**Оглавление**

Работа в Borland C++ Builder 3

Среда разработки C++ Builder 3

Компоненты C++ Builder 3

Свойства компонентов 4

События 4

Методы 5

Менеджер проектов 5

Создание приложений в C++ Builder 6

Компоненты Borland C++ Builder 8

Выбор компонентов для групповых операций 8

Установка разделяемых свойств компонентов 8

Изменение размера компонентов 8

Выравнивание компонентов 9

Работа с базами данных в Borland C++ Builder. 11

Компонент TDataSource 11

Компонент TTable 12

Компонент TField 15

Компонент TDBGrid 17

Поставка приложений, созданных с помощью С++ Builder 19

Поставка исполняемых файлов 19

Установка и настройка Borland Database Engine 19

Установка дополнительных компонентов приложения 21

Способы создания дистрибутивов 22

Создание дистрибутивов с помощью InstallShield Express 23

Список использованной литературы 27

### Работа в Borland C++ Builder

#### Среда разработки C++ Builder

C++ Builder представляет собой SDI-приложение, главное окно которого содержит настраиваемую инструментальную панель (слева) и палитру компонентов (справа). Помимо этого, по умолчанию при запуске C++ Builder появляются окно инспектора объектов (слева) и форма нового приложения (справа). Под окном формы приложения находится окно редактора кода.

Формы являются основой приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы элементов объектов C++ Builder, называемых компонентами. Компоненты C++ Builder располагаются на палитре компонентов, выполненной в виде многостраничного блокнота. Важная особенность C++ Builder состоит в том, что он позволяет создавать собственные компоненты и настраивать палитру компонентов, а также создавать различные версии палитры компонентов для разных проектов.

#### Компоненты C++ Builder

Компоненты разделяются на видимые (визуальные) и невидимые (невизуальные). Визуальные компоненты появляются во время выполнения точно так же, как и во время проектирования. Примерами являются кнопки и редактируемые поля. Невизуальные компоненты появляются во время проектирования как пиктограммы на форме. Они никогда не видны во время выполнения, но обладают определенной функциональностью (например, обеспечивают доступ к данным, вызывают стандартные диалоги Windows и др.)

Для добавления компонента в форму можно выбрать мышью нужный компонент в палитре и щелкнуть левой клавишей мыши в нужном месте проектируемой формы. Компонент появится на форме, и далее его можно перемещать, менять размеры и другие характеристики.

Каждый компонент C++ Builder имеет три разновидности характеристик: свойства, события и методы.

Если выбрать компонент из палитры и добавить его к форме, инспектор объектов автоматически покажет свойства и события, которые могут быть использованы с этим компонентом. В верхней части инспектора объектов имеется выпадающий список, позволяющий выбирать нужный объект из имеющихся на форме.

#### Свойства компонентов

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение. Многие свойства компонента в колонке свойств имеют значение, устанавливаемое по умолчанию (например, высота кнопок). Свойства компонента отображаются на странице свойств (Properties). Инспектор объектов отображает опубликованные (published) свойства компонентов. Помимо published-свойств, компоненты могут и чаще всего имеют общие (public), опубликованные свойства, которые доступны только во время выполнения приложения. Инспектор объектов используется для установки свойств во время проектирования. Список свойств располагается на странице свойств инспектора объектов. Можно определить свойства во время проектирования или написать код для видоизменения свойств компонента во время выполнения приложения.

При определении свойств компонента во время проектирования нужно выбрать компонент на форме, открыть страницу свойств в инспекторе объектов, выбрать определяемое свойство и изменить его с помощью редактора свойств (это может быть простое поле для ввода текста или числа, выпадающий список, раскрывающийся список, диалоговая панель и т.д.).

##### События

Страница событий (Events) инспектора объектов показывает список событий, распознаваемых компонентом (программирование для операционных систем с графическим пользовательским интерфейсом, в частности, для Windows 95 или Windows NT пре полагает описание реакции приложения на те или иные события, а сама операционная система занимается постоянным опросом компьютера с целью выявления наступления какого-либо события). Каждый компонент имеет свой собственный набор обработчиков событий. В C++ Builder следует писать функции, называемые обработчиками событий, и связывать события с этими функциями. Создавая обработчик того или иного события, вы поручаете программе выполнить написанную функцию, если это событие произойдет.

Для того чтобы добавить обработчик событий, нужно выбрать на форме с помощью мыши компонент, которому необходим обработчик событий, затем открыть страницу событий инспектора объектов и дважды щелкнуть левой клавишей мыши на колонке значений рядом с событием, чтобы заставить C++ Builder сгенерировать прототип обработчика событий и показать его в редакторе кода. При этом автоматически генерируется текст пустой функции, и редактор открывается в том месте, где следует вводить код. Курсор позиционируется внутри операторных скобок { ... }. Далее нужно ввести код, который должен выполняться при наступлении события. Обработчик событий может иметь параметры, которые указываются после имени функции в круглых скобках.

##### Методы

Метод является функцией, которая связана с компонентом, и которая объявляется как часть объекта. Создавая обработчики событий, можно вызывать методы, используя следующую нотацию: ->, например:

Edit1->Show();

Отметим, что при создании формы связанные с ней модуль и заголовочный файл с расширением \*.h генерируются обязательно, тогда как при создании нового модуля он не обязан быть связан с формой (например, если в нем содержатся процедуры расчетов). Имена формы и модуля можно изменить, причем желательно сделать это сразу после создания, пока на них не появилось много ссылок в других формах и модулях.

#### Менеджер проектов

Файлы, образующие приложение - формы и модули - собраны в проект. Менеджер проектов показывает списки файлов и модулей приложения и позволяет осуществлять навигацию между ними. Можно вызвать менеджер проектов, выбрав пункт меню View/Project Manager. По умолчанию вновь созданный проект получает имя Project1.cpp.

