**, для второго **Unit2** — и т.д.

 Текст модуля состоит из двух основных разделов: **interface** — *открытый интерфейс* модуля, и **implementation** — *реализация* модуля. Все, что помещается непосредственно в раздел **interface** (типы, переменные, константы, функции, процедуры), может быть использовано другими модулями программы. Все, что помещается в раздел **implementation** — внутреннее дело модуля. Внешние модули не могут видеть типы, переменные, константы, функции и процедуры, размещенные в разделе реализации.

 В разделе **interface** после предложения **uses**, содержащего список подключаемых библиотечных модулей, вы можете видеть объявление класса вашей формы, подготовленное Delphi. Имя класса вашей формы — **TForm1**. В класс включены те объекты которые размещены на форме

 В классе предусмотрено также два раздела: **private** — *закрытый* раздел класса, и **public** — *открытый* раздел класса. То, что вы или Delphi объявите в разделе **public**, будет доступно для других классов и модулей. То, что объявлено в разделе **private**, доступно только в пределах данного модуля.

**2.4. Проект приложения. Файлы Delphi для приложения**

 Проект Delphi состоит из форм, модулей, установок параметров проекта, его ресурсов (битовые файлы, пиктограммы) и т.д. На этапе проектирования Delphi создаёт совокупность файлов, из которых состоит приложение. Результатом проектирования приложения, разработанного в Delphi, является EXE-файл для выполнения в среде Windows. Он формируется на базе проекта приложения. Один проект соответствует одному приложению.

Проект должен иметь определённую структуру каталогов для хранения файлов проекта. Имена файлов, форм, компонентов и переменных должны отражать их содержание. *Проект* – это набор взаимосвязанных форм и модулей, образующих приложение. Модули проекта используют другие модули, определённые их оператором Uses.

В состав проекта входят следующие файлы:

1. файл проекта (расширение .DPR – Delphi PRoject); в проекте приложения он единственный;
2. описание всех форм, входящих в проект; каждая форма проекта имеет файл формы (расширение DFM – от Delphi ForM);
3. модули форм (расширение PAS); файл создаётся автоматически для каждой формы проекта;
4. файлы с параметрами проекта (расширение DOF, от Delphi Option File);
5. файлы с описаниями ресурсов (расширение RES, от Delphi Component RESourse).

# 2.5. Обзор палитры компонент - Standard, Additional в Delphi

Чтобы создать интерфейс программы, нужно иметь выбор из различных компонент. В Windows существует множество элементов, и все они доступны в [Delphi](http://www.itshop.ru/Embarcadero/Technologies/CodeGear/from/Borland/l2t1f3c0sc0) для использования. Познакомимся с двумя первыми вкладками *Палитры Компонент -* ***Standard*** *и* ***Additional***. Именно на них расположено большинство элементов, наиболее часто используемых в программах.

## ***Standard***

## Вкладка Standard палитры компонент

Рис. 6. Окно вкладки Standard

Из названия вкладки следует, что компоненты, представленные на ней, являются стандартными, системными. Так и есть.

Frames - позволяет разместить на форме так называемый " фрейм ". Фрейм из себя представляет другое окно. Чтобы создать окно-фрейм, следует выбрать пункт меню ***File - New - Frame***, либо выделить значок Frame в окне ***File - New - Other*** на вкладке *New* . До тех пор, пока в приложении не будет ни одного фрейма, использовать данный объект не удастся. Фреймы удобны в том случае, когда какие-либо настройки запрашиваются во время работы программы в виде отдельной формы, а также, например, на одной из вкладок основной формы.

*MainMenu* - основное меню окна. Связать его с формой можно через свойство ***Menu*** формы. При двойном щелчке по значку *MenuMenu* появляется дизайнер меню, в котором можно добавлять и удалять пункты. В ***Caption*** задаётся текст пунта меню. Чтобы создать черту - разделитель, следует в свойство ***Caption*** прописать знак " минус " (" - ", без кавычек).

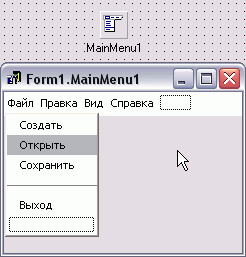


Рис. 7. Компонент MainMenu

*PopupMenu* - контекстное меню (вызывается правой кнопкой мыши). Его можно привязать ко многим компонентам (как правило, это делается через свойство ***PopupMenu*** у компонента, которому ассоциируется это меню). Одно и то же меню может быть привязано к нескольким компонентам.

*Label* - текстовая метка (надпись) на форме. Используется для отображения любого текста в окне. Текст задаётся в свойстве ***Caption***. Свойство ***Font*** позволяет настроить шрифт текста.

*Edit* - поле ввода. Используется для ввода любых данных (текста, числа и т.д.), представленных одной строкой. Свойство ***ReadOnly*** позволяет запретить редактирование текста в поле. Текст хранится свойством ***Text.*** Свойство ***MaxLength*** позволяет задать максимальное число символов, которое может быть введено в поле. Значение 0 означает, что ограничение не установлено.

Memo - многострочный ***Edit.*** Используется для ввода больших объёмов текста. Свойство ***ScrollBars*** позволяет указать полосы прокрутки, которые будут отображаться у поля при недостатке места для всего текста:

* *ssNone* - без полос прокрутки;
* *ssHorizontal -* горизонтальная полоса прокрутки;
* *ssVertical* - вертикальная;
* *ssBoth* - обе: и горизонтальная, и вертикальная.

*Button* - кнопка, самая что ни есть обыкновенная. ***Caption*** - текст на кнопке.

*CheckBox* - флажок-переключатель. Состояние хранится в свойстве ***Style:***

* *cbUnchecked* - не отмечен;
* *cbChecked* - отмечен;
* *cbGrayed* - затемнён.

