***ПРИМЕР 1* Курсовой работы оценка 6-7**

Характеристика и замечания:

Хорошее оформление, введение правильное и хорошее.

В теоретической части пункты 1.1 и 1.2 хорошие, пункт 1.3

удовлетворительный и не хватает пункта 1.4 **Понятие и основные признаки приложения модели.**

В практической части не хватает пунктов ПРОЕКТИРОВАНИЕGUI ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

Описание используемых компонент

И слишком подробно описан процесс создания приложения и не раскрыта ТЕМА применения OLE – технологии.

Заключение слишком короткое и посредственное.

Список использованных источников слишком мал.

Нет Содержания электронного носителя (диска).

|  |
| --- |
|  **МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ****УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ"ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"****факультет повышения квалификации и переподготовки кадров**  |
| Курсовая работа**ИСПОЛЬЗОВАНИЕ OLE-ТЕХНОЛОГИИ ПРИ СОЗДАНИИ GUI ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ – МОДЕЛИ** |
| Студентка гр. ИС-?? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Иванова подпись, датаРуководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шедько подпись, датаРабота защищена «\_\_» \_\_\_\_\_\_\_\_\_\_ 200\_\_ г. с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_».Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ подпись Фамилия И.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ подпись Фамилия И.О.  |
| Витебск, 2015 |

**Список сокращений**

RAD - Rapid Application Development

OLE - Object Linkingand Embedding

GUI - Graphical User Interface

SDI – Single Document Interface

MDI – Multi Document Interface

**СОДЕРЖАНИЕ**

Введение………………………………………………………………………….4

1. Теоритическая часть
	1. GUI интерфейс…………………………………………………………….6
	2. Понятие OLE технологии и ее использование…………………………..7
	3. Обоснование выбора среды разработки…………………………….10
	4. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. Практическая часть
	1. Создание главного окна программы.………………………………….11
	2. Создание окна «О программе»………………………………………..16
	3. Сохранение приложения……………………………………………… 18
	4. Компиляция и запуск программы...................................................... 19

2.5 Настройка приложения ………………………………………………...20

Заключение………………………………………………………………………23

Список использованных источников…………………………………………..24

Приложение А…………………………………………………………………...25

Приложение Б…………………………………………………………………..27

**ВВЕДЕНИЕ**

Интерес к программированию постоянно растет. Это связано с развитием и внедрением в повседневную жизнь информационных технологий. Если человек имеет дело с компьютером, то рано или поздно у него возникает желание, а иногда и необходимость, научиться программировать. Среди пользователей персональных компьютеров в настоящее время наиболее популярна операционная система Windows, и естественно, что тот, кто хочет программировать, хочет и писать программы, которые будут работать в Windows.

Актуальность данной работы обусловлена:

1. внедрением и развитием новых информационных технологий в процессе современного образования, позволяющих существенно повысить уровень обучения;

 2. создание наглядного приложения – моделиСолнечной системы, широко используемого в процессе обучения.

Целью курсовой работы является создание GUIинтерфейса
приложения – модели, используя OLE-технологии. Исходя из указанной цели, можно выделить задачи, поставленные в курсовой работе:

– разработка GUIинтерфейса приложения;

– изучениеOLE-технологий;

– разработка и запускприложения – модели.

Объектом исследования в курсовой работе является визуальное программирование.

Предметом исследования в курсовой работе являются создание приложения, в рамках темы «Использование OLE-технологии при создании GUIинтерфейса приложения – модели».

Данная программа представляет наглядное приложение, не только для учеников 11 классовсредней школы по предмету астрономия, но и для учащихся средних специальных и высших учебных заведений и включает в себя:

1. наглядную модель Солнечной системы в целом;

2. объектов Солнечной системы в частности;

3. справочный материал физических характеристик планет.

**ТЕОРЕЧИЧЕСКАЯ ЧАСТЬ**

**1.1 GUI интерфейс**

Интерфейс – способ взаимодействия человека с компьютером. Средствами пользовательского интерфейса в приложении обеспечивается доступ к данным и задачам, решаемым приложением[1]

Имеется два типа интерфейсов: SDI (SingleDocumentInterface - однодокументный интерфейс) и MDI (MultiDocumentInterface - многодокументный интерфейс). SDI-приложения работают одновременно с одним документом, MDI-приложения предназначены для одновременной работы со множеством однотипных документов. При этом все документы располагаются внутри одного контейнера.

Я буду реализовывать SDI-приложения. Его окна могут быть двух видов - модальные и немодальные. Создаются они одинаково, разница заключается только в способе вывода этих окон на экран. Модальное окно блокирует программу, не даёт с ней работать, пока вы это окно не закроете. В моем приложении будет модальное окно– окно «О программе». Которое находится в меню «?».Пока вы не нажмете «ОК», закрыв это окно, вы не сможете работать с основной программой. Немодальные окна позволяют переключаться между ними, и программой, и работать одновременно и там, и там. [2]

Под графическим интерфейсом пользователя (GraphicalUserInterface — GUI)подразумевается тип экранного представления, при котором пользователь можетвыбирать команды, запускать задачи и просматривать списки файлов, указывая напиктограммы или пункты в списках меню, показанных на экране. Действия могут,как правило, выполняться с помощью мыши, либо нажатием клавиш на клавиатуре.[3, с.289]

C++Builder предоставляет разработчику приложения возможностибыстрого и качественного проектирования графического интерфейса – различных окон, кнопок, меню и т.д.

Графический интерфейс моего приложения будет включать в себя:

- окно приложения и модальное окно;

- главное меню,реализуется компонентом MainMenu;

- клавиши быстрого доступа ко всем разделам меню;

- ярлычки подсказок, всплывающие при перемещении курсора мыши над компонентами Image;

- файл справки, темы которого отображаются при нажатии клавиши F1;

- кнопку Button1 для выхода из приложения и закрытия окна «О программе».

**1.2 Понятие OLE технологии и ее использование**

OLE (ObjectLinkingandEmbedding) – технология связывания и внедрения объектов в другие документы и объекты. OLE позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад. Для этого дан­ные оформляются в виде OLE-объекта, несущего информацию о том, каким приложением он создан и редактируется. Приложение, которое обеспечивает создание и редактирование OLE-объекта, называется OLE-сервером. Прило­жение, которое содержит и отображает OLE-объект, называется OLE-контейнером.[4]

В форме OLE-объектов можно встраивать в свои приложения все что угодно: документы, электронные таблицы, диаграммы, картинки, а также любые другие данные, создаваемые OLE-серверами, такими, например, как MicrosoftWord, MicrosoftExcel, CorelDRAW, CorelPhoto-Paint и другие. Та­ким образом, технология OLE открывает безграничные возможности для ин­теграции разнородных данных с целью создания составных документов*.* Ти­пичным составным документом является документ MicrosoftWord. Наряду с текстом в нем могут содержаться электронные таблицы, диаграммы, картин­ки, любые другие данные. Сам по себе такой документ может быть интегри­рован в другой документ.

Существует два различных способа вставки OLE-объекта в OLE-контейнер: привязка (linking) и внедрение (embedding).Привязказаключается в том, что в OLE-контейнере хранится ссылка на файл с данными OLE-объекта.Внедрениезаключается в том, что в OLE-контейнере хранятся все данные OLE-объекта.

Данные привязанного OLE-объекта остаются доступны для модификации дру­гими приложениями, а данные внедренного OLE-объекта защищены от этого.

Различие между привязкой и внедрением особенно хорошо проявляется при создании составных документов. Привязка экономит оперативную память и употребляется в тех случаях, когда одни и те же данные должны фигуриро­вать в нескольких местах составного документа. Внедрение обеспечивает за­щиту данных от удаления или модификации другими приложениями и упро­щает перенос составных документов с компьютера на компьютер, так как все данные собраны в одном файле.[3, c.472]

Данное приложениепредоставляет пользователю возможность создания OLE-объектов и имеется в своем меню командаФайл- >Новый. Когда пользователь выберет эту команду, приложение выдает на экран стандартное окно диалога, в котором предлагает на выбор:

а) Либо создать OLE-объект заново, выбрав его тип из списка зареги­стрированных в WindowsOLE-серверов (рисунок 1.1 — этот подход позволяет лишь внедрять OLE-объекты)

Рисунок 1.1 – Окно вставки объекта OLE — вставка нового OLE-объекты

б) Либо создать OLE-объект из существующего на диске файла (рисунок 1.2 — этот подход позволяет и привязывать, и внедрять OLE-объекты)

**Рисунок 1.2 – Окно вставки объектаOLE - вставка объектаиз файла**

Когда OLE-объект находится в OLE-контейнере, можно легко перейти к его редактированию, сделав на объекте двойной щелчок мыши. В ответ на это действие активизируется OLE-сервер, который предоставляет необходимые для редактирования средства. Существуют два способа активизации OLE-сервера: в отдельном окне и по месту.

Активизация в отдельном окнеозначает, что OLE-сервер запуска­ется как отдельное приложение в своем собственном окне. Это единственный способ запуска сервера в OLE первой версии. В моем приложении в отдельном окне будет запускаться файлы с изображением планет.