По умолчанию проект первоначально содержит файлы для одной формы и исходного кода одного модуля. Однако большинство проектов содержат несколько форм и модулей. Чтобы добавить модуль или форму к проекту, нужно щелкнуть правой кнопкой мыши и выбрать пункт New Form из контекстного меню. Можно также добавлять существующие формы и модули к проекту, используя кнопку Add контекстного меню менеджера проектов и выбирая модуль или форму, которую нужно добавить. Формы и модули можно удалить в любой момент в течение разработки проекта. Однако, из-за того, что форма связаны всегда с модулем, нельзя удалить одно без удаления другого, за исключением случая, когда модуль не имеет связи с формой. Удалить модуль из проекта можно, используя кнопку Remove менеджера проектов.

Если выбрать кнопку Options в менеджере проектов, откроется диалоговая панель опций проекта, в которой можно выбрать главную форму приложения, определить, какие формы будут создаваться динамически, каковы параметры компиляции модулей (в том числе созданных в Delphi, так как C++ Builder может включать их в проекты) и компоновки.

Важным элементом среды разработки C++ Builder является контекстное меню, появляющееся при нажатии на правую клавишу мыши и предлагающее быстрый доступ к наиболее часто используемым командам.

Разумеется, C++ Builder обладает встроенной системой контекстно-зависимой помощи, доступной для любого элемента интерфейса и являющейся обширным источником справочной информации о C++ Builder.

#### Создание приложений в C++ Builder

Первым шагом в разработке приложения C++ Builder является создание проекта. Файлы проекта содержат сгенерированный автоматически исходный текст, который становится частью приложения, когда оно скомпилировано и подготовлено к выполнению. Чтобы создать новый проект, нужно выбрать пункт меню File/New Application.

C++ Builder создает файл проекта с именем по умолчанию Project1.cpp, а также make-файл с именем по умолчанию Project1.mak. При внесении изменений в проект, таких, как добавление новой формы, C++ Builder обновляет файл проекта.

Проект или приложение обычно имеют несколько форм. Добавление формы к проекту создает следующие дополнительные файлы:

* Файл формы с расширением .DFM, содержащий информацию о ресурсах окон для конструирования формы
* Файл модуля с расширением .CPP, содержащий код на C++.
* Заголовочный файл с расширением .H, содержащий описание класса формы.

Когда вы добавляете новую форму, файл проекта автоматически обновляется.

Для того чтобы добавить одну или более форм к проекту , выберите пункт меню File/New Form. Появится пустая форма, которая будет добавлена к проекту. Можно воспользоваться пунктом меню File/New, выбрать страницу Forms и выбрать подходящий шаблон из репозитория объектов.

Для того, чтобы просто откомпилировать текущий проект, из меню Compile нужно выбрать пункт меню Compile. Для того чтобы откомпилировать проект и создать исполняемый файл для текущего проекта, из меню Run нужно выбрать пункт меню Run. Компоновка проекта является инкрементной (перекомпилируются только изменившиеся модули).

Если при выполнении приложения возникает ошибка времени выполнения, C++ Builder делает паузу в выполнении программы и показывает редактор кода с курсором, установленным на операторе, являющемся источником ошибки. Прежде чем делать необходимую коррекцию, следует перезапустить приложение, выбирая пункт меню Run из контекстного меню или из меню Run, закрыть приложение и лишь затем вносить изменения в проект. В этом случае уменьшится вероятность потери ресурсов Windows.

### Компоненты Borland C++ Builder

#### Выбор компонентов для групповых операций

Для эффективной разработки пользовательских интерфейсов приложений C++ Builder нередко возникает необходимость в манипулировании компонентами на формах. Большинство операций для манипулирования компонентами находятся в меню Edit: К различным опциям этого меню следует обращаться после того, как на форме вы ран один или несколько компонентов, свойства которых требуется изменить.

Выбрать один компонент можно следующими способами:

* Выбрав с помощью мыши компонент на форме
* Выбрав имя компонента в селекторе объектов.
* Переходом к компоненту на форме, нажимая клавишу Tab.

Выбрать несколько компонентов можно следующими способами:

* Удерживая нажатой клавишу Shift, щелкнуть мышью на каждом компоненте.
* Нажать левую клавишу мыши и окружить нужные компоненты прямоугольным контуром.

#### Установка разделяемых свойств компонентов

Большинство визуальных компонентов имеют общие свойства, (например, Visible, Width, Left). Для установки одинаковых значений общих свойств для нескольких компонентов необходимо выполнить следующие действия:

1. Выбрать несколько настраиваемых компонентов. При этом страница свойств объектов будет отображать только те свойства, которые имеются у всех выбранных компонентов.

2. Установить значения свойств, общих для выделенных компонентов.

#### Изменение размера компонентов

Изменение размера компонента можно проводить как при добавлении его на форму, так и после этого.

При добавлении компонента следует выбрать его на палитре компонентов. Далее нужно поместить курсор мыши на форму, нажать левую клавишу и перемещать мышь, в результате чего на форме появится прямоугольник, изображающий границы будущего компонента. Когда прямоугольник приобретет необходимые размеры, нужно отпустить кнопку мыши.

Если перевести курсор мыши на один из появившихся вокруг компонента маленьких черных квадратиков, курсор мыши изменяет форму. Перемещая этот курсор и вместе с ним границу компонента, можно изменять его размеры.