Само состояние следует изменять с помощью свойства ***Checked.*** Таких флажков может быть несколько и каждый может быть отмечен независимо от остальных.

*RadioButton* - радио-переключатель. Аналогичен *CheckBox*, но при наличии нескольких переключателей может быть выбран только один из них. Свойство ***Checked*** позволяет изменить состояние переключателя.

*ListBox* - список текстовых строк. Используется для выбора одного из вариантов. Строки задаются в свойстве ***Items***. Чтобы изменить исходные значения, следует щёлкнуть по кнопке с тремя точкам (" ... ") в строке свойства ***Items*** и в открывшемся редакторе ввести требуемые значения. Свойство ***MultiSelect*** позволяет включить/выключить одновременный выбор нескольких строк из списка.

*ComboBox* - ещё один список для выбора, но выпадающий - на экране видна всего одна строка, а сам список появляется при нажатии на кнопку со стрелкой. Установив свойство ***Style*** в ***csDropDownList*** можно запретить ввод произвольного значения в ComboBox . Текст выбранной строки доступен в свойстве ***Text.***

*ScrollBar* - полоса прокрутки. Свойство ***Kind*** определяет ориентацию полосы на форме: *sbHorizontal* - горизонтально, *sbVertical* - вертикально.

*GroupBox* - "контейнер" для компонент. Используется для объединения элементов в отдельные области. При перемещении *GroupBox'а* все компоненты, расположенные на нём, также перемещаются. Для размещения компонент в этом контейнере следует после выбора их на Палитры компонент щёлкнуть по самому контейнеру (а не по форме, как обычно). У *GroupBox* можно задать текст заголовка (свойство ***Caption***).

*RadioGroup* - группа из нескольких *RadioButton'ов*. Тексты строк задаются в свойстве ***Items***. Номер выбранной строки - свойство ***ItemIndex*** (нумерация строк начинается с нуля!).

*Panel* - практически такой же контейнер, что и *GroupBox*, однако без заголовка и с возможностью отключения рельефных границ.

*ActionList* - позволяет управлять действиями (*Actions)*, которые привязываются к пунктам меню, кнопкам и т.д.

## ***Additional***Вкладка Additional палитры компонент

Рис. 8. Окно вкладки Additional

Компоненты с этой вкладки используются реже, но в некоторых случаях без них не обойтись. Среди них есть как компоненты для создания украшений, так и компоненты-модификации стандартных компонент.

*BitBtn* - кнопка, на которой помимо текста можно разместить изображение. За изображение отвечает свойство ***Glyph***. Формат - \*.bmp . Чтобы добавить картинку, следует щёлкнуть по кнопке с тремя точками в строке свойства ***Glyph***, в открывшемся окне нажать ***Load...*** и указать файл с картинкой.

*SpeedButton* - ещё один тип кнопки. Также поддерживает добавление изображения. Помимо этого кнопку можно сделать "плоской", т.е. без границ - за это отвечает свойство ***Flat****.*

*MaskEdit* - аналог *Edit* , но используется для ввода текста по маске. Например, можно указать ввод только цифр. Маска задаётся в свойстве ***EditMask.*** Имеется несколько стандартных вариантов.

*StringGrid* - таблица, в каждой ячейке которой может быть расположен текст. Часть ячеек можно сделать фиксированными (свойства ***FixedCols*** и ***FixedRows***). Количество строк и столбцов таблицы задаётся соответственно свойствами ***RowCount*** и ***ColCount.***

*DrawGrid* - таблица, но в ячейках помимо текста могут располагаться и другие объекты, например изображения.

*Image* - изображение, которое можно разместить на форме. Изображение задаётся в свойстве ***Picture.*** Свойство ***Center*** позволяет отцентрировать изображение, если размеров недостаточно для его полного отображения. ***Stretch*** позволяет задать сжатие/растяжение картинки под размеры компонента, ***Proportional*** указывает, следует ли при этом сохранять пропорции исходного изображения.

*Shape* - один из вариантов геометрических фигур. Тип фигуры определяется в свойстве ***Shape:***

* *stCircle* - окружность;
* *stEllipse* - эллипс;
* *stRectangle* - прямоугольник;
* *stRoundRect* - прямоугольник с закруглёнными углами;
* *stRoundSquare* - квадрат с закруглёнными углами;
* *stSquare* - квадрат.

Свойства ***Pen*** и ***Brush*** позволяют задать стиль границ фигуры и её внутреннюю заливку.

*Bevel* - компонент для создания рельефа на форме - линий, окошек и т.д.

*ScrollBox* - прокручиваемая область, на которой можно разместить другие элементы.

*CheckListBox* - аналог *ListBox* , но к каждой строке добавляется флажок. Таким образом более удобно выбирать значения из списка.

*Splitter* - разделитель для создания областей изменяемых размеров. Компонент следует разместить между двумя элементами, которые должны быть изменяемого размера. После этого следует настроить выравнивание (*Align* ) элементов и *Splitter* автоматически начнёт работать.

*StaticText* - текстовая метка, аналог *Label* , но в отличие от него является полноценным элементом управления *Windows.* Использовать *StaticText* имеет смысл в том случае, если нужно разместить текст поверх какого-либо другого компонента. С *Label* этого сделать не удастся.

*ControlBar* - один из типов панелей инструментов. Автоматически перетаскивается по области, отведённой для панелей инструментов.

*ApplicationEvents* - используется для доступа к некоторым свойствам и событиям объекта ***TApplication***. Этот объект - и есть само приложение. Работать с этим объектом можно и программно, но с помощью этого компонента всё же удобнее. Он является невизуальным.

*ValueListEditor* - таблица из двух колонок - поля и значения. Некое подобие *StringGrid* .

*LabeledEdit - Label* и *Edit* "в одном флаконе". Оба компонента являются полностью настраиваемыми.

*ColorBox* - выпадающий список для выбора цвета.