Активизация по местуозначает, что OLE-сервер запускается в окне OLE-контейнера. При этом меню OLE-сервера и OLE-контей­нера объединяются, а панели инструментов и строка состояния OLE-сервера заменяют собой панели инструментов и строку состо­яния OLE-контейнера. Редактирование происходит в окне OLE-контейнера, но вся обработка данных происходит в OLE-серве­ре. Это преимущественный способ запуска сервера в OLE второй версии. В данном приложении будет запускаться в окне OLE-контейнера файл с физическими характеристиками планет. Их можно будет изменять, дополнять и сохранять как этим документом, так и новым.

Хранение и отображение OLE-объектов в приложениях на C++Builder обеспе­чивает специальный компонент OleContainer,расположенный в палитре компо­нентов на странице System (рисунок 1.3).

Рисунок 1.3– Компонент OleContainer

Основные свойстваOleContainer:

Свойство Stateпозволяет определить состояние объекта и его сервера. Его значения:

- osEmpty–контейнер не содержит объекта;

- osLoaded– объект в контейнере, сервер не выполняется;

- osRunning– сервер запущен;

- osOpen– OLE –объект открыт в отдельном окне сервера;

- osInPlaceActive– объект активизирован «на месте», но меню еще не изменено. Это промежуточное состояние объекта перед полной загрузкой сервера.

- osUIActiveобъектактивизирован «на месте», меню изменено.

Свойство boolAllowInPlaceопределяет возможность редактировать внедренный объект «на месте». ЕслиAllowInPlace = trueиIconic = false(свойствоIconicопределяет должен ли объект быть представлен в виде пиктограммы), то «InPlace» – редактирование разрешено. ПриAllowInPlace = falseсервер будет открываться в отдельном окне.

Свойство AutoActivate имеет три возможных значения:aaManual,aaGetFocus,aaDoubleClickи определяет способы активизации загруженного в контейнер объекта.

По умолчанию AutoActivate = aaDoubleClick, то есть объект становится активным при двойном щелчке. ЗначениеaaGetFocusопределяет активизацию при получении фокуса ввода. ПриAutoActivate = aaManualза активизацию объекта отвечает программист. В этом случае можно использовать метод компонента-контейнераDoVerb, который отвечает за передачу команд от контейнера серверу.В данном приложении AutoActivateостаетсяaaDoubleClick.

**1.3 Обоснование выбора среды разработки**

Раньше рядовому программисту оставалось только мечтать о создании своих собственных программ, работающих в Windows. Единственным средством разработки был Borland C++ forWindows, явно ориентированный на профессионалов, обладающих серьезными знаниями и опытом. Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению на рынке целого ряда систем программирования, ориентированных на так называемую «быструю разработку», среди которых особо следует отметить MicrosoftVisualBasic и BorlandDelphi. В основе систем быстрой разработки (RAD-систем,RapidApplicationDevelopment − среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть работы по генерации кода программы, оставляя программисту работу по конструированию диалоговых окон и написанию функций обработки событий. Производительность программиста при использовании RAD систем фантастическая!

 Успех и популярность Delphi вызвал желание распространить методы быстрой разработки на область профессионального программирования, что и привело к появлению Borland C++ Builder.

C++ Builder− это среда быстрой разработки, в которой в качестве языка программирования используется язык *C++ Builder*(C++ BuilderLanguage).

Я в своей работе использовала шестую версию пакета −Borland C++ Builder 6.Borland C++ Builder 6 позволяет создавать различные программы: от простейших однооконных приложений до программ управления распределенными базами.

**1.4 Понятие и основные признаки приложения модели**

*// Здесь должен быть пункт описания, что такое приложение – модель //*

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

*2.0 ПРОЕКТИРОВАНИЕ GUI ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ*

//

**2.1 Создание главного окна программы**

Разрабатываемое приложение в C++ Builder начнем с создания стартовой формы— главного окна программы.

Стартовая форма создается путем изменения значений свойств формы Form1 (настройки формы) и добавления к форме необходимых компонентов.

Для изменения значений свойств объектов, в том числе и формы, используется вкладка Properties(Свойства) диалогового окна ObjectInspector. В левой колонке этой вкладки перечислены свойства выбранного объекта, в правой — указаны значения свойств.

При создании формы в первую очередь следует изменить значение свойства Caption (Заголовок). В случае надо заменить текст Form1 на «Модель Солнечной системы». Чтобы это сделать, нужно в окне ObjectInspectorщелкнуть левой кнопкой мыши в строке Caption(в результате будет выделено значение свойства и появится курсор) и введем текст: Модель солнечной системы.

В данном приложении значения свойству BorderStyle присвоим значение bsSingle, что означает тонкую границу, которая не позволяет размер окна во время работы путем захвата и перемещения изменить программ границы.

Некоторые свойства являются сложными, т. е. их значение определяется совокупностью значений других (уточняющих) свойств. Например, свойство Bordericons определяет, какие кнопки управления окном будут доступны во время работы программы. Значения этого свойства определяется совокупностью значений свойств MSystemMenu, biMinimize, biMaximize и biHelp, каждое из которых, в свою очередь, определяет наличие соответствующей командной кнопки в заголовке окна во время работы программы. Перед именами сложных свойств стоит значок "+", в результате щелчка на котором раскрывается список уточняющих свойств, значения которых можно задать обычным образом. В данном случае, значения свойствbiMinimize и biMaximize присваиваем значение False, что означает в заголовке окна нет кнопки Свернуть и Развернуть.

В результате выбора некоторых свойств (щелчка кнопкой мыши на свойстве), рядом со значением свойства появляется командная кнопка с тремя точками. Это значит, что задать значение свойства можно в дополнительном диалоговом окне, которое появится в результате щелчка на этой кнопке. Например, значение сложного свойства Font можно задать в окне ObjectInspectorпутем ввода значений уточняющих свойств, а можно воспользоваться стандартным диалоговым окном Шрифт,которое появится в результате щелчка на кнопке с тремя точками. В нем изменим значения свойств Name и Size на значения TimesNewRoman и 14 соответственно.

Теперь создадим меню. Для этого поместим на главную форму приложения компонент TMainMenu со страницы Standard. Нажав правую клавишу мыши, из контекстного меню выберем пункт MenuDesigner. Перемещаясь с помощью стрелок клавиатуры, создадим новые компоненты - пункты меню верхнего и последующего уровней, вводя текстовые строки в колонку значений напротив свойства Caption.

Создадим следующие меню: "Файл" с пунктами "Новый", "Открыть", "Сохранить", "Закрыть" и "?" с пунктом "О программе".

Для работы с меню с помощью клавиатуры используются клавиши быстрого доступа. Подходящую комбинацию клавиш можно выбрать, установив значение свойства ShortCut. Выбираем для пункта "Новый" – Ctr+Z, "Открыть" – Ctr+A, "Сохранить" – Ctr+C, "Закрыть" – F4 ,"О программе" – F11.

На странице Dialogs выбираем компонент OpenDialog и SaveDialog.

Для выбора нужных файлов установить фильтры следующим образом: выбрав компоненту, дважды щелкнуть кнопкой мыши по правой части свойства Filter инспектора объектов, и в появившемся окне FilterEditor, в левой части записать текст, характеризующий выбор, в правой части – маску. Для OpenDialog1 и SaveDialog 1 записываем в левой части формат \*.ole и \*.\*. Формат \*.ole означает, что будут видны все файлы с расширением ole, а формат \*.\* – будут видны все файлы (с любыми именами и расширениями).

Добавляем себе на форму компонент OleContainer. Задаем свойства значений Align – alTop.

Теперь перейдем к программированию. Введем в определении класса формы глобальную переменную FName, в которой будет храниться имя файла объекта OLE: AnsiStringFName.

Процедура, соответствующая разделу меню Новыйпоказана в листинге 2.1.

Листинг2.1 – Код программы раздела меню Новый

void \_\_fastcall TForm1::MNewClick(TObject \*Sender)

{ AnsiStringFName;

if (OleContainer1->InsertObjectDialog() )

{

FName= " ";

OleContainer1->DoVerb (ovShow);

}}

ВызываемымэтимоператоромметодInsertObjectDialogосуществляетобращениекстандартномуокнуWindowsInsertObject (вставкаобъекта), вкоторомпользовательможетуказатьтипвставляемогообъекта, инициализируетобъектOLEизагружаетеговконтейнерOleContainer1. Вызываемый далее методDoVerb обеспечивает немедленное открытие программы, связанной со вставленным документом.

Процедура, соответствующая разделу меню Закрыть, показана в листинге 2.2.

Листинг2.2 – Код программы раздела меню Закрыть

void \_\_fastcallTForm1::MCloseClick(TObject \*Sender)

{

OleContainer1->DestroyObject () ;

}

Эта процедура разрушает объект в контейнере OLE.

Процедура, соответствующая разделу меню Сохранить, показана в листинге 2.3.

Листинг2.3 – Код программы раздела меню Сохранить

void \_\_fastcallTForm1::MSaveClick(TObject \*Sender)

{ AnsiStringFName;

if (FName == "")

if (SaveDialog1->Execute () )

FName = SaveDialog1->FileName;

else return;

 OleContainer1->SaveToFile(ChangeFileExt (FName, " .ole") ) ;

}

Процедура, соответствующая разделу меню Открыть, показана в листинге 2.4.