Для изменения размеров нескольких компонентов следует выбрать их одним из описанных выше способов. Далее нужно выбрать пункт меню Edit/Size. Появится диалоговое окно Size. Выберите опции размера. Для точной установки размера в пикселях можно ввести числа в поля Width и Height. Далее нужно нажать кнопку OK.

Можно добавить несколько копий компонента одного типа, выбирая компонент из палитры при нажатой клавише Shift. В этом случае вокруг компонента появляется прямоугольник, окружающий этот компонент. После этого каждый щелчок мышью на форме приводит к появлению на ней копии компонента. Закончив режим многократного копирования, следует щелкнуть мышью на инструменте выбора курсора (первая кнопка на палитре компонентов с изображением стрелки).

#### Выравнивание компонентов

Для выравнивания компонентов на форме можно использовать следующие комбинации клавиш:

|  |  |
| --- | --- |
| Shift + стрелки | Изменяет размер компонента на один пиксель в направлении выбранной стрелки |
| Shift + Ctrl + стрелки | Перемещает компонент на одну единицу сетки в направлении выбранной стрелки |
| Ctrl + стрелки | Перемещает компонент на один пиксель в направлении выбранной стрелки |

Можно также выровнять компоненты, используя пункт меню View/Alignment Palette. Для этого нужно:

1. Выбрать компоненты для выравнивания.
2. Выбрать пункт меню View/Alignment Palette.
3. Выбрать нужную кнопку.

Можно выровнять компоненты, используя пункт меню Edit/Align. Для этого нужно:

1. Выбрать компоненты для выравнивания.
2. Выбрать пункт меню Edit/Align. Появится диалоговое окно Alignment.
3. Выбрать нужную опцию и нажать на кнопку OK.

Можно изменить условия выравнивания компонент, используя пункт меню Options/Environment. Для этого нужно:

1. Выбрать пункт меню Options/Environment. Диалоговое окно Environment появится открытым на странице Preferences.

2. В группе Form designer можно выбрать следующие опции:

* Display grid - сделать сетку из точек на форме видимой для выравниваемых компонентов
* Snap to grid - заставить левые и верхние стороны компонентов расположиться на линиях сетки.

3. Для того, чтобы изменить расстояние между узлами сетки, нужно ввести новые значения вместо имеющихся. Значение по умолчанию -- 8 пикселей по оси X (по горизонтали) и по оси Y (по вертикали).

4. Нажать OK.

### Работа с базами данных в Borland C++ Builder.

Используя Borland C++ Builder, можно создать приложения, работающие как с однопользовательскими базами данных (БД), так и с серверными СУБД, такими как Oracle, Sybase, Informix, Interbase, MS SQL Server, DB2, а также с ODBC-источниками. Возможности C++ Builder, связанные с созданием приложений, использующих базы данных, весьма обширны для того, чтобы описать их в одной статье. Поэтому сегодня мы рассмотрим лишь простейшие возможности работы с таблицами баз данных.

Набор данных в C++ Builder - это объект, состоящий из набора записей, каждая из которых, в свою очередь, состоит из полей, и указателя текущей записи. Набор данных может иметь полное соответствие с реально существующей таблицей или быть результатом запроса, он может быть частью таблицы или объединять между собой несколько таблиц.

Набор данных в C++ Builder является потомком абстрактного класса TDataSet (абстрактный класс - это класс, от которого можно порождать другие классы, но нельзя создать экземпляр объекта данного класса). Например, классы TQuery, TTable и TStoredProc, содержащиеся на странице палитры компонентов Data Access, - наследники TDBDataSet, который, в свою очередь, является наследником TDataSet. TDataSet содержит абстракции, необходимые для непосредственного управления таблицами или запросами, обеспечивая средства для того, чтобы открыть таблицу или выполнить запрос и перемещаться по строкам.

#### Компонент TDataSource

Компонент DataSource действует как посредник между компонентами TDataSet (TTable, TQuery, TStoredProc) и компонентами Data Controls - элементами управления, обеспечивающими представление данных на форме. Компоненты TDataSet управляют связями с библиотекой Borland Database Engine (BDE), а компонент DataSource управляет связями с данными в компонентах Data Controls.

В типичных приложениях БД компонент DataSource, как правило, связан с одним компоненом TDataSet (TTable или TQuery) и с одним или более компонентами Data Controls (такими, как DBGrid, DBEdit и др.). Связь этого компонента с компонентами TDataSet и DataControls осуществляется с использованием следующих свойств и событий:

* Свойство DataSet компонента DataSource идентифицирует имя компонента TDataSet. Можно присвоить значение свойству DataSet на этапе выполнения или с помощью инспектора объектов на этапе проектирования.
* Свойство Enabled компонента DataSource активизирует или останавливает взаимосвязь между компонентами TDataSource и Data Controls. Если значение свойства Enabled равно true, то компоненты Data Controls, связанные с TDataSource, воспринимают изменения набора данных. Использование свойства Enabled позволяет временно разъединять визуальные компоненты Data Controls и TDataSource, например, для того, чтобы в случае поиска в таблице с большим количеством записей не отображать на экране пролистывание всей таблицы.
* Свойство AutoEdit компонента DataSource контролирует, как инициируется редактирование в компонентах Data Controls. Если значение свойства AutoEdit равно true, то режим редактирования начинается непосредственно при получении фокуса компонентом Data Controls, связанным с данным компонентом TDataSet. В противном случае режим редактирования начинается, когда вызывается метод Edit компонента TDataSet, например, после нажатия пользователем кнопки Edit на компоненте DBNavigator. · Событие OnDataChange компонента DataSource наступает, когда происходит изменение значения поля, записи, таблицы, запроса.
* Событие OnUpdateData компонента DataSource наступает, когда пользователь пытается изменить текущую запись в TDataSet. Обработчик этого события следует создавать, когда требуется соблюсти условия ссылочной целостности или ограничения, накладываемые на значения полей изменяемой базы данных.