*Chart* - мощный объект для построения диаграмм и графиков.

*ActionManager* - используется для управления действиями *Taction.*. Аналогичен *ActionList* со страницы *Standard*, но имеет намного больше возможностей. При создании приложения на основе этой технологии рекомендуется использовать именно *ActionManager* .

*ActionMainMenuBar* - меню, работающее на основе *TAction* .

*ActionToolBar* - аналогично, панель инструментов на основе *TAction* .

*XPColorMap, StandardColorMap, TwilightColorMap* - стандартные цветовые схемы для объектов на основе *TAction* (кстати, они называются *Action-band* компонентами ).

*CustomizeDlg* - диалог для настройки *Action-band* компонентов. Такой диалог используется, например в Microsoft Word для индивидуальной настройки панелей инструментов и меню. Помимо этого, *Action-band* компоненты позволяют сохранять и восстанавливать состояние всех объектов. Это делает приложение полностью настраиваемым.

**Глава 3. СОЗДАНИЕ ИГРЫ-ТРЕНАЖЕРА «ЯДЕРНЫЕ РЕАКЦИИ» НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ DELPHI**

***Задача практической части* -** написать программу создания игры-тренажера «Ядерные реакции» на языке программирования Delphi, которая, во-первых, генерирует ячейки ядерной реакции с некоторыми пустыми клетками; во-вторых, проверяет, является ли написанная реакция ядерной.

**3.1. Оформление главного диалогового окна**

В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Position* установить *PoScreenCenter* – для того, чтобы после запуска программы, форма появлялась в центре экрана. Создать форму: *Файл - Новый - Форма.* На экране появилось окно с названием *Form1*. В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Caption* изменить на *Ядерные реакции*.

В редакторе *Image Editor* (вкладка *Инструменты*) создать Icon File, нарисовать смайлик. Сохранить в файле Icon1.ico. В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* справа от свойства Icon щелкнуть мышью и в окне выбора файла иконы загрузить созданный файл. Рядом с названием *Ядерные реакции* появился смайл.

1) *TMainMenu* – главное меню формы, *Name* – *MainMenu1* (Содержит один пункт меню *«Help» (Name – Help1)* с двумя пунктами подменю *«Ядерные реакции» (Name – N1)* и *«Инструкция к выполнению» (Name – N2)).*

После установки компонента на форму создать пункт меню *«Help».* Для этого дважды щелкнуть на компоненте левой кнопкой мыши, в окне *Инспектора объектов* на вкладке *Свойства* ввести название пункта меню *«Help»* в строке свойства *Caption*, нажать клавишу *Enter*. Для создания пунктов подменю щелкнуть мышью в окне конструктора меню ниже пункта меню *«Help»,* в строке свойства *Caption* ввести название *«Ядерные реакции»,* нажать клавишу *Enter*, набрать *«Инструкция к выполнению»,* нажать *Enter*. Чтобы связать с выбором пункта меню нужное действие следует определить процедуру обработки его события *OnClick* – вызов соответствующего диалогового окна. Для этого перейти на вкладку *События* (диалоговое окно *Инспектор объектов*), в строке события *OnClick* выбрать *Help1.* Для второго события выполнить аналогичные действия.

2) *TPanel* – панель для размещения кнопок. Установить компонент на форму. На вкладке *Свойства з*начение свойства *Align* выбрать *alBottom*, чтобы панель разместить внизу формы. Свойство *Caption* очистить, *Name – Panel1*.

3) *TButton* – кнопка (одна кнопка *«Проверить»*, вторая, расположенная над ней, *«Начать»*). Кнопку *Button1* разместить на панели, *Name – Button1*. Свойство *Caption* – *«Проверить»*. Кнопку *Button2 (Name – Button2*) разместить на кнопке *Button1*. Свойство *Caption – «Начать».*

4) *TLabel* – метка для вывода текста комментариев действий пользователя. Метку *Label* разместить на форме. *Name – Lb1.* Свойство *Caption* очистить. Значение свойства *Align* выбрать *alBottom.* Свойство *Color* – выбрать цвет. С помощью свойства *Font* откорректировать шрифт.

5) *TEdit* – поле с маской ввода символов химических элементов. Установить 4 поля. Name – Edit1 .. Edit4. С помощью свойства *Font* откорректировать шрифт. *Text* – очистить.

Перейти на вкладку *Additional.*

6) *TbitBtn* – кнопка с изображением (для создания кнопки *Close* - закрытие окна). Кнопку *Close* разместить на панели. *Name – BitBtn1*. Свойство *Kind – bkClose*. Значение *bkClose* определяет типовой значок, надпись *"Close"* на кнопке, а также типовую функцию закрытия окна, которая будет связана с этой кнопкой.

7) *TMaskEdit* – поле с маской ввода. Установить 8 полей с маской ввода только числовой информации c ниже перечисленными значениями (для записи атомных масс и зарядов частиц). *Name* – *MaskEdit1 .. MaskEdit8*. Раскрыть диалоговое окно свойства *EditMask.* В поле Input Mask ввести следующую последовательность символов: *9999;0;.* В поле *Character for Blanks* вместо умалчиваемого символа-заполнителя, которым является символ подчеркивания *(\_),* указать *пробел* и закрыть окно.

8) *TStaticText* – поле постоянного текста (для показа знаков «+» и «=»). Установить 3 поля со следующими свойствами: *Name – StaticText1 .. StaticText3, Caption – «+»* или *«=»*, *Color* – выбрать цвет.

9) *TImage* – компонент, обеспечивающий фоновую заставку. Разместить на форме. Свойство *Align* *– allClient, Name – Image1*. Для того, чтобы разместить заставку выполнить следующий путь: свойство *Picture – Открыть – Выбрать картинку – ОК*. На форме появилась заставка. Свойство *Proportional – True*.