Листинг 2.4 – КодпрограммыразделаменюОткрыть

void \_\_fastcall TForm1::MOpenClick(TObject \*Sender)

{ AnsiStringFName;

if (OpenDialog1->Execute () )

{

OleContainer1->LoadFromFile (OpenDialog1->FileName) ;

FName = OpenDialog1->FileName;

OleContainer1->DoVerb (ovShow) ; }

}

Она обычным образом вызывает диалоговое окно, в котором пользователь выбирает открываемый файл. Затем объект, соответствующий этому файлу, загружается в контейнер методом LoadFromFile. Имя файла запоминается в переменной FName. Последний оператор методом DoVerb открывает программу, связанную с загруженным объектом.

При выборе из меню ? команды О программе на экране должно появиться одноименное окно.

Вывод окна О программе выполняет функция обработки события click, которое происходит в результате выбора из меню ? команды О программе (листинг 2.5). Непосредственно вывод окна выполняет метод showModal , который выводит окно как модальный диалог. Модальный диалог перехватывает все события, адресованные другим окнам приложения, в том числе и главному. Таким образом, пока модальный диалог находится на экране, продолжить работу с приложением, которое вывело модальный диалог, нельзя.

Листинг2.5. ВыводокнаОпограмме

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{

Form2->Top=Form1->Top+ Form1->Height/2-Form2->Height/2;

Form2->Left=Form1->Left+ Form1->Width/2-Form2->Width/2;

Form2->ShowModal();

}

Продолжим добавлять компоненты на Form1. Компонент Image1. Используем свойство Picture, чтобы вставить графический файл Модель Солнечной системы.jpg. Задаем свойству значения Align – alRight.

Добавим поверх компонента Image1 еще Image2, Image3, Image4, Image5, Image6, Image7, Image8, Image9, Image10 так, чтобы их области были поверх изображений планет. Заполним свойство значения Hint – Уран, Земля, Солнце, Марс, Венера, Меркурий, Юпитер, Сатурн, и Нептун соответственно. А свойство значения ShowHint установим в true. Таким образом, мы добились того когда пользователь приложения будет наводить на планету курсор, у него будет выплывающая подсказка в виде названия планеты.

 Добавим на нашу форму еще кнопку Button. Для этого вформу разрабатываемого приложения добавить командную кнопку Button1, надо в палитре компонентов, на вкладке Standard,щелкнуть на значке компонента Button, установить курсор в ту точку формы, в которой должен быть левый верхний угол компонента, и еще раз щелкнуть кнопкой мыши. В результате на форме появляется компонент Button1 - кнопка. Путем изменения значения свойства Name может изменить имя компонента. Однако в простых программах имена компонентов, как правило, не изменяют.

После добавления к форме командной кнопки нужно установить значения свойств Name – имя компонента. Используется в программе для доступа к компоненту и его свойствам; Caption – текст на кнопке;Enabled – Признак доступности кнопки. Кнопка доступна, если значение свойства равно true, и недоступна, если значение свойства равно false. Установим значение свойства Caption – Выход, Enabled – true.

Теперь пропишем код для командной кнопки Button1 как показано в листинге 2.6.

Листинг2.6. Выходиз приложения

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

 Form1->Close();

}

Эта процедура закроет приложение.

Полный код программы «Модель Солнечной системы» показан в приложении А.

**2.2 Создание окна О программе**

Чтобы программа во время своей работы могла вывести на экран окно, отличное от главного (стартового), нужно добавить в проект форму. Делается это выбором из меню File команды Newform . В результате выполнения команды Newform в проект добавляется новая форма и соответствующий ей модуль.

 Если в проекте несколько форм, то для того чтобы получить доступ к нужной форме и, соответственно, к модулю, надо выбрать имя нужной формы в списке диалогового окна ViewForm, которое становится доступным в результате щелчка на командной кнопке ViewForm или нажатия комбинации клавиш <Shift>+<F12>.

Добавим в новую форму компоненты Image1, Label1, Label2 и Button1.

Компонент Image расположен на странице Additionol библиотеки компонентов. Он обеспечивает вывод на поверхность формы иллюстраций, представленных в формате BMP (чтобы компонент можно было использовать для отображения иллюстраций в формате JPG, надо подключить модуль JPEG — включить в текст программы директиву # include<jpeg.hpp>). Для этого используем свойство Picture. Нажмем на кнопку с многоточием около этого свойства или просто сделаем двойной щелчок на Image, и перед нами откроется окно PictureEditor, позволяющее загрузить в свойство Picture какой-нибудь графический файл (кнопка Load), а также сохранить открытый файл под новым именем или в новом каталоге. Щелкаем на Load, чтобы загрузить графический файл Рисунок.jpg. После загрузки файла щелкаем ОК и в нашем компоненте Image отобразиться выбранная компонента. Задаем свойству значения Proportional – true. Признак автоматического масштабирования картинки без искажения, а чтобы масштабирование было выполнено, значение свойства AutoSize ставим false.

Компонент Label, расположен на странице Standard библиотеки компонентов (рисунок 2.1), предназначен для вывода текста на поверхность формы.

**Рисунок 2.1 – Компонент Label**

Добавляем компоненты Label1 и Label2 на форму. Свойства компонент определяют вид и расположение текста. Свойство Caption – отображаемый текст. Для компоненты Label1 отображаемый текст будет название приложение, а для Label2 – о программе. Alignment – задает способ выравнивания текста внутри поля. Текст может быть выровнен по левому краю (taLeftJustify), по центру (taCenter) или по правому краю (taRightJustify). Font – шрифт, используемый для отображения текста. Уточняющие свойства определяют шрифт (Name), размер (size), стиль (style) и цвет символов (Color). Wordwrap – признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку (значение свойства AutoSize должно быть false).

Вид формы Form2 после добавления необходимых компонентов приведен на рисунке 2.2, значения ее свойств — в таблице 2.1. Код программы «О программе» показан в приложении Б.

**Рисунок 2.2 – Форма О программе**

**Таблица 2.1 Значения свойств формы О программе**

|  |  |
| --- | --- |
| Свойство | Значение |
| Name | Form2 |
| Caption | Oпрограмме |
| BorderStyle | bsSingle |
| Borderlcons.biSystemMenu | false |
| Borderlcons.biMininize | false |
| Borderlcons.biMaximize | false |

**2.3 Сохранение приложения**

Приложение готово теперь можно его сохранить. Сохраняем проект.

Проект *—* это набор файлов, используя которые компилятор создает выполняемый файл программы (ехе-файл). В простейшем случае проект составляют: файл описания проекта (bpr-файл), файл главного модуля (срр-файл), файл ресурсов (res-файл), файл описания формы (dfm-файл), заголовочный файл формы (h-файл) и файл описания функций формы (срр-файл).

Чтобы сохранить проект, нужно в меню File выбратькомандуSave Project As. Если проект еще ни разу не был сохранен, то C++ Builder сначала предлагает сохранить модуль (содержимое окна редактора кода) и поэтому на экране появляется окно SaveUnitlAs. В этом окне надо выбрать папку, предназначенную для проектов, создать в ней папку для сохраняемого проекта, открыть ее и ввести имя модуля. В результате щелчка на кнопке OK в указанной папке будут созданы три файла: срр, h и dfm, и на экране появится диалоговое окно SaveProjectlAs, в которое надо ввести имя проекта.

Обратите внимание, что имена файла модуля (срр) и файла проекта (bpr) должны быть разными, т. к. C++ Builder в момент сохранения файла проекта создает одноименный срр-файл (файл главного модуля). Кроме того, надо учесть, что имя генерируемого компилятором выполняемого файла совпадает с именем проекта. Поэтому файлу проекта следует присвоить такое имя, которое, по вашему мнению, должен иметь выполняемый файл программы, а файлу модуля — какое-либо другое имя, например, полученное путем добавления к имени проекта порядкового номера модуля.

**2.4 Компиляция и запуск программы**

Процесс преобразования исходной программы в выполняемую состоит из двух этапов: непосредственно компиляции и компоновки. На этапе компиляции выполняется перевод исходной программы в некоторое внутреннее представление. На этапе компоновки выполняется сборка (построение) программы.

После ввода текста функции обработки события и сохранения проекта можно, выбрав в меню Projectкоманду CompileUnit, выполнить компиляцию. Процесс и результат компиляции отражается в диалоговом окне Compiling (рисунок 2.3). Если, в программе нет синтаксических ошибок, то окно будет содержать сообщение: Done: CompileUnit, в противном случае будет выведено сообщение Done: Thereareerrors.

**Рисунок 2.3 – Результат компиляции: в программе нет ошибок**

В случае если компилятор обнаружит в программе ошибки и неточности, диалоговое окно Compilingбудет содержать информацию о количестве синтаксических (Errors) и семантических (Warnings) ошибок, а также о числе подсказок (Hints). Сами сообщения об ошибках, предупреждения и подсказки находятся в нижней части окна редактора кода.