#### Компонент TTable

Наиболее простым способом обращения к таблицам баз данных является использование компонента TTable, предоставляющего доступ к одной таблице. Для этой цели наиболее часто используются следующие свойства:

* Active - указывает, открыта (true) или нет (false) данная таблица.
* DatabaseName - имя каталога, содержащего искомую таблицу, либо псевдоним (alias) удаленной БД (псевдонимы устанавливаются с помощью утилиты конфигурации BDE, описание которой присутствует во многих источниках, посвященных продуктам Borland, либо с помощью SQL Explorer, вызываемого с помощью пункта меню Database/Explore). Это свойство может быть изменено только в случае, если таблица закрыта (ее свойство Active равно false), например:

Table1->Active = false;  
Table1->DatabaseName = "BCDEMOS"  
Table1->Active = true;

* TableName - имя таблицы.
* Exclusive - если это свойство принимает значение true, то никакой другой пользователь не может открыть таблицу, если она открыта данным приложением. Если это свойство равно false (значение по умолчанию), то другие пользователи могут открывать эту таблицу.
* IndexName - идентифицирует вторичный индекс для таблицы. Это свойство нельзя изменить, пока таблица открыта.
* MasterFields - определяет имя поля для создания связи с другой таблицей.
* MasterSource - имя компонента TDataSource, с помощью которого TTable будет получать данные из связанной таблицы.
* ReadOnly - если это свойство равно true, таблица открыта в режиме "только для чтения". Нельзя изменить свойство ReadOnly, пока таблица открыта.
* Eof, Bof - эти свойства принимают значение true, когда указатель текущей записи расположен на последней или соответственно первой записи таблицы.
* Fields - массив объектов TField. Используя это свойство, можно обращаться к полям по номеру, что удобно, когда заранее неизвестна структура таблицы:

Edit1->Text=Table1->Fields[2]->AsString;

Наиболее часто при работе с компонентом TTable используются следующие методы:

* Open и Close устанавливают значения свойства Active равными True и False соответственно.
* Refresh позволяет заново считать набор данных из БД.
* First, Last, Next, Prior перемещают указатель текущей записи на первую, последнюю, следующую и предыдущую записи соответственно, например:

Table1->First();  
while (!Table1->Eof)  
{  
//что-то делаем...  
Table1->Next();  
};

* MoveBy перемещает указатель на указанное число строк (оно может быть и отрицательным) в пределах таблицы
* Insert, Edit, Delete, Append - переводят таблицу в режимы вставки записи, редактирования, удаления, добавления записи соответственно.
* Post - осуществляет физическое сохранение измененных данных. Например:

Table2->Insert();   
Table2->Fields[0]->AsInteger = 100;   
Table2->Fields[1]->AsString =Edit1->Text;   
Table2->Post();

* Cancel - отменяет внесенные изменения, не сохраненные физически.
* FieldByName - предоставляет возможность обращения к данным в полях по имени поля:

S=Table1->FieldByName("area")->AsString;

* SetKey переключает таблицу в режим поиска.
* GotoKey начинает поиск строки, значение Fields[n] которой равно выбранному, где n - номер колонки таблицы, начиная с 0:

Table1->SetKey();  
Table1->Fields[0]->AsString=Edit1->Text;  
Table1->GotoKey();

* SetRangeStart, SetRangeEnd, ApplyRange позволяют выбрать нужные строки на основе диапазона значений какого-либо поля.

Table1->SetRangeStart();  
Table1->Fields[0]->AsString = Edit1->Text;  
Table1->SetRangeEnd();  
Table1->Fields[0]->AsString = Edit2->Text;  
Table1->ApplyRange();

* FreeBookmark, GetBookmark, GotoBookmark- позволяют создать помеченную строку в таблице и затем вернуться к ней позже. Методы Bookmark используют класс TBookmark. Метод GetBookmark устанавливает закладку на текущей cтроке таблицы. GotoBookmark осуществляет перемещение в таблице к строке, ранее отмеченной закладкой. Метод FreeBookmark используется для уничтожения объекта типа TBookmark:

TBookmark Marker =Table1->GetBookmark();  
Table1->GotoBookmark(Marker);  
Table1->FreeBookmark(Marker);

Cобытия компонента TTable позволяют строить и контролировать поведение приложе ий БД. Например, событие BeforePost наступает перед вставкой или изменением записи, событие AfterPost - после сохранения вставленной или измененной записи, событие AfterDelete - после удаления записи и т.д.

Чтобы внести компонент TTable в форму, нужно выполнить следующее:

1. Используя страницу Data Access палитры компонентов, разместить компонент TTable на форме или в модуле данных.

2. Свойству DatabaseName присвоить имя каталога, где находится БД, либо псевдо има БД.

3. Свойству TableName присвоить имя таблицы или выбрать таблицу из выпадающего списка.

4. Внести в форму компонент DataSource и установить значение свойства DataSet равным имени компонента TTable.

5. Внести компоненты Data Controls и связать их с компонентом DataSource для того, чтобы отобразить на экране данные из таблицы БД.

#### Компонент TField

Объекты класса TField являются свойством объекта TDataSet (напомним, что некоторые свойства объектов сами являются объектами с их собственными наборами свойств, и TField - один из них).

Свойство Fields объекта типа TDataSet позволяет обращаться к отдельным полям набора данных. Свойство Fields является массивом или набором объектов TField, динамически создающимся во время выполнения приложения. Элементы массива соответствуют колонкам таблицы.