**3.2. Оформление диалогового окна «Ядерные реакции»**

В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Position* установить *PoScreenCenter* – для того, чтобы после запуска программы, форма появлялась в центре экрана. Создать форму: *Файл - Новый - Форма.* На экране появилось окно с названием *Form2*. В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Caption* изменить на *Ядерные реакции*. *Name – OKBottomDlg.* Справа от свойства Icon щелкнуть мышью и в окне выбора файла иконы загрузить смайл. Рядом с названием *Ядерные реакции* появился смайл.

1) *TMemo* – многострочное редактируемое текстовое поле вкладки *Standart*. Разместить на форме. Щелкнуть мышью справа от свойства *Lines*. В окне *String List Editor* ввести текст об ядерных реакциях. С помощью свойства *Font* откорректировать шрифт. *Color* – выбрать цвет. *Name – Memo1*.

2) *TButton* – кнопка «ОК» для выхода из диалогового окна. Кнопку *Button1* разместить на панели, *Name – Button1*. Свойство *Caption* – *«ОК»*.

3) *TImage* – компонент, обеспечивающий фоновую заставку. Разместить на форме. Свойство *Align* *– allClient, Name – Image1*. Для того, чтобы разместить заставку выполнить следующий путь: свойство *Picture – Открыть – Выбрать картинку – ОК*. На форме появилась заставка. Свойство *Proportional – True*.

**3.3. Оформление диалогового окна «Инструкция к выполнению»**

В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Position* установить *PoScreenCenter* – для того, чтобы после запуска программы, форма появлялась в центре экрана. Создать форму: *Файл - Новый - Форма.* На экране появилось окно с названием *Form3*. В диалоговом окне *Инспектор Объекта* на вкладке *Свойства* значение свойства *Caption* изменить на *Инструкция к выполнению*. *Name – OKBottomDlg1.* Справа от свойства Icon щелкнуть мышью и в окне выбора файла иконы загрузить смайл. Рядом с названием *Инструкция к выполнению* появился смайл.

1) *TMemo* – многострочное редактируемое текстовое поле вкладки *Standart*. Разместить на форме. Щелкнуть мышью справа от свойства *Lines*. В окне *String List Editor* ввести текст с правилами игры. С помощью свойства *Font* откорректировать шрифт. *Color* – выбрать цвет. *Name – Memo1*.

2) *TButton* – кнопка «ОК» для выхода из диалогового окна. Кнопку *Button1* разместить на панели, *Name – Button1*. Свойство *Caption* – *«ОК»*.

3) *TImage* – компонент, обеспечивающий фоновую заставку. Разместить на форме. Свойство *Align* *– allClient, Name – Image1*. Для того, чтобы разместить заставку выполнить следующий путь: свойство *Picture – Открыть – Выбрать картинку – ОК*. На форме появилась заставка. Свойство *Proportional – True*.

**3.4. Обработка событий и написание программы**

После оформления необходимых диалоговых окон, нужно определить способы обработки всех событий (нажатие кнопок, вызов меню, запуска игры и т.д), а также написать алгоритм программы на языке программирования *Delphi* (*Приложение*).

1) Заполнение полей-клеток ядерной реакции.

а) Процедура обработки события *OnClick* кнопки *Button2 («Начать»).* Процедура *zapolnenie(x,y:integer)* (по входящему параметру *х* одним из трех способов заполняет ячейки, по входящему параметру *у* одним из четырех способов очищает некоторые ячейки, выводит в метке соответствующую надпись, показывает кнопку *«Проверить»,* прячет кнопку *«Начать»).*

б) В теле главной процедуры проверить, была ли ситуация *«Попробуй еще раз» (z=true).* Если была, то вызвать процедуру *zapolnenie(x,y:integer)* со старыми значениями *х* и *у*; если нет *(z=false),* генерировать *у* – способ удаления ячеек, *х* – способ заполнения ячеек. Заполнить ячейки.

2) Проверка реакции на предмет, является ли она ядерной реакцией.

Процедура обработки события *OnClick* кнопки *Button1 («Проверить»).* Следует преобразовать к числовому виду значения полей *TMaskEdit*. Если получилось (не осталось пустых полей), найти суммы по парам химических элементов. Если полученные суммы равны и ячейки заполнены правильными символами, то вывести в метке надпись «Молодец! Все верно!», скрыть кнопку «Проверить», показать кнопку «Начать», z:=false (следующее заполнение будет сгенерировано). Если полученные суммы не равны или ячейки заполнены не верными символами, то вывести в метке надпись «Попробуй еще раз», скрыть кнопку «Проверить», показать кнопку «Начать», z:=true (следующее заполнение будет тем же, х и у не изменятся). Если не получилось преобразовать к числовому виду значения полей TMaskEdit, подать звуковой сигнал, вывести сообщение «Не все клетки заполнены», выйти из процедуры.

**3.5. Создание проекта**

После записи программы следует создать приложение. Для этого нужно выполнить следующие действия: *Файл – Сохранить проект как* – на данном этапе присвоить проекту имя с расширением *.exe* – нажать кнопку «*ОК»*.

## **3.6. Анализ и тестирование**

## Программа запускается с помощью файла *Project2.exe.* После запуска открывается окно, в котором необходимо нажать кнопки «Начать» или покинуть программу.