Чтобы перейти к фрагменту кода, который, по мнению компилятора, содержит ошибку, надо выбрать сообщение об ошибке (щелкнуть в строке

сообщения левой кнопкой мыши) и из контекстного меню выбрать команду EditSource.

Процесс компиляции можно активизировать, выбрав в меню Runкоманду Run**,** которая запускает разрабатываемое приложение. Если будет обнаружено, что с момента последней компиляции в программу были внесены изменения или программа еще ни разу не компилировалась, то будет выполнена компиляция, затем — компоновка, и после этого программа будет запущена (естественно, только в том случае, если в программе нет ошибок).

Пробный запуск программы можно выполнить непосредственно из среды разработки, не завершая работу с C++ Builder. Для этого нужно в меню Runвыбрать команду Runили щелкнуть на командной кнопке Run(рисунок 2.4)

**Рисунок 2.4 – Запуск программы из среды разработки**

**2.5 Настройка приложения**

После того как программа отлажена, необходимо выполнить ее окончательную настройку: задать название программы и значок, который будет изображать исполняемый файл приложения в папке, на рабочем столе и на панели задач, во время работы программы.

Название программы отображается во время ее работы в панели задач Windows, а также в заголовках окон сообщений, выводимых функцией ShowMessage.

****Название программы надо ввести в поле Title(рисунок 2.5) вкладки Applicationдиалогового окна ProjectOptions,которое появляется в результате выбора в меню Projectкоманды Options.

**Рисунок 2.5 – Название программы надо ввести в поле Title**

Чтобы назначить приложению значок, отличный от стандартного, нужно в меню Projectвыбрать команду Optionsи в открывшемся окне на вкладке Applicationщелкнуть на кнопке LoadIcon. В результате этих действий откроется стандартное окно, используя которое можно просмотреть каталоги и найти подходящий значок (значки хранятся в файлах с расширением ico).

В состав C++ Builder входит утилита ImageEditor (Редактор изображений), при помощи которой программист может создать для своего приложения уникальный значок. Запустить ImageEditor можно из C++ Builder, выбрав в меню Toolsкоманду ImageEditor,или из Windows — командой Пуск | Программы | BorlandC++ Builder | ImageEditor.

Чтобы начать работу по созданию нового значка, нужно в меню Fileвыбрать команду New | IconFile(рисунок 2.6).

После выбора типа создаваемого файла открывается окно IconProperties, в котором необходимо выбрать характеристики создаваемого значка: Size(Размер) – 32x32 (стандартный размер значков Windows) и Colors(Палитра) – 16 цветов. В результате нажатия кнопки ОКоткрывается окно Iconl.ico, в котором можно, используя стандартные инструменты и палитру, нарисовать нужный значок.

**Рисунок 2.6 – Начало работы над новым значком**

В процессе создания картинки можно удалить (стереть) ошибочно нарисованные элементы, закрасив их прозрачным цветом, которому на палитре соответствует левый квадрат в нижнем ряду.

Кроме "прозрачного" цвета в палитре есть "инверсный" цвет. Нарисованные этим цветом части рисунка при выводе на экран будут окрашены инверсным относительно цвета фона цветом.

Чтобы сохранить нарисованный значок, надо в меню Fileвыбрать команду Save,в открывшемся диалоговом окне раскрыть папку проекта (приложения, для которого создан значок) и задать имя файла значка, которое обычно совпадает с именем проекта (выполняемого файла приложения).

По окончании работы над проектом удалим не нужные файлы. Файлы с расширением tds и obj создает компилятор в процессе генерации выполняемого файла. Это временные файлы. Они не нужны, и их можно удалить, тем более что размер tds-файла может достигать нескольких мегабайт. Файлы, расширение которых начинается значком ~, - это резервные копии соответствующих файлов проекта. Их тоже можно удалить.

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсовой работы было разработано обучающее приложение «Модель Солнечной системы» в среде программированияС++ Builder.

При разработке приложения был создан GUI интерфейс
приложения – модели Солнечной системы, используя OLE-технологии.

С помощью разработанной программы можно наглядно изучать строение Солнечной системы, пользоваться справочным материалом физических характеристик планет. Так же можно создавать новые файлы в приложении, т.е. совершенствовать его.

В дальнейшем планирую свое приложение использовать на уроках астрономии, как наглядно-информационный материал и совершенствовать его.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. <http://allrefs.net/c21/4ecab/p38/>
2. <http://www.intuit.ru/studies/courses/13745/1221/lecture/23310>
3. Архангельский, А.Я. – Программирование в C++Builder 6: Научно-техническое издание / А.Я. Архангельский. – М.: Бином, 2003. – 1152 с.ЗАО «Издательство БИНОМ», 2003 г.
4. [http://www.studfiles.ru](http://www.studfiles.ru/preview/1544685/)

**ПРИЛОЖЕНИЕ А**

Листинг2.7 – Программа «Модель Солнечной системы»

#include<vcl.h>*/\*подключает заголовочный файл vcl.h, содержащий объявления, используемые в библиотеке визуальных компонентов C++Builder\*/*

#pragmahdrstop\*/*вызывает действия, зависящие от указанной опции\*/*

#include "Unit1.h" */\* включает файл Unit1.h, который ищется прежде всего в каталоге, в котором расположен файл, содержащий данную директиву\*/*

#include "Unit2.h*"/\* добавляет содержимое заданного файла в другой файл\*/*

*//сюда размещаются дополнительные директивы подключения модулей*

#pragma package(smart\_init)

#pragma resource "\*.dfm"

*//Объявление объекта формы*

TForm1 \*Form1;

*// вызовконструктораформы*

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

 : TForm(Owner)

{

}

void \_\_fastcall TForm1::MNewClick(TObject \*Sender)

*/\*процедура исполняемая при нажатии в меню пункт Новый MNew на форме TForm1\*/*

{ AnsiStringFName;

if (OleContainer1->InsertObjectDialog() )

{

FName= " ";

OleContainer1->DoVerb (ovShow) ;

}}

*/\*процедураисполняемаяпринажатиивменюпунктОткрытьMOpenнаформе TForm1\*/*

void \_\_fastcall TForm1::MOpenClick(TObject \*Sender)

{ AnsiStringFName;

if (OpenDialog1->Execute () )

{

OleContainer1->LoadFromFile (OpenDialog1->FileName) ;

FName = OpenDialog1->FileName;

OleContainer1->DoVerb (ovShow) ; }

}

*/\*процедураисполняемаяпринажатиивменюпунктСохранитьMSaveнаформе TForm1\*/*

void \_\_fastcall TForm1::MSaveClick(TObject \*Sender)

{ AnsiStringFName;

if (FName == "")

if (SaveDialog1->Execute () )

FName = SaveDialog1->FileName;

else return;

 OleContainer1->SaveToFile(ChangeFileExt (FName, " .ole") ) ;

}

*/\*процедура исполняемая при нажатии в меню пунктЗакрытьMClose на форме TForm1\*/*

void \_\_fastcall TForm1::MCloseClick(TObject \*Sender)

{

OleContainer1->DestroyObject () ;

}

*/\*процедураисполняемаяпринажатиикоманднойкнопкиButton1 Закрыть \*/*

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

 Form1->Close();

}

*/\*процедура исполняемая при нажатии в меню пунктО программе на форме TForm1\*/*

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{

Form2->Top=Form1->Top+ Form1->Height/2-Form2->Height/2;

Form2->Left=Form1->Left+ Form1->Width/2-Form2->Width/2;

Form2->ShowModal();

}

**ПРИЛОЖЕНИЕ Б**

Листинг 2.8 – Программа «О программе»

#include<vcl.h>

#pragma hdrstop

#include "Unit2.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

\_\_fastcall TForm2::TForm2(TComponent\* Owner)

: TForm(Owner)

{

}

void \_\_fastcall TForm2::Button1Click(TObject \*Sender)

{

ModalResult = mrOk; *// убрать окно О программе*

}

***ПРИМЕР* 2 Курсовая работа оценка 6-7**

Характеристика и замечания:

Хорошее оформление, введение правильное и хорошее.

Теоретическая часть плохая очень мала и перемешена с практической

Практическая часть хорошая не хватает только

1 Обоснования выбора визуальной среды и

2 Описания используемых компонент

Хорошо раскрыта ТЕМА применения OLE – технологии.

Заключение слишком хорошее.

Хорошо сделаны Список использованных источников и

Содержание электронного носителя (диска).

|  |
| --- |
| **МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ****УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ "ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"****факультет повышения квалификации и переподготовки кадров**  |
| Курсовая работа**ПРИМЕНЕНИЕ OLE КОМПОНЕНТА В GUI-ИНТЕРФЕЙСЕ MDI- ПРИЛОЖЕНИЙ** |
| Студент гр. ИС-13 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, датаРуководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шедько подпись, датаРабота защищена «\_\_» \_\_\_\_\_\_\_\_\_\_ 201\_\_ г. с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_».Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ подпись Фамилия И.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ подпись Фамилия И.О.Витебск, 2015 |
|  |

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ "ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"**

|  |  |
| --- | --- |
| Факультет *факультет повышения квалификации и переподготовки кадров*Кафедра *информатики*  | «Утверждаю»зав. кафедрой \_\_\_\_\_\_\_\_\_\_ доц.Казаков В.Е.«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. |

**ЗАДАНИЕ**на курсовую работу

Студенту *2* курса

 Фамилия И.О.