Объект TField не делает никаких предположений относительно типов данных, с которыми он связан. Он имеет несколько свойств, позволяющих установить или вернуть обратно значения поля, например, AsString, AsBoolean, AsFloat, AsInteger. Наиболее часто используются свойства Text (cтрока текста, выводимого в связанный с данным полем интерфейсный элемент) и FieldName (имя поля базы данных).

Fields Editor позволяет создать так называемый статический список полей таблицы, добавляемых к описанию класса формы. Когда впервые используются такие компоненты TDataSet, как компонент TTable или TQuery, список полей для них динамически генерируется в процессе выполнения приложения на основе имеющихся столбцов таблиц или результатов SQL-запроса. Fields Editor позволяет определить и затем модифицировать статический список компонентов Field на этапе проектирования приложения. При внесении колонок с использованием Fields Editor для каждого из полей, добавленных к TDataSet, возникают объекты TField, после чего можно увидеть эти поля в инспекторе объектов и использовать в приложениях их свойства, события и методы.

Использовать Fields Editor нужно следующим образом:

1. Разместить компонент TTable или TQuery на форме.
2. Установить свойство DatabaseName для TTable или TQuery.
3. Установить свойство TableName компонента TTable или свойство SQL компонента TQuery.
4. Выбрать компонент TDataSet на форме и нажать правую клавишу мыши, после чего появится контекстное меню.
5. Из контекстного меню выбрать Fields Еditor. Появится пустое окно с заголовком, совпадающим с именем компонента TTable.
6. Снова нажать правую клавишу мыши над пустым окном и из контекстного меню выб ать опцию Add Fields.Имена всех колонок таблицы или запроса появятся в диалоговой панели Add Fields.
7. Выбрать поля, которые нужно внести в список объектов, и нажать OK.
8. Если требуется создать вычисляемое поле на основе имеющихся полей, нажать правую клавишу мыши и из контекстного меню выбрать New Field для создания нового поля на основе существующего или для создания вычисляемого поля (в дальнейшем следует создать код обработчика события OnCalcFields компонента TTable, где и производятся необходимые вычисления).
9. Если необходимо удалить статическое поле из списка полей в наборе данных, нужно нажать правую клавишу мыши и из контекстного меню выбрать Delete.

После того, как в Fields Editor добавлены поля, они появятся в инспекторе объектов, а ссылки на них - в h-файле формы.

Если теперь применить операцию drag-and-drop к выделенным в Fields Editor полям, перенеся их на форму, то можно получить готовую форму с необходимым набором интерфейсных элементов (в нашем случае - DBEdit, позволяющий отображать и редактировать строковые, числовые, денежные и другие поля, чьи значения представимы в виде строки символов, и DBImage, позволяющий отображать графические поля и использовать Clipboard для их редактирования). Если к такой форме добавить компонент TDBNavigator (этот компонент реализует основные методы TTable и TQuery, связанные с редактированием данных) и связать его с имеющимся компонентом TDataSource, а затем скомпилировать проект, получим приложение для просмотра и редактирования данных в таблице.

При работе Fields Editor создаются объекты, соответствующие видимым в инспекторе объектов полям. Эти объекты являются потомками объектного типа TField. Таблица 1 описывает существующие классы таких объектов:

Таблица 1. Потомки TField

|  |  |
| --- | --- |
| Потомок | Описание |
| TStringField | Текстовые данные фиксированной длины до 8192 символов. |
| TAutoIncField | Целые числа от -2,147,483,648 до 2,147,483,647. Предназначен для нумерации ст ок в наборе данных. Потомок TIntegerField. |
| TIntegerField | Целые числа от -2,147,483,648 до 2,147,483,647. |
| TSmallIntField | Целые числа от -32768 до 32767. |
| TWordField | Целые числа от 0 до 65535. |
| TFloatField | Действительные числа с абсолютной величиной от 1.2x10e-324 до 1.7x10e308 с точностью до 15-16 цифры. |
| TCurrencyField | Действительные числа с абсолютной величиной от 1.2x10e-324 до 1.7x10e308 с точностью до 15-16 цифры. |
| TBooleanField | Значения true или false. |
| TDateTimeField | Значения даты и времени. |
| TDateField | Значения даты. |
| TTimeField | Значения времени. |
| TBlobField | Произвольное поле данных без ограничений размера. |
| TBytesField | Произвольное поле данных без ограничений размера. |
| TVarBytesField | Произвольное поле данных до 65535 символов с фактической длиной, представленной в первых двух байтах. |
| TMemoField | Текст произвольной длины. |
| TGraphicField | Графическое поле произвольной длины, например, битовый массив. |

#### Компонент TDBGrid

Компонент TDBGrid обеспечивает табличный способ отображения на экране строк данных из компонентов TTable или TQuery. Приложение может использовать TDBGrid для отображения, вставки, уничтожения, редактирования данных БД. Обычно DBGrid используется в сочетании с DBNavigator, хотя можно использовать и другие интерфейсные элементы, включив в их обработчики событий методы First, Last, Next, Ptior, Insert, Delete, Edit, Append, Post, Cancel компонента TTable.

Внешний вид таблицы (например, надписи в заголовках столбцов) может быть изменен с помощью редактора свойств Columns Editor. Для вызова Columns Editor нужно либо выбрать соответствующую опцию в контекстном меню компонента DBGrid или щелкнуть мышью в колонке значений напротив свойства Columns в инспекторе объектов.

Вторым способом получения контроля над характеристиками DBGrid или другими компонентами является создание описанным выше способом статического набора компонентов TField. Имея компонент типа TField, созданный для каждого из полей в наборе данных, можно установить ширину, формат, маску, расположение, метку для отображения в DBGrid и другие характеристики.