Рис. 9. Программа после запуска

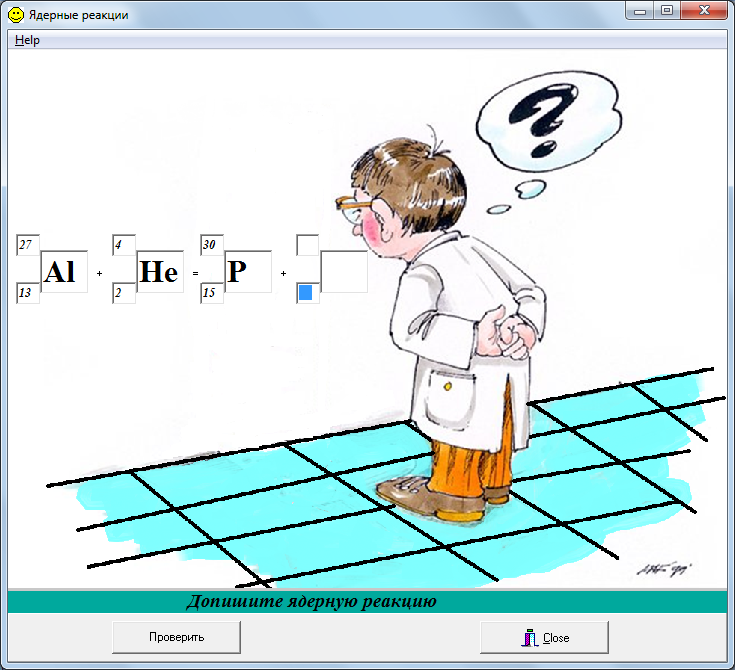


Рис. 10. Программа после нажатия кнопки «Начать»

После заполнения ячеек нажать на кнопку «Проверить», и на экране выводится сообщение (рисунок 11 или 12).

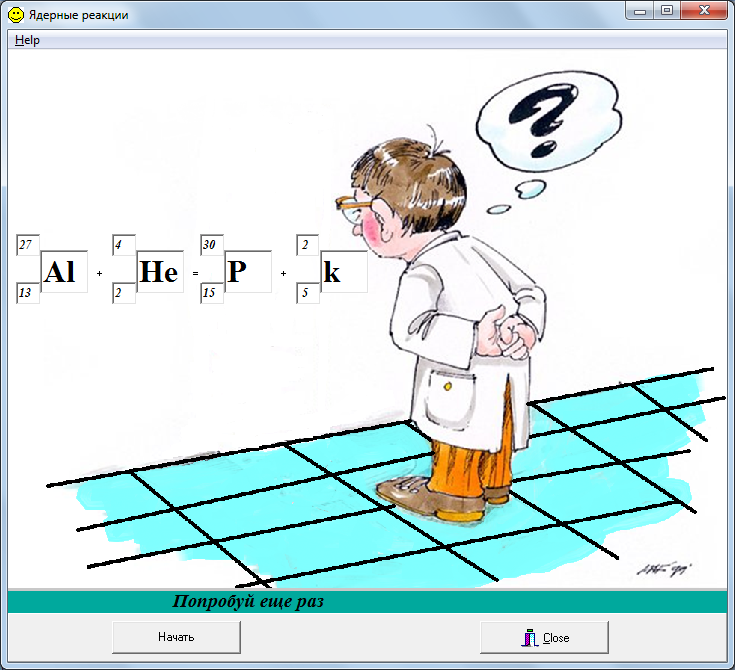


Рис. 11. Программа после щелчка мыши по кнопке «Проверить»

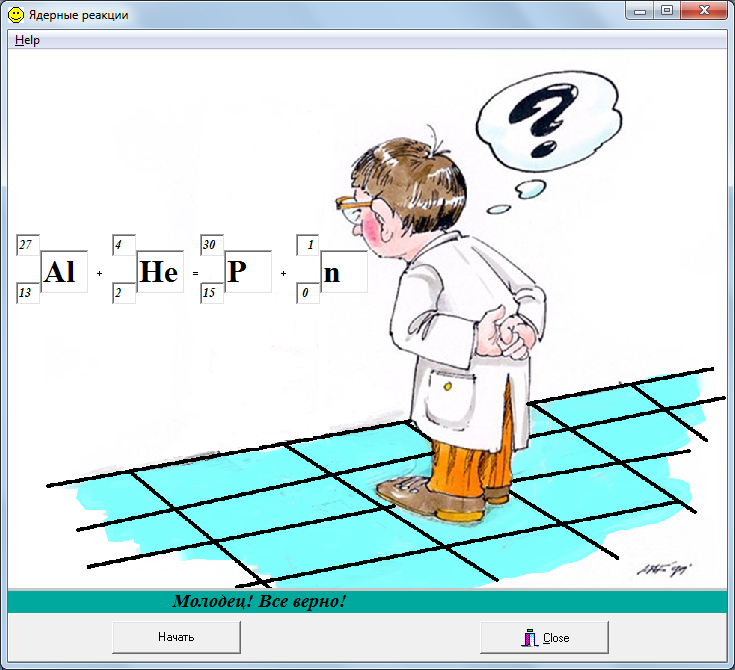


Рис. 12. Программа после щелчка мыши по кнопке «Проверить»

На этом этапе программа была неоднократно протестирована, ошибок не выявлено.

Программа имеет меню «Help», которое содержит два пункта: «Ядерные реакции» и «Инструкция к выполнению».

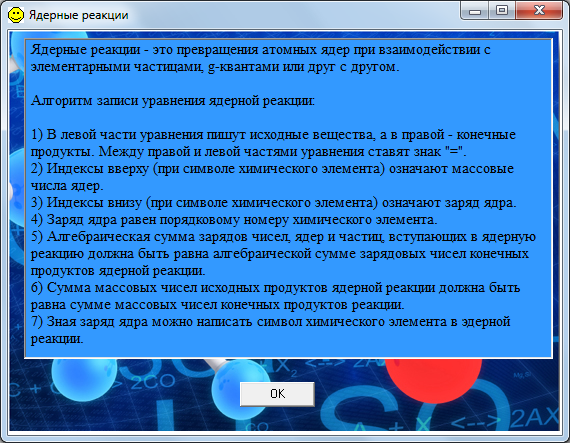


Рис. 13. Программа после выбора пункта меню «Ядерные реакции»

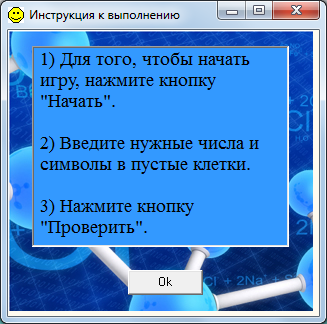


Рис. 14. Программа после выбора пункта меню «Инструкция к выполнению»

**Заключение**

Повышение эффективности обучения химии средствами дидактической игры может быть осуществлено в том случае, если в учебном процессе дидактические игры применять целенаправленно и систематически, с учетом их классификации, соблюдая поэтапный алгоритм их составления и методические рекомендации по организации и проведению.