1. Тема работы: *Применение OLE компонента в GUI- интерфейсе MDI-приложений*

2. Срок сдачи студентом законченной работы: *21 марта 2015 года*

3. Исходные данные к работе

4. Перечень подлежащих разработке в курсовой работе вопросов или краткое содержание курсовой работы:

а) Применение OLEконтейнера в GUI интерфейсе

б) Разработка MDI приложения

в) Разработка и реализация GUI интерфейса

5. Перечень графического или программного материала (с точным указанием обязательных чертежей) или модулей программы

6. Консультанты по работе с указанием относящихся к ним разделов

7. Дата выдачи задания «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_ /

подпись                            Фамилия И.О.

Задание принял к исполнению «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 подпись студента

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение……………………………………………………………………… | 4 |
| 1. Понятие OLE……………………………………………………………… | 5 |
| 2. Использование OLE………………………………………………………. | 7 |
| 2.1 Внедрение объекта ……………………………………………………… | 11 |
| 2.2 Перетаскивание объекта из одного приложения в другое……………. | 13 |
| 2.3 Связывание объекта………………………………… ………………….. | 15 |
| 3. Разработка MDI приложения………………… …………………………. | 17 |
| 4. Разработка и реализация графического интерфейса………………..….. | 21 |
| 4.1 Понятие графического интерфейса……………………………………. | 21 |
| 4.2 Применение OLE компонента в GUI интерфейсе……………………... | 25 |
| 4.3 Описание пользовательского интерфейса……………………………... | 27 |
| Заключение ………………………………………………………………….. | 28 |
| Список использованных источников ……………………………………… | 29 |

**ВВЕДЕНИЕ**

В прошлом инструментальные средства разработки программного обеспечения были практически одинаковы, и отличались, в основном, способностью обмена информацией с другими инструментальными средствами. Системы программирования можно было рассматривать как острова, изолированные друг от друга. При создании приложений использовались методы структурного программирования.

С появлением OLE в разработке программного обеспечения был осуществлен переход от процедурного к объектно-ориентированному программированию. С помощью OLE появилась возможность использовать объекты среды программирования и объекты, принадлежащие другим прикладным программам. Сущность технологии OLE состоит в использовании элементов управления или приложений которые уже разработаны и его функциональные возможности можно позаимствовать для собственных целей.

Целью работы является изучить возможность использование OLE компонента в GUI интерфейсе MDI приложений.

Для достижения поставленной цели необходимо решить следующие задачи: разработать приложение, в котором будет применен OLE контейнер; изучить возможности и способ реализации MDI приложения в среде программирования; создать GUI интерфейс, изучить его элементы, которые являются элементами управления визуальной среды программирования.

В данной работе в качестве среды программирования использована среда разработки программного обеспечения Microsoft Visual Basic 6.0. Visual Basic сочетает в себе процедуры и элементы объектно-ориентированных и компонентно-ориентированных языков программирования. Visual Basic позволяет скрыть от пользователя многие подробности использования объектов и управления ими.

Этот язык был выбран мной из-за: высокой скорости создания приложения с графическим интерфейсом, простого синтаксиса, который позволяет быстро освоить язык.

Основными источниками для получения информации при разработке MDI приложения с GUI интерфейсам и применением OLE компонента использовались интернет сайты, такие как microsoft.com, wikipedia.org и другие, а также книги Евангелос Петрусос «Visual Basic 6. Руководство разработчика» и в которых излагается информация для изучения возможностей языка Visual Basic и принципах реализации GUI интерфейс в MDI приложении.

**1. ПОНЯТИЕ OLE**

OLE (англ. Object Linking and Embedding) — технология связывания и внедрения объектов в другие документы и объекты [10].

OLE позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад. Например, установленная на персональном компьютере издательская система может послать некий текст на обработку в текстовый редактор, либо некоторое изображение в редактор изображений с помощью OLE технологии.

Основное преимущество использования OLE в том, что она позволяет создать главный файл, картотеку функций, к которой обращается программа. Этот файл может оперировать данными из исходной программы, которые после обработки возвращаются в исходный документ.

OLE используется при обработке составных документов (англ. compound documents), может быть использована при передаче данных между различными несвязанными между собой системами посредством интерфейса переноса (англ. drag-and-drop), а также при выполнении операций с буфером обмена. Идея внедрения широко используется при работе с мультимедийным содержанием на веб-страницах, где используется передача изображения, звука, видео, анимации в страницах HTML (язык гипертекстовой разметки) либо в других файлах, также использующих текстовую разметку (например, XML и SGML). Однако, технология OLE использует архитектуру сетевого ПК с избыточными вычислительными ресурсами. Это означает, что тип файла либо программа, которую пытаются внедрить, должна присутствовать на машине клиента. Например, если OLE оперирует таблицами Microsoft Excel, то программа Excel должна быть инсталлирована на машине пользователя.

OLE 1.0 был выпущен в 1990 году на основе технологии DDE (Dynamic Data Exchange), использовавшейся в более ранних версиях операционной системы Microsoft Windows. В то время как технология DDE была сильно ограничена в количестве и методах передачи данных между двумя работающими программами, OLE имел возможность оперировать активными соединениями между двумя документами либо даже внедрить документ одного типа в документ другого типа.

OLE сервера и клиенты взаимодействуют с системными библиотеками при помощи таблиц виртуальных функций (англ. virtual function tables, VTBL). Эти таблицы содержат указатели на функции, которые системная библиотека может использовать для взаимодействия с сервером или клиентом. Библиотеки OLESVR.DLL (на сервере) и OLECLI.DLL (на клиенте) первоначально были разработаны для взаимодействия между собой с помощью сообщения WM\_DDE\_EXECUTE, предоставляемого операционной системой.

OLE 1.1 позднее развился в архитектуру COM (component object model) для работы с компонентами программного обеспечения. Позднее архитектура COM была преобразована и стала называться DCOM.

Когда объект OLE помещен в буфер обмена информацией, он сохраняется в оригинальных форматах Windows (таких как bitmap или metafile), а также сохраняется в своём собственном формате. Собственный формат позволяет поддерживающей OLE программе внедрить порцию другого документа, скопированного в буфер, и сохранить её в документе пользователя.

Следующим эволюционным шагом стал OLE 2.0, сохранивший те же цели и задачи, что и предыдущая версия. Но OLE 2.0 стал надстройкой над архитектурой COM вместо использования VTBL. Новыми особенностями стали автоматизация технологии drag-and-drop, in-place activation и structured storage.

В 1996 году Microsoft переименовала технологию OLE 2.0 в ActiveX. Были представлены элементы управления ActiveX, ActiveX документы и технология Active Scripting. Эта версия OLE в основном используется веб-дизайнерами для вставки в страницы мультимедийных данных.

Подводя итог, можно сделать вывод, что OLE является набором средств, позволяющим легко подготавливать документы, включающие в себя данные, подготовленные в различных приложениях. Чтобы вы могли объединить данные, подготовленные в различных приложениях, необходимо, чтобы эти приложения поддерживали технологию OLE. Стандартные приложения Windows - Paintbrush, Write, Sound Recorder, Cardfile, Object Pakager поддерживают OLE. Кроме стандартных приложений Windows, многие другие приложения, разработанные фирмой Microsoft и другими независимыми фирмами, включают в себя поддержку OLE-технологии. Microsoft Word for Windows 2.0 и 6.0, Microsoft Excel 4.0 и 5.0, ZSoft PhotoFinish 2.0, Designer, FoxPro for Windows, Access и многие другие пакеты включают поддержку OLE-технологии.

**2. ИСПОЛЬЗОВАНИЕ OLE**

Связывание и внедрение объектов позволяет использовать информацию из одного приложения в другом. Для использования технологии OLE необходимо, чтобы и исходное приложение, и целевое приложение поддерживали OLE.

С помощью технологии связывания и внедрения объектов (OLE), реализованной в Microsoft Windows, можно копировать и размещать информацию из одного приложения в другое, сохраняя возможность редактирования ее в исходном приложении.

Разница между внедренными и связанными объектами состоит в том, где и как размещаются данные представляющие объект. Внедренный объект хранится в самом приложении и является его составной частью. Для связанного объекта храниться только ссылка на данные, которые могут находиться в другом приложении.

Приложения, которые создают объекты для передачи, называются OLE-серверами [3]. Это приложение, функциональные возможности которого доступны для приложения, разработанного в Visual Basic

 Приложения, которые позволяют внедрить или связывать чужие объекты называется OLE-клиентами [3]. Это приложение, которое содержит OLE- объекты.

OLE-объект – это объект, который делается доступным для других приложений с помощью OLE-сервера [3]. Приложение OLE-сервер делает доступными объекты различных типов. OLE-объекты используются в приложениях OLE-клиент. Объекты могут быть связанны (linked) или встроены (embedded).