Поля Float, Integer и Date обладают свойством DisplayMask. Это свойство можно использовать, чтобы форматировать данные в компоненте DBGrid или другом компоненте Data Controls. Например, экранный формат mm-dd-yy может использоваться для размещения полей типа дата.

Некоторые компоненты TField (например, TStringField) обладают свойством EditMask, которое можно установить, вводя данные в DBGrid и другие компоненты Data Controls. Для установки свойства EditMask нужно установить компонент Field в Object Inspector и выбрать свойство EditMask, после чего появится диалоговая панель Input Mask Editor,. Чтобы проверить маску редактирования, нужно ввести значение в поле Test Input.

### Поставка приложений, созданных с помощью С++ Builder

Одной из важнейших проблем при разработке заказных программных продуктов является окончательная сборка проекта и поставка приложений. Эта проблема может быть решена различными способами в зависимости от условий поставки, числа установок, состава поставляемого продукта.

#### Поставка исполняемых файлов

При окончательной сборке исполняемого файла (то есть непосредственно перед созданием дистрибутива) следует убрать из исполняемого файла отладочную информацию. Для этого следует из менеджера проектов вызвать диалоговую панель Project Options и нажать кнопку Release.

Если в проекте используются модули и формы Delphi, следует отключить все опции в секции Debugging и включить опцию Optimization в секции Code Generation на странице Pascal.

В той же диалоговой панели следует выбрать пиктограмму приложения (страница Application). Eе можно создать с помощью входящего в комплект поставки С++ Builder графического редактора, который можно запустить, выбрав пункт меню Tools/image Editor. Помимо этого, следует указать название приложения (то есть строку, которая будет отображаться в панели задач Windows) и имя файла справки, в котором содержатся экраны контекстно-зависимой помощи.

Если ссылка на справочный файл присутствует в опциях проекта, можно связать экраны помощи с интерфейсными элементами приложения. Для этой цели следует установить значения свойства HelpContext интерфейсных элементов приложения равным номерам разделов соответствующих экранов помощи из секции [MAP] файла проекта справки. В этом случае после компиляции приложения нажатие на клавишу F1 будет выводить на экран окно справочной системы с разделом справки, описывающим интерфейсный элемент, имеющий фокус ввода.

#### Установка и настройка Borland Database Engine

Если приложение не использует базы данных, в большинстве случаев будет достаточно просто скопировать на компьютер пользователя исполняемый файл и другие входящие в комплект поставки файлы (например, файл справки, файл оглавления справки и др.).

Если приложение использует базы данных, следует, помимо приложения, установить на компьютер пользователя библиотеку Borland Database Engine. Установка этой библиотеки заключается в копировании файлов этой библиотеки на компьютер пользователя, внесении сведений о ней в реестр (раздел HKEY\_LOCAL\_MACHINE//SOFTWARE//Borland//Database Engine), установке пиктограммы для утилиты конфигурации BDE, а также настройке псевдонимов, необходимых для корректной работы данного приложения.

В процессе установки BDE могут возникнуть следующие проблемы. Во-первых, у пользователя на компьютере могут быть установлены другие приложения, использующие эту библиотеку. Если скопировать на такой компьютер файл конфигурации BDE поверх имеющегося, работоспособность этих приложений может быть нарушена из-за того, что исчезнут описания псевдонимов, используемых ими. Поэтому при добавлении своих псевдонимов лучше пользоваться опцией File/Merge утилиты конфигурации BDE или создать псевдонимы вручную.

Можно посмотреть на эту проблему с другой стороны. Как избежать потери работоспособности вашего приложения, если другой программист установит свой файл конфигурации BDE поверх установленного вами? Возможное решение этой проблемы - создавать псевдонимы динамически во время работы приложения или анализировать их существование при запуске и создавать их, если по каким-то причинам они исчезли. Многие разработчики хранят информацию о них в локальных ini-файлах вне IDAPI32.CFG.

Вторая проблема может быть связана с несовпадением версий BDE. Если с момента выхода используемой вами версии BDE прошло какое-то время (а на данный момент после C++ Builder появились такие продукты, как Delphi и IntraBuilder 1.5, содержащие более новую версию BDE, чем С++ Builder), есть риск заменить старой версией BDE более новую, если на компьютере пользователя установлены эти средства разработки или созданные с их помощью приложения, что может нарушить их работоспособность.

Чтобы избежать подобных неприятностей, можно воспользоваться одной недокументированной особенностью BDE. Дело в том, что время создания файлов этой библиотеки совпадает с номером версии BDE, чем и можно воспользоваться. Путь к файлам BDE следует искать в реестре Windows (ключ HKEY\_LOCAL\_MACHINE\SOFTWARE\Borland\Database Engine, параметры CONFIGFILE01 и DLLPATH).

Обязательно следует убедиться, что эти каталоги и файлы действительно существуют. К сожалению, не все пользователи грамотно деинсталлируют приложения. Часто бывает, что ставший ненужным каталог просто стирается, при этом каталог с BDE может быть как удален, а может быть, и нет. При этом, скорее всего (исключения бывают очень редко) соответствующий ключ реестра сохранится, "сбивая с толку" инсталляционное приложение.

Отметим, что следует также установить на компьютер пользователя необходимые драйверы SQL Links для доступа к серверным СУБД, если они используются, а также клиентское программное обеспечение для работы с соответствующей серверной СУБД (например, SQL\*Net для Oracle). Кроме того, пользователь должен обладать необходимыми привилегиями для доступа к используемым вашим приложением объектам базы данных - таблицам, хранимым процедурам и др. Если вы готовите и серверную часть приложения, следует предоставить администратору базы данных так называемый DDL-сценарий серверной части (DDL - Data Definition Language), представляющий собой сценарий создания объектов базы данных, написанный на SQL либо на его процедурном расширении, характерном для данного сервера баз данных. Подобный сценарий может быть создан вручную либо с помощью какого-либо CASE-средства типа ERwin компании Logic Works (CASE, что расшифровывается как Computer-Aided Software/System Engineering, представляет собой технологию проектирования БД, основанную на графическом описании сущностей, атрибутов и связей между ними).