В настоящее время в нашу жизнь широко внедряются информационные технологии. На сегодняшний день, наверное, не осталось такой области, где бы они ни применялись.

Постоянно растет объем и сложность обрабатываемой информации, требуются все новые и новые виды ее представления. Как показывает практика, большинство пользователей уже не представляют себе, как бы они выполняли свою работу, с которой прекрасно справлялись еще 5 – 10 лет назад, без помощи компьютера.

В ходе исследовательской работы разработана тестирующая программа. Проектирование выполнено в системе Delphi. В работе:

- представлена информация о дидактических играх и влиянии их на учебно-воспитательный процесс;

- приведен теоретический материал о компонентах, используемых в программе;

- представлен листинг программы;

- приведен анализ и тестирование программы с подробным описанием.

Создавая свои игры и используя уже кем-то созданные учитель приобщается к информационной культуре, которая в настоящее время очень быстро развивается, отстать от которой, значит потеряться в информационном обществе.

Для хранения подобных тренажеров не нужны шкафы, библиотеки, они компактно хранятся в памяти компьютера или на внешних носителях. Их не так сложно структурировать и систематизировать, нежели материалы, которые находятся в распечатанном виде. Это один из огромного множества плюсов, ведь накапливая с годами тестирующие программы, структурируя их и, «раскладывая по полочкам» учитель создает свою систему, на которую он опирается в процессе обучения. И пусть на начальном этапе учитель будет не регулярно использовать игры в своей преподавательской деятельности, но потом он поймет, что начавшая складываться система значительно помогает ему в процессе обучения. Это позволит сформировать базу данных хранимых ресурсов, а при достаточном их наборе и разумных систематизации и каталогизации организовать быстрый и доступный поиск и выбор наиболее соответствующих интересующей теме игры как для самого учителя, так и для учеников.

**Список использованной литературы**

1) Штремплер Г.И., Химия на досуге. Книга для учащихся. Учебное издание. Фрунзе. Главная редакция КСЭ. 1990. 192 с.

2) Штремплер Г.И., Пичугина Г.А.  Дидактические игры в обучении химии /Учебное издание/ Москва: Дрофа, 2003. 96 с.: ил.

3) Космодемьянская С. С., Гильманшина С. И. Методика обучения химии: учебное пособие. – Казань: ТГГПУ, 2011. – 136 с.

4) Архангельский А.Я. Delphi 6. Справочное пособие – М.: ЗАО «Издательство БИНОМ», 2001.

5) Фаронов В.В. Delphi. Программирование на языке высокого уровня: Учебник для вузов – СПб.: Питер, 2008. -640 с.

6) Культин Н.Б. Основы программирования в Delphi 7 – СПб.: БХВ-Петербург, 2003.

7) <http://physics.herzen.spb.ru/library/03/01/delphi/LR_1/L_02.htm>

*Приложение*

**Листинг программы создания главного окна**

unit igra;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Menus, StdCtrls, ExtCtrls, Mask, Buttons;

type

TForm1 = class(TForm)

MainMenu1: TMainMenu; //Главное меню формы

Help1: TMenuItem; //Один пункт меню "Help"

N1: TMenuItem; //Пункт подменю "Ядерные реакции"

N2: TMenuItem; //Пункт подменю "Инструкция к выполнению"

Panel1: TPanel; //Панель, на которой размещены кнопки

Button1: TButton; //Кнопка «Проверить»

Button2: TButton; //Кнопка «Начать»

Label1: TLabel; //Метка, для вывода комментариев к программе

BitBtn1: TBitBtn; //Кнопка "Close"

MaskEdit1: TMaskEdit; //Поля с маской ввода только числовой информации

MaskEdit2: TMaskEdit;

MaskEdit3: TMaskEdit;

MaskEdit4: TMaskEdit;

MaskEdit5: TMaskEdit;

MaskEdit6: TMaskEdit;

MaskEdit7: TMaskEdit;

MaskEdit8: TMaskEdit;

StaticText1: TStaticText; //Поля постоянного текста (для "+" или "=")

StaticText2: TStaticText;

StaticText3: TStaticText;

Image1: TImage; //Фоновая заставка

Edit1: TEdit; //Поля для ввода текстовой информации (записи ядерной реакции)

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

procedure FormCanResize(Sender: TObject; var NewWidth,

NewHeight: Integer; var Resize: Boolean);

procedure FormActivate(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure N1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses instrukchia, reakchii;

var

x,y:integer; //х – номер способа заполнения ячеек, у – номер способа очистки ячеек

z:boolean; //Индикатор правильности заполнения пользователем пустых ячеек

{$R \*.dfm}

procedure TForm1.FormCanResize(Sender: TObject; var NewWidth, //Процедура, запрещающая изменение размеров формы

NewHeight: Integer; var Resize: Boolean); //

begin

Resize:=false;

end;

procedure TForm1.FormActivate(Sender: TObject); //Активация формы

begin

OKBottomDlg1.Hide; //Скрыть диалоговое окно "Инструкция к выполнению"

OKBottomDlg.Hide; //Скрыть диалоговое окно «Ядерные реакции»

Caption:='Ядерные реакции’; //Установить заголовок "Ядерные реакции"

Button2.Show; //Показать кнопку "Начать"