Чтобы включить функциональные возможности OLE в VB-приложении, необходим элемент управления OLE Container, позволяющий вставлять в программу объекты из других приложений. В элемент управления OLE Container можно поместить только один объект. На форму можно поместить несколько элементов OLE Container, каждый из которых может содержать собственный объект.

Способ использования OLE в моем приложении – это применение элемента OLE Container или просто OLE. Этот объект позволит встроить и связать OLE-объекты в моей программе. Элемент управления OLE Container – это доступ к различным объектам операционной системы, которые Visual Basic не в состоянии использовать самостоятельно (например, документы Word, звуковые файлы, растровые изображения или электронные таблицы Excel). При добавлении OLE Container на форму Visual Basic отображает диалоговое окно Insert Object (Вставка объекта) в котором можно указать объект помещаемый в элемент OLE Container.

Элемент управления OLE Container может содержать любой из объектов, перечисленных в диалоговом окне Insert Object (Вставка объекта) как на рисунке 2.1, и сделать программу на Visual Basic приложением-контейнером.



**Рисунок 2.1 – окно вставки OLE-объекта**

Преимущество использования OLE Container является то, что он скрывает сложности, связанные с применением OLE.

При создании OLE-объекта в форме окна Properties (Свойства) будет содержаться следующая информация:

Имя прикладной программы с помощью которой создан объект (свойство Class);

Данные или ссылка на данные (свойство SourceDoc);

Образ данных (свойство SourceItem), данное значение устанавливается только при связывании объекта и определяет данные связанные с управлением OLE.

В окне Properties (Свойства) есть свойство OLETypeAllowed. От установки значения этого свойства зависит, будет исходный документ внедрен или связан с элементом управления OLE Container.

 При значении Either(2) свойства OLETypeAllowed элемента управления OLE Container может содержать либо встроенный, либо связанный объект. Это свойство установлено по умолчанию.

Значение Linked(0) определят, что элемент управления может содержать только связанный объект.

Значение Embedded(1) означает, что элемент управления OLE Container может содержать только встроенный объект.

Значение свойства OLETypeAllowed по умолчанию не задает тип создаваемого OLE-объекта, а лишь позволяет определить тип объекта, который может быть создан. Реально используемый тип OLE-объекта указан в диалоговом окне Insert Object (Вставка объекта). Если значение свойства OLETypeAllowed равно Either(2), то в диалоговом окне Insert Object (Вставка Объекта) переключатель Create New (Создать новый) и Create from File (Создать из файла), а также флажок Link (Связь) становятся доступными это видно на рисунке 2.2.



**Рисунок 2.2 – Окно вставки OLE-объекта при значении Either(2) свойства OLETypeAllowed**

Если же значение свойства OLETypeAllowed равно Linked(0), то переключатель Create New (Вставка объекта) блокируется (нельзя связаться с несуществующим объектом). Окно вставки OLE-объекта отображено на рисунке 2.3.



**Рисунок 2.3 – Окно вставки OLE-объекта при значении Linked(0) свойства OLETypeAllowed**

Если устанавливается значение Embedded(1), то флажок Link (Связь) пропадает из диалогового окна Insert Object (Вставка Объекта), как на рисунке 2.4.



**Рисунок 2.4 – Окно вставки OLE-объекта при значении Embedded(1) свойства OLETypeAllowed**

**2.1 Внедрение**

Используя внедрение, можно вставить объект из одного приложения (приложения-сервера) в другое (приложение-контейнер). Встроенный объект является копией оригинала, им можно манипулировать независимо от оригинала.

Чтобы вставить объект OLE в приложение, надо всего лишь поместить в соответствующую форму элемент управления OLE Container. После добавления элемента управления OLE в форму отображается диалоговое окно вставки (Insert Object) OLE-объекта как на рисунке 2.1. С помощью этого окна можно создать новый объект или создать его из файла, указав, будет ли это связанный или внедренный объект. Если установить переключатель в положение Create New (Вставка объекта), то на экране появится прокручивающийся список объектов, доступных системе (рисунок 2.1). Каждое приложение, выполняющее функции приложения-сервера, регистрирует предоставляемые объекты в системе, чтобы приложения-контейнеры могли ими воспользоваться. Когда вставляется объект, приложение для работы с ним запускается автоматически при открытии каждого нового документа. При выборе из списка объекта приложение-контейнер устанавливает связь с приложением-сервером и выведет на экран его интерфейс пользователя. На экране появиться панель инструментов приложения-сервера. Теперь можно работать с приложением-сервером пользуясь его средствами, но процесс будет происходить в приложении-контейнере. Это показано на рисунке 2.5. При этом не нужно запускать другое приложение создания объекта, а затем копировать этот объект в наше приложение.



**Рисунок 2.5 – приложение-контейнер со встроенным объектом**

Не зависимо от того, как был вставлен объект – новый или уже существующий – для его редактирования достаточно выполнить двойной щелчок в области OLE. Операционная система обращается к Реестру, чтобы найти приложение, ассоциированное с вставленным объектом. Будет запущена приложение-сервер. Теперь можно отредактировать объект, а затем возвратиться в приложение-контейнер.

При внедрении OLE-объект, помещенный в приложение-контейнер и объект, содержащийся на дисковом файле – это два отдельных объекта. Можно изменить этот объект на диске, заменить его другим, можно даже удалить его – на содержание приложения-контейнера это не повлияет. В приложении-контейнере содержится копия объекта

В данной работе OLE- объект добавлен динамически с помощью метода Create Embed. При вызове этого метода достаточно указать имя класса внедряемого объекта как в листинге 2.1.

Листинг 2.1 – Добавление OLE-объекта

OLE1.CreateEmbed vbNullString, "Word.Document"

**2.2 Перетаскивание объектов из одного приложения в другое**

При необходимости добавить объект из одного приложения в другое можно воспользоваться механизмом перетаскивания OLE.

Свойство перетаскивания (drag-and-drop) OLE – это, прежде всего операция копирования и вставки данных. Механизм перетаскивания OLE позволяет передать данные в пределах одного документа, между разными документами или между приложениями.

Перетаскивание между приложениями возможно, только если второе приложение поддерживает технологию OLE. Объекты, перенесенные таким образом, становятся внедренными, а не связанными. Обычное перетаскивание данных аналогично их последовательному вырезанию и вставке. Данные полностью удаляются из приложения-сервера и вставляются в приложение-контейнер. Если же перетаскивание производится при нажатой клавише CTRL, вместо вырезания осуществляется копирование. В приложении-контейнере создается копия данных, а исходный их вариант остается неизменным.

В данной работе перетаскивание с помощью механизма перетаскивания OLE осуществлено на Form5, изображенной на рисунке 2.6.



**Рисунок 2.6 – OLE Container поддерживающий механизм перетаскивания**

 Для этого на форму добавлен элемент OLE Container. В диалоговом окне Insert Object (Вставка объекта), которое отобразил Visual Basic для вставки объекта в элемент OLE Container, выбрана кнопка Cancel. Это означает, что в форму будет помещен пустой элемент управления OLE Container и можно будет перетащить туда OLE-объект. Для того чтобы в окно управления OLE Container поместить объект, установлено значения свойства OLEDropAllowed равным True. Объект можно просто взять из другого приложения, выделив его, а затем перетащить его, нажав левую кнопку мыши в любое место объекта.

**2.3 Связывание объекта**

Этот метод напоминает встраивание, за исключением того, что данные связаны с документом, из которого они переданы. Изменение объекта в приложении-сервере автоматически отражается в приложении-контейнере. При связывании происходит встраивание ссылки на объект, открытый в приложении-сервере. Каждый раз, когда открывается документ, содержащий связанный объект, приложение-контейнер связывается с приложением-сервером, которое и открывает последнюю версию связанного объекта.

Если в диалоговом окне Insert Object (Вставка объекта) установить переключатель Create from File (Создать из файла) и установить флажок Link (Связать), то объект в приложении-контейнере будет связан с исходным объектом.



**Рисунок 2.6 – Окно вставки связанного объекта**

При связывании объекта в системе будет только один объект. При этом в приложении-контейнере будет содержаться ссылка на него, а не копия объекта. Тот же объект может быть связан с другим приложением, возможно, с другими приложениями. Независимо от количества документов, ссылающихся на данный объект, в системе будет находиться один исходный объект. Если его изменить, то во всех приложениях, ссылающихся на него, появится уже измененная версия объекта.

Если объект, связанный, то его нельзя отредактировать из приложения-контейнера. Это можно сделать из приложения сервера. Сущность связывания заключается в том, что пользователь не может редактировать объект из какого-либо приложения. В память компьютера находится только один экземпляр объекта, и его можно изменить только с помощью программы, в которой он был создан. Как только объект будет изменен, все программы, имеющие на него ссылку, также изменятся.

Для связанного объекта в моей работе в диалоговом окне Insert Object (Вставка объекта) выбран переключатель Create from File (Создать из файла) и флажок Связь, что позволило связать объект.