Если ваше приложение использует ODBC, следует установить соответствующий ODBC-драйвер и, возможно, 32-разрядный ODBC-администратор - он в общем случае не обязан присутствовать на компьютере пользователя.

#### Установка дополнительных компонентов приложения

Если ваше приложение использует ActiveX-компоненты, следует установить в каталог Windows\System соответствующий файл \*.OCX и внести в реестр соответствующие ветви (HKEY\_CLASS\_ROOT\CLSID\....) для регистрации соответствующих OLE-серверов.

Если ваше приложение использует какие-либо продукты третьих фирм (например, Run-time-версии генераторов отчетов сторонних производителей, например, Crystal Reports), следует установить их на компьютер пользователя в соответствии с инструкциями по поставке, предоставленными компанией-производителем.

Еще одна проблема, характерная для русскоязычных приложений, может быть связана с отсутствием на компьютере пользователя используемых вашим приложением шрифтов либо отображением их аналогов, содержащих вместо русских букв диакритические символы европейских алфавитов. Как решить эту проблему?

Самый простой способ - не использовать экзотических шрифтов. Шрифты типа Arial, Courier и Times New Roman, как правило, есть на всех компьютерах. Если же такой уверенности нет, можно либо предложить пользователю выбирать шрифты для приложения при его запуске и сохранять настройки в файлах (это неудобно, так как при этом надписи могут не уместиться на кнопках, метки могут "наехать" друг на друга и т.д.), либо инсталлировать нужные шрифты вместе с приложением. В случае Windows 95 достаточно скопировать их в каталог Fonts. Можно также присвоить "своим" шрифтам уникальные нестандартные имена, чтобы случайно не заменить ими шрифты, используемые другими приложениями (заменив в англоязычной версии Windows шрифт Arial его русскоязычной версией, вы в некоторых случаях можете лишить вашего пользователя возможности писать в текстовых процессорах по-французски или по-немецки этим шрифтом).

#### Способы создания дистрибутивов

Прежде чем описать процесс создания дистрибутива, рассмотрим, что именно делает инсталляционное приложение.

Во-первых, оно, как следует из вышеизложенного, копирует файлы (\*.exe, \*.ocx, \*.dll, \*.hlp, шрифты, данные и др.) в соответствующие каталоги.

Во-вторых, оно может модифицировать реестр.

В- третьих, оно может модифицировать переменную окружения PATH.

В-четвертых, оно создает программную группу и пиктограммы, а также модифицирует меню Windows-95 или Windows NT.

В-пятых, согласно общепринятым стандартам, оно должно предоставлять пользователю возможность выбора параметров установки (каталог, выбор устанавливаемых частей приложения), чтения лицензионного соглашения и файла readme, содержащего последнюю информацию о продукте.

В-шестых, оно может при необходимости перезапустить Windows.

Как создать инсталляционное приложение? Существует несколько вариантов решения этой проблемы. Во-первых, его можно написать на C++ Builder или Delphi, используя функции Windows API для модификации реестра согласно алгоритму, изложенному выше. Во-вторых, такое приложение можно создать с использованием генераторов дистрибутивов (InstallShield, Wise и др.). Отметим, что использование этих средств обычно подразумевает написание специализированного скрипта на языке, напоминающем С (в случае InstalShield) или Basic (в случае Wise).

Однако самый простой способ создать инсталляционное приложение - использовать входящий в комплект поставки версий Borland C++ Builder версий Professional и Client/Server Suite упрощенный генератор дистрибутивов Install Shield Express, позволяющий предоставить пользователю возможность выбора варианта инсталляции, добавить деинсталлятор (точнее, файл с расширением \*.isu, используемый утилитой установки и удаления программ Windows 95), установить BDE, добавить псевдонимы баз данных, редактировать диалоги инсталляционного приложения, создавать программные группы, создавать дистрибутивные дискеты.

Сразу же отметим недостатки дистрибутивов, созданных с помощью этого продукта. Во-первых, этот продукт предполагает, что устанавливаемая (то есть имеющаяся на компьютере, где создается дистрибутив) версия BDE - самая новая, что может привести к замене имеющейся на компьютере пользователя версии BDE на более раннюю. Во-вторых, некоторые диалоги инсталляционного приложения оказываются англоязычными.

Тем не менее, если перечисленные недостатки не очень важны или могут быть устранены в конкретном случае (например, путем выбора варианта инсталляции), использование данного средства может сэкономить довольно много времени при создании дистрибутива приложения.

#### Создание дистрибутивов с помощью InstallShield Express

При запуске Install Shield Express появляется окно с радиогруппой, предлагающее открыть существующий проект или создать новый.

При создании нового проекта следует ввести имя проекта и указать каталог, в котором расположены файлы приложения.

Флажок "Include custom setup type" следует выбрать, если вы планируете предоставить пользователю возможность выбора варианта инсталляции Custom Setup для самостоятельного выбора варианта инсталляции. После этого появится главный экран InstallShield, в котором перечислена последовательность шагов, которые следует выполнить для создания дистрибутива.

В разделе Set the Visual Design следует заполнить поля со сведениями о приложении: название приложения, как оно будет выглядеть в программной группе, имя исполняемого файла, каталог, в который следует установить приложение.