Button1.Hide; //Показать кнопку "Проверить"

MaskEdit1.Hide; //Скрыть поля ввода

MaskEdit2.Hide;

MaskEdit3.Hide;

MaskEdit4.Hide;

MaskEdit5.Hide;

MaskEdit6.Hide;

MaskEdit7.Hide;

MaskEdit8.Hide;

Edit1.Hide;

Edit2.Hide;

Edit3.Hide;

Edit4.Hide;

StaticText1.Hide;

StaticText2.Hide;

StaticText3.Hide;

end;

procedure TForm1.Button2Click(Sender: TObject); //обработка нажатия кнопки «Начать»

procedure zapolnenie(x,y:integer); //Процедура заполнения

begin

MaskEdit1.Show; //Показать поля вода

MaskEdit2.Show;

MaskEdit3.Show;

MaskEdit4.Show;

MaskEdit5.Show;

MaskEdit6.Show;

MaskEdit7.Show;

MaskEdit8.Show;

Edit1.Show;

Edit2.Show;

Edit3.Show;

Edit4.Show;

StaticText1.Show;

StaticText2.Show;

StaticText3.Show;

MaskEdit1.ReadOnly:=false; //Очистить поля ввода

MaskEdit2.ReadOnly:=false;

MaskEdit3.ReadOnly:=false;

MaskEdit4.ReadOnly:=false;

MaskEdit5.ReadOnly:=false;

MaskEdit6.ReadOnly:=false;

MaskEdit7.ReadOnly:=false;

MaskEdit8.ReadOnly:=false;

Edit1.ReadOnly:=false;

Edit2.ReadOnly:=false;

Edit3.ReadOnly:=false;

Edit4.ReadOnly:=false;

Case x of //Выбрать способ заполнения значений клеток

0:begin

Edit1.Text:='Al'; //Заполнить поля ввода символами

Edit2.Text:='He';

Edit3.Text:='P';

Edit4.Text:='n';

MaskEdit1.Text:=IntToStr(27); //Заполнить поля ввода числами, которые нужно преобразовать к строковому представлению

MaskEdit2.Text:=IntToStr(4);

MaskEdit3.Text:=IntToStr(30);

MaskEdit4.Text:=IntToStr(1);

MaskEdit5.Text:=IntToStr(13);

MaskEdit6.Text:=IntToStr(2);

MaskEdit7.Text:=IntToStr(15);

MaskEdit8.Text:=IntToStr(0);

end;

1:begin

Edit1.Text:='B';

Edit2.Text:='n';

Edit3.Text:='He';

Edit4.Text:='Li';

MaskEdit1.Text:=IntToStr(10);

MaskEdit2.Text:=IntToStr(1);

MaskEdit3.Text:=IntToStr(4);

MaskEdit4.Text:=IntToStr(7);

MaskEdit5.Text:=IntToStr(5);

MaskEdit6.Text:=IntToStr(0);

MaskEdit7.Text:=IntToStr(2);

MaskEdit8.Text:=IntToStr(3);

end;

2:begin

Edit1.Text:='N';

Edit2.Text:='n';

Edit3.Text:='B';

Edit4.Text:='He';

MaskEdit1.Text:=IntToStr(14);

MaskEdit2.Text:=IntToStr(1);

MaskEdit3.Text:=IntToStr(11);

MaskEdit4.Text:=IntToStr(4);

MaskEdit5.Text:=IntToStr(7);

MaskEdit6.Text:=IntToStr(0);

MaskEdit7.Text:=IntToStr(5);

MaskEdit8.Text:=IntToStr(2);

end;

end;

Case y of //Выбрать способ удаления значений некоторых клеток

0:begin

Edit4.Text:=''; Edit4.SetFocus; //Удалить символы с поля и передать ему фокус ввода

MaskEdit4.Text:=''; MaskEdit4.SetFocus; //Удалить число с поля и передать ему фокус ввода

MaskEdit8.Text:=''; MaskEdit8.SetFocus;

MaskEdit1.ReadOnly:=true; //Запретить ввод чисео и изменение остальных полей

MaskEdit2.ReadOnly:=true;

MaskEdit3.ReadOnly:=true;

MaskEdit5.ReadOnly:=true;

MaskEdit6.ReadOnly:=true;

MaskEdit7.ReadOnly:=true;

Edit1.ReadOnly:=true;

Edit2.ReadOnly:=true;

Edit3.ReadOnly:=true;

end;

1:begin

Edit1.Text:=''; Edit4.SetFocus;

MaskEdit1.Text:=''; MaskEdit4.SetFocus;

MaskEdit5.Text:=''; MaskEdit8.SetFocus;

MaskEdit4.ReadOnly:=true;

MaskEdit2.ReadOnly:=true;

MaskEdit3.ReadOnly:=true;

MaskEdit8.ReadOnly:=true;

MaskEdit6.ReadOnly:=true;

MaskEdit7.ReadOnly:=true;

Edit4.ReadOnly:=true;

Edit2.ReadOnly:=true;

Edit3.ReadOnly:=true;

end;

2:begin

Edit2.Text:=''; Edit4.SetFocus;

MaskEdit2.Text:=''; MaskEdit4.SetFocus;

MaskEdit6.Text:=''; MaskEdit8.SetFocus;

MaskEdit4.ReadOnly:=true;

MaskEdit1.ReadOnly:=true;

MaskEdit3.ReadOnly:=true;

MaskEdit8.ReadOnly:=true;

MaskEdit5.ReadOnly:=true;

MaskEdit7.ReadOnly:=true;

Edit4.ReadOnly:=true;

Edit1.ReadOnly:=true;

Edit3.ReadOnly:=true;

end;

3:begin

Edit3.Text:=''; Edit4.SetFocus;

MaskEdit3.Text:=''; MaskEdit4.SetFocus;