**3. РАЗРАБОТКА MDI ПРИЛОЖЕНИЯ**

Термин MDI (Multiple Document Interface) дословно означает многодокументный интерфейс. Multiple Document Interface (MDI) – способ организации графического интерфейса пользователя, предполагающий использование оконного интерфейса, в котором большинство окон расположены внутри одного общего окна [9]. Общее окно называют родительским окном, а расположенные внутри – дочерними или подчиненными окнами. Этим он и отличается от SDI (одно-документный интерфейс) в котором окна располагаются независимо друг от друга. Вопрос о том, какой тип интерфейса предпочтителен – MDI или SDI – часто становится предметом обсуждений. Разработчики широко используют оба типа интерфейса.

В интерфейсе типа MDI общая панель меню и панель инструментов для всех дочерних окон. Все окна приложения можно прятать и показывать, сворачивать и разворачивать и проводить с ними другие манипуляции, как с одним окном. Дочерние окна можно размещать “каскадом”, “вертикально”, “горизонтально”.

MDI приложение содержит MDI форму (родительскую форму), которая служит контейнером для подчиненных форм. Подчиненная форма (дочерняя форма) – это обычная форма, у которой значение свойства MDIChild равно True. Во время выполнения подчиненные окна отображаются в пределах рабочего пространства формы родителя MDI (область в границах формы ниже заголовка и строк меню), это показано на рисунке 3.1.



**Рисунок 3.1 – Подчиненные формы внутри MDI формы**

При закрытии формы-контейнера закрываются все подчиненные формы.

В приложении MDI можно открыть много окон построенных по одному образцу. При этом в режиме проектирования создается один экземпляр формы. Для того чтобы в режиме выполнения открыть много окон в коде программы создается новые экземпляры подчиненной формы (Листинг 3.1).

Листинг 3.1 – Создание нового экземпляра дочерней формы

Private Sub mnufileNewForm\_Click()

Dim frmNewForm As New Form1 ’ Создаем новый экземпляр формы

frmCount = frmCount + 1

frmNewForm.Caption = "Анкета" + Str(frmCount)

frmNewForm.Show ’загрузка формы Form1

End Sub

В тексте объявлена переменная frmNewForm для ссылки на вновь созданный экземпляр окна. Затем с помощью метода Show новый экземпляр визуализируется в родительском окне. Для вычисления номера дочернего окна используется переменная frmCount. Этот код выполняется при выборе команды Создать новый из меню Файл родительского окна.

В проекте для реализации MDI приложения использовались следующие процедуры:

1. Создание формы MDI.

Выполнив команду Add MDI Form (Добавить MDI-форму) меню Project(Проект).

2. Создание дочерней формы приложения.

Для создания дочерней формы MDI была создана новая форма и установлена ее свойства MDIChild равно True.

3. Для реализации некоторых действий создано простое меню MDI-формы. Для проектирования меню использовался редактор меню Menu Editor (Редактор меню) изображенный на рисунке 3.1. Редактор создает меню для активного данный момент окна. Редактор меню вызывался следующим образом:

Командой Menu Editor (Редактор меню) меню Tools (Инструменты).



**Рисунок 3.1 – Окно редактора меню Menu Editor**

В меню содержатся следующие пункты меню: Анкета, Пример анкеты, Трудовое соглашение, Справка.

При выборе пункта меню Анкета, который содержит команды Создать новую и Окно можно создать новый экземпляр дочернего окна и упорядочить дочерние окна в родительском окне (Приложение А).

Пункт меню Пример анкеты вызывает Form2 (Листинг 3.2) на которой расположен элемент управления OLE Container при помощи которого происходит связывания документа Word.

Листинг 3. 2 – Программный код при нажатии на пункт меню Пример анкеты

Private Sub primer\_Click()

 Dim primer As New Form2

 primer.Caption = "Пример автобиографии"

 primer.Show ’загрузка формы Form2

End Sub

Пункт меню Трудовое соглашение загружает форму Form5 (Листинг 3.3). На ней расположен элемент управления OLE Container и осуществляется механизм перетаскивания OLE.

Листинг 3.3 – Программный код при нажатии на пункт меню Трудовое соглашение

Private Sub trud\_Click()

 Form5.Caption = "Трудовое соглашение"

 Form5.Show ’загрузка формы Form5

End Sub

При помощи пункта меню Справка можно получить информацию о программе и об авторе. При выборе пункта О программе вызывается Form3 содержащая информацию о программе (Листинг 3.4).

Листинг 3.4 – Программный код при нажатии на пункт меню О программе

Private Sub prog\_Click()

 Form3.Show ’загрузка формы Form3

End Sub

 Пункт меню Об авторе вызывает Form4 на которой располагается информация об авторе программы (Листинг 3.5).

Листинг 3.5 – Программный код при нажатии на пункт меню Об авторе

Private Sub avtor\_Click()

 Form4.Show ’загрузка формы Form4

End Sub

Чтобы закрыть приложения можно использовать пункт меню ВЫХОД (Листинг 3.6).

Листинг 3.6 – Программный код при нажатии на пункт меню ВЫХОД

Private Sub vihod\_Click()

 End

End Sub

На основании анализа данного раздела можно сделать вывод, что разработанное приложение является MDI приложением, так как большинство окон расположены внутри одного общего окна и во время выполнения подчиненные окна отображаются в пределах одного – MDI формы.

**4. РАЗРАБОТКА И РЕАЛИЗАЦИЯ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА**

**4.1 Понятие графического интерфейса пользователя**

Интерфейс – совокупность технических, программных и методических средств сопряжения в вычислительной системе пользователей с устройствами и программами, а также устройств с другими устройствами и программами.

Различают: интерфейс пользователя – набор методов взаимодействия компьютерной программы и пользователя этой программы; программный интерфейс - набор методов для взаимодействия между программами; физический интерфейс - способ взаимодействия физических устройств.

Рассмотрим более подробно пользовательский интерфейс – совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером.

Современными видами пользовательских интерфейсов являются: командный интерфейс, реализованный в виде технологии командной строки; GUI – интерфейс, где диалог с пользователем ведется с помощью графических образов - меню, окон, других элементов; SILK - интерфейс (Speech - речь, Image - образ, Language - язык, Knowlege - знание),где компьютер находит для себя команды, анализируя человеческую речь.

Графический интерфейс пользователя (англ. Graphical user interface, GUI) – разновидность интерфейса в котором элементы интерфейса представлены на дисплее, исполнены в виде графических изображений.[5]

В GUI пользователь имеет произвольный доступ ко всем видимым экранным объектам (элементам интерфейса) и осуществляет непосредственное манипулирование ими.

Графический интерфейс пользователя является частью интерфейса, в котором одна сторона представлена пользователям, другая – машиной или устройством и определяет взаимодействие с пользователем на уровне визуализированной информации [5]. Это означает, что GUI интерфейс позволяет осуществлять взаимодействие человека с компьютером в форме диалога с использованием окон, меню и элементов управления (диалоговых панелей, кнопок и так далее).

Одним из требований к хорошему графическому интерфейсу программной системы является предсказуемая робота системы, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

Таким образом, GUI-интерфейс определяет взаимодействие программы с пользователем на уровне визуализированной информации.

В разработанном MDI приложении реализован GUI-интерфейс в визуальной среде программирования Microsoft Visual Basic 6.0, соответственно диалог с пользователем в данной программе ведется на уровне визуализированной информации, а именно, с помощью графических образов - меню, окон, кнопок и других элементов. Команды подаются пользователем программе через эти графические образы.

 Основой для создания графического интерфейса разработанного проекта является объект форма. На ее основе создан интерфейс пользователя. На форме размещаются элементы интерфейса в нужном месте окна приложения. Элементы интерфейса имеют различные назначение. В данной работе используется 5 форм и одна MDI форма. В проекте используются следующие элементы GUI-интерфейса, являющимися элементами управления:

Label (метка) – для вывода данных и пояснительных текстов. В отличие от текстового окна в ней не предусмотрены возможности редактирования и прокрутки данных. В работе элемент Label помещен на Form3, Form4, Form5 что показано на рисунке 4.1, рисунке 4.2, рисунке 4.3 . Используется для предоставления информации пользователю.



**Рисунок 4.1 – Расположение элемента Label на Form3**



**Рисунок 4.2 – Расположение элемента Label на Form4**

****

**Рисунок 4.3 – Расположение элемента Label на Form 5**

Для добавления текста в Label использовалось свойство Caption. Для изменения шрифта его размера и начертания – свойство Font. Для выравнивания текста – свойство Alignment. С помощью свойство BackColor изменен цвет фона.

CommandButton (кнопк) – элемент управления предназначена для запуска событийных процедур, для чего используется событие Click, которое происходит при нажатии пользователем мыши на кнопку. Наиболее часто применяемый элемент в моем проекте. В данной работе используется на всех формах (Приложение В, Приложение Г, Приложение Д, Приложение Е).

Для изменения цвета фона кнопки использовалась свойство BackColor. Для добавления изображения на кнопку и смены цвета свойство Style установлено в Graphical. Чтобы добавить изображения нужно указать полный путь к изображению с помощью свойства Picture.

CommonDialog элемент управления расположен на Form1. Этот компонент позволяет использовать отдельно стандартные окна Windows. Прежде чем его использовать к проекту была подключена библиотека Microsoft Common Dialog Control 6.0.