Отметим, что Install Shield содержит несколько переменных, идентифицирующих диски и каталоги компьютера пользователя, что позволяет не знать реальную систему каталогов и дисков этого компьютера:

* <INSTALLDIR> - каталог для установки, указанный пользователем,
* <WINDIR> - каталог, в котором содержится Windows
* <WINSYSDIR> - каталог Windows\System
* <WINDISK> - диск, на котором содержится Windows
* <WINSYSDISK> - диск, на котором содержится каталог Windows\System
* <ProgramFilesDir> - каталог Program Files.

Щелкнув на закладке Main Window, можно установить заголовок инсталляционного приложения, цвет фона и логотип.

Щелкнув на закладке Features, можно обнаружить одну-единственную опцию - Automatic Uninstaller. Ее рекомендуется оставить выбранной.

Следующий раздел - Specify InstallShield Options for Borland C++ - предназначен для выбора компонентов, часто поставляемых с приложениями: BDE, SQL Links и др. Выбрав нужный компонент (в нашем случае BDE) и нажав кнопку Settings, получим последовательность из четырех диалоговых окон для выбора частей BDE, поставляемых с данным приложением, и создания псевдонимов.

При установке параметров псевдонимов можно выбрать тип псевдонима и местоположение данных. Остальные параметры псевдонима (в том числе и языковые драйверы) можно указать в текстовом редакторе в нижней части окна.

Щелкнув по закладке Advanced Options, можно ознакомиться со списком файлов дополнительных компонентов (в нашем случае BDE) и сведениями о них.

Следующий раздел - Specify Components and Files - предназначен для определения групп файлов, компонентов приложения и типов установки. Щелкнув на закладке Groups, можно создать группы файлов для создания из них компонентов дистрибутива. Можно также использовать Explorer и переносить из него нужные файлы методом drag-and-drop (для этого нужно нажать кнопку Launch Explorer). Не рекомендуется оставлять группы пустыми.

Щелкнув на закладке Components, можно определить компоненты дистрибутива (их сможет выбирать пользователь в случае выбора варианта инсталляции Custom, поэтому можно дать им русскоязычные названия) и указать, из каких групп файлов они состоят. Не следует оставлять компоненты, не содержащие групп.

Следующая закладка - Setup Types - предназначена для определения вариантов установки. Если щелкнуть на ней, может оказаться, что вариант инсталляции его один - Typical (например, вы забыли отметить флажок "Include custom setup type"). В этом случае следует выбрать раздел Select User Interface Components и, щелкнув на закладке Components, в списке Dialog Boxes отметить опции Setup Type и Custom Setup. После этого можно вернуться к закладке Setup Types и определить варианты инсталляции. Как правило, варианты Custom и Typical содержат все возможные компоненты, а вариант Compact - минимальный набор компонентов, пригодный для нормальной работы приложения.

В разделе Select User Interface Components можно выбрать диалоги, в которых пользователь вводит необходимую информацию во время инсталляции (например, сведения о себе и компании, серийный номер продукта), знакомится с лицензионным соглашением и файлам readme, указывает каталог для инсталляции, выбирает тип установки и т.д. Возможен также предварительный просмотр диалогов при нажатии кнопки Preview.

Следующий раздел - Make Registry Changes - позволяет создавать на компьютере пользователя новые ключи реестра (закладка Keys) и значения ключей (Values). Это может оказаться полезным, если вы используете в вашем приложении компоненты ActiveX или создаете OLE-сервер. Ключи и их значения можно копировать из редактора реестра, если он поддерживает такую опреацию.

Следующий раздел - Specify Folders and Icons - позволяет определить состав будущей программной группы, а также определить параметры командной строки (закладка Advanced).

Наконец, последний раздел - Run Disk Builder. После сохранения инсталляционного скрипта (с помощью нажатия на кнопку с изображением дискеты на панели инструментов главного окна InstallShield) и выбора типа носителей происходит создание на жестком диске образов дистрибутивных дискет. Выбрав затем раздел Create Distribution Media, можно записать на дискеты созданный дистрибутив.

Опцию Test Run можно использовать для проверки работы инсталляционного приложения. Однако не рекомендуется делать это на компьютере, где производится разработка приложений. Лучше провести тестовые испытания на компьютере, похожем на компьютеры ваших пользователей. Кроме того, рекомендуется создать на этом компьютере копию Windows, чтобы в случае некорректной работы инсталлятора можно было вернуть программное обеспечение в исходное состояние.

Запуск инсталляционной программы приводит к последовательному появлению выбранных в разделе Select User Interface Components диалогов.

Результатом работы инсталляционного приложения является установка приложения и необходимых для его работы файлов на компьютер пользователя, создание программной группы, внесение необходимых ключей в реестр.

Если при создании дистрибутива вы выбрали опцию Automatic Uninstaller, то в случае возникновения необходимости деинсталляции установленного приложения следует использовать утилиту "Установка и удаление программ" в панели управления Windows.

Таким образом, хотя InstallShield Express и не решает полностью всех проблем, возникающих при поставке приложений, с его помощью во многих случаях возможно быстрое создание дистрибутивов, удовлетворяющих современным требованиям к функциональности и дизайну инсталляционных приложений.

В заключение отметим, что поставка приложений, созданных с помощью Delphi, осуществляется практически точно так же, как и поставка приложений, созданных с помощью C++ Builder.

### Список использованной литературы

* 1. Джарод Холингвэрт, Дэн Баттерфилд, Боб Сворт, Джэйми Оллсоп C++Builder 5. Руководство разработчика.
  2. Borland C++ Builder 5. Энциклопедия программиста. Калверт Ч., Рейсдорф К., "ДиаСофт" - 2001, 944 стр.
  3. http://www.codenet.ru/