MaskEdit7.Text:=''; MaskEdit8.SetFocus;

MaskEdit4.ReadOnly:=true;

MaskEdit2.ReadOnly:=true;

MaskEdit1.ReadOnly:=true;

MaskEdit8.ReadOnly:=true;

MaskEdit6.ReadOnly:=true;

MaskEdit5.ReadOnly:=true;

Edit4.ReadOnly:=true;

Edit2.ReadOnly:=true;

Edit1.ReadOnly:=true;

end;

end;

Label1.Caption:='Допишите ядерную реакцию'; //Появление соответствующей надписи в метке

Button1.Show; //Показать кнопку «Проверить»

Button2.Hide; //Скрыть кнопку "Начать"

end;

BEGIN

if z then //Проверка правильности ввода недостающих чисел и символов

zapolnenie(x,y) //Заполнение «Попробуй еще раз» старыми

значениями х и у

else //Заполнение в первый раз и при каждом правильном решении

begin

y:=random(4); //Выбрать случайным образом способ удаления значений полей

x:=random(3); //Выбрать случайным образом способ заполнения полей

zapolnenie(x,y); //Заполнить ячейки

end;

end;

procedure TForm1.Button1Click(Sender: TObject); //Обработка нажатия кнопки «Проверить»

var s1,s2,s3,s4:integer; //Попарные суммы элементов

begin

try //Попытка преобразовать в числовое представление значения полей и нахождение соответствующих сумм

s1:=StrToInt(Trim(MaskEdit1.Text))+StrToInt(Trim(MaskEdit2.Text));

s2:=StrToInt(Trim(MaskEdit3.Text))+StrToInt(Trim(MaskEdit4.Text));

s3:=StrToInt(Trim(MaskEdit5.Text))+StrToInt(Trim(MaskEdit6.Text));

s4:=StrToInt(Trim(MaskEdit7.Text))+StrToInt(Trim(MaskEdit8.Text));

If ((s1=s2) and (s3=s4) and (Edit1.Text='Al') and (Edit2.Text='He') and

(Edit3.Text='P') and (Edit4.Text='n')) or ((s1=s2) and (s3=s4) and (Edit1.Text='B')

and (Edit2.Text='n') and (Edit3.Text='He') and (Edit4.Text='Li')) or ((s1=s2)

and (s3=s4) and (Edit1.Text='N') and (Edit2.Text='n') and (Edit3.Text='B')

and (Edit4.Text='He'))

then //Если соответствующие суммы и символы равны

begin

Label1.Caption:='Молодец! Все верно!'; //Вывести в метке соответствующую надпись

z:=false; //z определяет, что заполнение следующих значений ячеек нужно начать с новой генерации начального значения способа заполнения х и способа удаления у

Button2.Show; //Показать кнопку «Начать»

Button1.Hide; //Скрыть кнопку «Проверить»

end

else //Если хоть одно из условий не верно

begin

Label1.Caption:=' Попробуй еще раз'; //Вывести в метке соответствующую надпись

z:=true; //z определяет, что заполнение ячеек нужно произвести старыми значениюми х и у

Button2.Show; //Показать кнопку «Начать»

Button1.Hide; //Скрыть кнопку «Проверить»

end;

Except //Какое-то поле осталось не заполнено

MessageBeep(mb\_Ok); //Звуковой сигнал

ShowMessage('Не все клетки заполнены!'); //Вывод окна сообщения

Exit; //Выход из процедуры

end;

end;

procedure TForm1.N2Click(Sender: TObject); //Выбор меню «Инструкция к выполнению»

begin

OKBottomDlg1.Show; //Показ диалогового окна «Инструкция к выполнению»

end;

procedure TForm1.N1Click(Sender: TObject); //Выбор меню «Ядерные реакции»

begin

OKBottomDlg.Show; //Показ диалогового окна «Ядерные реакции»

end;

end.

**Листинг модуля вызова диалогового окна «Ядерные реакции»**

unit reakchii;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, jpeg, ExtCtrls;

type

TOKBottomDlg = class(TForm)

Memo1: TMemo; //Текстовое поле

Image1: TImage; //Фоновая заставка

Button1: TButton; //Кнопка "ОК"

procedure FormCanResize(Sender: TObject; var NewWidth,

NewHeight: Integer; var Resize: Boolean);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

OKBottomDlg: TOKBottomDlg;

implementation

{$R \*.dfm}

procedure TOKBottomDlg.FormCanResize(Sender: TObject; var NewWidth, //Процедура,

NewHeight: Integer; var Resize: Boolean); //запрещающая изменение размеров формы

begin

Resize:=false;

end;

procedure TOKBottomDlg.Button1Click(Sender: TObject);

begin

OKBottomDlg.Close; //Закрытие окна по щелчку кнопки "ОК"

end;

end.

**Листинг модуля вызова диалогового окна «Инструкция к выполнению»**

unit instrukchia;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, jpeg, ExtCtrls, StdCtrls, Buttons;

type

TOKBottomDlg1 = class(TForm)

Memo1: TMemo; //Текстовое поле

Image1: TImage; //Фоновая заставка

Button1: TButton; //Кнопка "ОК"

procedure Button1Click(Sender: TObject);

procedure FormCanResize(Sender: TObject; var NewWidth,

NewHeight: Integer; var Resize: Boolean);

private

{ Private declarations }

public

{ Public declarations }

end;

var

OKBottomDlg1: TOKBottomDlg1;

implementation

{$R \*.dfm}

procedure TOKBottomDlg1.Button1Click(Sender: TObject);

begin

OKBottomDlg1.Close; //Закрытие окна по щелчку кнопки "ОК"

end;

procedure TOKBottomDlg1.FormCanResize(Sender: TObject; var NewWidth, //Процедура, запрещающая изменение размеров формы

NewHeight: Integer; var Resize: Boolean);

begin

Resize:=false;

end;

end.