OLE Container дает возможность включать OLE-объекты в приложения.

Важным элементом GUI является стандартное меню пользователя. В данном приложении также имеется меню пользователя, которое находится на MDIForm.

Все элементы интерфейса, представленные пользователю на форме, исполнены в виде графических изображений.

Пользователь с помощью устройств ввода осуществляет манипулирование элементами интерфейса: нажатие на кнопку, выбор пункта меню.

**4.2 Применение OLE компонента в GUI интерфейсе.**

Основным компонентом для работы с OLE в моем приложении является OLE Container. В разработанном GUI интерфейсе применяемый элемент управления OLE Container, является элементам управления (объектам формы) среды Microsoft Visual Basic 6.0.

Для добавления OLE Container необходимо выбрать этот элемент управления из панели элементов. После чего элемент OLE Container запрашивает тип объекта, который требуется поместить контейнер.

В разработанном приложении элемент управления OLE Container располагается на формах Form1, Form2, Form5. Рассмотрим элемент управления OLE Container применяемые в графическом интерфейсе данного приложения.

Главным элементам интерфейса окна Form1 является OLE1 при помощи которого на форму внедрен документ Word. Для создания нового документа в формате Word на форме помещена командная кнопка Command1 (значение свойства Сaption – «Создать»). При возникновении свойства Click в OLE1 создается новый документ (Рисунок 2.5) и появляется панель инструментов. Теперь в OLE1 можно поместить текст.

Чтобы сохранить добавленный документ на форме создана командная кнопка Command2 (значение свойства Сaption – «Сохранить») при нажатии на которую вызывается диалоговое окно.

Для удобства пользователя для кнопки Создать, изображенной на рисунке 4.3, установлено значение свойство Default равное True. Это означает, что кнопка окаймлена черной рамкой и реагирует на все нажатие клавиши Enter на клавиатуре. Для кнопки Сохранить значение свойство Enabled равное False. Это позволяет заблокировать кнопку на форме (Рисунок 4.3) и ограничения доступа к ней до выполнения операции Создать.



**Рисунок 4.4 – Активная кнопка Создать и ограничение доступа к кнопке Сохранить.**

Для того чтобы пользователю был виден весь документ помещенный в контейнер для OLE1 на Form1 были установлены свойства, что можно увидеть из листинга 4.1

Листинг 4.2 – Изменение размера элемента OLE Container

OLE1.SizeMode = 3 ’Размеры объекта изменяются так, чтобы он

 занял максимально возможную часть окна

 элемента управления OLE Container, сохранив

 при этом свои первоначальные пропорции

OLE1.DoVerb -1

На форме Form2 также расположен элемент управления OLE1 в данном случае контейнер, предназначен для связывания документа.

При нажатии на пункт меню Пример анкеты вызывается Form2 на которой расположен элемент управления OLE Container в нем расположен документ Word в котором находится информация нужная для пользователя. Для того чтобы автоматически изменялось окно чтобы документ полностью помещался в окно значение свойство SizeMode установлено равным AutoSize. Это позволяет пользователю увидеть весь документ в контейнере.

При нажатии на пункт меню Трудовое соглашение вызывает Form5, в этом окне располагается элемент управления OLE Container OLE1. С помощью этого элемента осуществляется механизм перетаскивания с одного приложения в другое. Нужно перетащить документ (или его часть) в пустой элемент управления OLE Container.

Для пояснения использования окна OLE контейнера на форме распложён элемент Label в котором содержится пояснительный текст.

Таким образом, элемент управления OLE Container является центральным элементом в разработанном графическом интерфейсе. С помощью этого элемента приложения осуществляет работу с текстовыми документами, что является основным для разработанного приложения.

**4.3 Описание пользовательского интерфейса**

Запуск программы осуществляется следующим образом: необходимо открыть папку, содержащую приложение, выполнить двойной щелчок левой клавишей манипулятора мышь на исполняемом файле с расширением exe.

После запуска программы на экран выводится рабочее окно с названием приложения. В пределах этого окна расположено еще одно окно с названием Анкета. При нажатии на кнопку Создать пользователь может создать анкету. Чтобы сохранить анкету нужно нажать на кнопку Сохранить. После чего перед пользователям появится диалоговое окно в котором вводится название файла. Можно закрыть окно анкеты нажав на кнопку Закрыть, при этом появиться окно, уточняющее, действительно ли пользователь желает закрыть это окно. Если пользователю нужно создать еще вариант анкеты необходимо воспользоваться пунктом меню Анкета Создать.

При помощи меню расположенном в рабочем окне пользователь может ознакомиться с перечнем документов, которые он может создать, познакомиться с примером написания анкеты. Получить сведения о программе и об авторе. Можно выйти из программы нажав на пункт меню выход.

При выборе пункта меню Пример анкеты перед пользователям появится окно, в котором расположен текст примера. При нажатии на кнопку Перейти к написанию окно Пример анкеты закрывается.

При выборе пункта меню Трудовое соглашения загружается окно с пустым элементом. Вверху окна помещен пояснительный текст. Для закрытия этого окна необходимо нажать кнопку Закрыть.

Для создания удобного интерфейса доступны только те элементы, которые необходимы пользователю в данный момент. Кнопки с одинаковыми функциями на разных формах расположены в одном и том же месте, например кнопка Закрыть. Все элементы выделены контрастным цветом по отношению к фону окон, так чтобы пользователь легко воспринимал и читал текст.

Таким образом, пользовательский интерфейс данного приложения реализован таким образом, что обеспечивает эффективное взаимодействие пользователя и приложения. Примененный OLE контейнер в графическом интерфейсе легок в использования и понятен пользователю.

**ЗАКЛЮЧЕНИЕ**

В результате выполненной работе разработано и реализовано MDI приложение с GUI интерфейсом и применением OLE компонента в среде программирования Microsoft Visual Basic 6.0.

Данное приложение предназначено для создания и хранения в электронном виде личного дела сотрудника. При этом некоторые документы необходимо создать, а некоторые уже готовые поместить на форму для редактирования, после чего их сохранить в папку личного дела сотрудника. Это реализовано с помощью применения OLE контейнера, что позволило работать с документами .doc без запуска приложения Word.

Разработанное приложение является MDI приложением, так большинство окон, разработанных по одному образцу, располагаются внутри одного окна.

В разработанном приложении реализован GUI интерфейс. Взаимодействие пользователя осуществляется в форме диалога с использованием элементов управления. Пользователь имеет произвольный доступ ко всем элементам интерфейса и манипулирует ими. В GUI интерфейсе большое значение занимает элемент управления OLE контейнер.

В приложении используются следующие графические образы элементов GUI интерфейса: CommanButton, OLE Container, Label, CommonDialog, меню.

Недостатком данного приложения является то, что не все документы для создания личного дела есть возможность создать в приложении, и нет возможности при необходимости вывести их на печать. Это означает, что данная программа имеет перспективы развития.

Таким образам в данном приложении реализованы возможности использования OLE контейнера в графическом интерфейсе MDI приложения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Быков В.Л. – Основы программирования на языке Visual Basic 6.0: пособие – Брест: БГТУ, 2002. – 360 с.
2. Глушаков С.В., Мельников В.В., Сурядный А.С. – Программирование в среде Windows. Учебный курс – Харьков: Фолио; М.: ООО “Издательство АСТ”, 2000. – 487 с.
3. Петрусос Е. – Visual Basic 6 Руководство разработчика: в 2 т./ Пер с англ. – К.: Издательская группа BHV, 2000. – т2 – 560 с.
4. Графический интерфейс пользователя [Электронный ресурс] – Режим доступа: http://academic.ru. Дата доступа: 18.01.2015.
5. Графический интерфейс пользователя [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org. Дата доступа: 16.02.2015.
6. Многодокументальный интерфейс [Электронный ресурс] – Режим доступа: http://codenet.ru. Дата доступа: 18.01.2015.
7. Работа с элементами управления [Электронный ресурс] – Режим доступа: http://visualprogs.ru Дата доступа: 23.01.2015.
8. Элементы управления OLE Container для пользователей Visual Basic 6.0 [Электронный ресурс] – Режим доступа: http://microsoft.com. Дата доступа: 23.01.2015.
9. Multiple document interface [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org. Дата доступа: 20.01.2015.
10. Object Linking and Embedding [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org. Дата доступа: 23.01.2015.
11. Visual Basic для начинающих [Электронный ресурс] – Режим доступа: http://codenet.ru. Дата доступа: 20.01.2015.

**Содержание электронного носителя**

На электронном носителе расположены следующие директории и файлы:

- Файл **«Базыко Я.И. ИС-12. Применение OLE компонента в GUI интерфейсе MDI приложений.doc»** представляет собой текстовый документ с курсовой работой.

- Директория **«Software»** содержит программное обеспечение, необходимое для запуска проекта.

- Директория **«Hello World Project»** содержит исходный код проекта **Проект1**.

- Директория **«Final 21-03-2015»** содержит финальную версию исполняемых файлов проекта **Проект1**.

- Файл **«Readme.txt»** содержит описание содержания электронного носителя.