Курсовая работа

**РАЗРАБОТКА ЭЛЕКТРОННОГО ОРГАНАЙЗЕРА СРЕДСТВАМИ C++ BUILDER 6**

Оглавление

Введение

Глава 1. Реализация программного кода "Organizer 1.0"

1.1 Borland C++ Builder 6. Общие сведения

1.2 Borland C++ Builder 6. Разработка приложений баз данных

1.3 Компоненты C++ Builder 6, предназначенные для работы по технологии InterBase

Глава 2. Технология разработки программного кода и интерфейса "Organizer 1.0"

2.1 Структура приложения "Organizer 1.0"

2.2 Программная структура приложения "Organizer 1.0"

2.3 Главная форма

2.3.1 Структура Главного меню

2.3.2 Календарь

2.3.3 Список текущих событий

2.4 Планировщик задач

2.4.1 Форма управления планировщиком задач

2.4.2 Форма добавления нового напоминания

2.4 3 Форма редактирования выбранного напоминания

2.5 Адресная книга

2.5 1 Форма управления адресной книгой

2.5.2 Форма поиска по параметрам

Заключение

Список использованных источников

## Введение

При современном темпе жизни и объеме информации, которую всегда нужно держать под рукой, электронный органайзер становится самой необходимой вещью любого современного человека. И неважно, кто он по профессии - главный директор крупной компании или забывчивый студент. Электронный органайзер поможет любому человеку помнить все даты, события и поручения.

Электронный органайзер "Organizer 1.0" - это не просто планировщик задач или ежедневник. Кроме функций классического органайзера он может использоваться как база клиентов, программа записной книжки для хранения контактов людей и предприятий, заметок, событий, праздников, юбилеев. Электронный ежедневник содержит удобный планировщик задач с напоминалкой. Каждая персона может сопровождаться такой информацией как Ф.И. О, адрес, телефон (дом, раб, сотовый), ICQ, e-mail, дата рождения, название фирмы, должность. В приложении есть функция быстрого поиска по адресной книге по нескольким параметрам, что существенно сокращает время для поиска нужной информации. В планировщике задач есть возможность просмотра событий за определенный промежуток времени, что поможет еще лучше спланировать свой рабочий график.

## Глава 1. Реализация программного кода "Organizer 1.0"

## 1.1 Borland C++ Builder 6. Общие сведения

Система программирования *Borland C++ Builder 6* завоевала достаточно прочные позиции среди профессиональных и начинающих программистов. Здесь можно отметить ряд причин: большую популярность языка программирования C++, удобство визуального конструирования приложений, развитые возможности доступных средств системы, эффективность генерируемого кода и др.

Несмотря на появление современных технологий типа.net и соответствующих систем программирования, таких как Visual C++.net, система C++ Builder будет устойчиво занимать свою нишу. Это обусловлено меньшей требовательностью к аппаратным ресурсам при разработке приложений, большей легкостью в освоении и применении средств системы для разработки приложений различной степени сложности.

Программа на C++ состоит из объявлений (переменных, констант, типов, классов, функций) и описаний функций. Среди функций всегда имеется главная - ***main*** для консольных приложений (работающих с WIN32) или WinMain для приложений Windows. Именно эта главная функция выполняется после начала работы программы. Обычно в C++Builder эта функция очень короткая и выполняет только некоторые подготовительные операции, необходимые для начала работы. А далее при объектно-ориентированном подходе работа приложения определяется происходящими событиями и реакцией на них объектов.

Как правило, программы строятся по модульному принципу и состоят из множества ***модулей***. Принцип модульности очень важен для создания надежных и относительно легко модифицируемых и сопровождаемых приложений. Четкое соблюдение принципов модульности в сочетании с принципом скрытия информации позволяет внутри любого модуля проводить какие-то модификации, не затрагивая при этом остальных модулей и головную программу.

В C++Builder все объекты компонентов размещаются в объектах - ***формах***. Для каждой формы, которую вы проектируете в своем приложении, C++Builder создает отдельный модуль. Именно в модулях и осуществляется программирование задачи. В обработчиках событий объектов - форм и компонентов, вы помещаете все свои алгоритмы. В основном они сводятся к обработке информации, содержащейся в свойствах одних объектов, и задании по результатам обработки свойств других объектов. При этом вы постоянно обращаетесь к методам различных объектов.

Согласно принципам скрытия информации обычно текст модуля разделяют на ***заголовочный файл*** интерфейса, который содержит объявления классов, функций, переменных и т.п., и ***файл реализации***, в котором содержится описание функций.

Стандартное расширение файлов реализации - ***срр***. Стандартное расширение заголовочных файлов - ***h***.

После того как программа написана, на ее основе должен быть создан ***выполняемый файл*** (модуль). Этот процесс осуществляется в несколько этапов.

Сначала работает препроцессор, который преобразует исходный текст. Препроцессор осуществляет преобразования в соответствии со специальными директивами препроцессора, которые размещаются в исходном тексте. Препроцессор может в соответствии с этими директивами включать тексты одних файлов в тексты других, развертывать макросы - сокращенные обозначения различных выражений и выполнять множество других преобразований.

После окончания работы препроцессора начинает работать компилятор. Его задача - перевести тексты модулей в машинный (объектный) код. В результате для каждого исходного файла. ***срр*** создается объектный файл, имеющий расширение. ***obj***.

После окончания работы компилятора работает компоновщик, который объединяет объектные файлы в единый загрузочный выполняемый модуль, имеющий расширение. ***ехе***. Этот модуль можно запускать на выполнение.

## 1.2 Borland C++ Builder 6. Разработка приложений баз данных

В зависимости от взаимного расположения приложения и БД можно выделить: локальные и удаленные БД. Для выполнения операций с локальными БД разрабатываются и используются так называемые локальные приложения, а для операций с удаленными БД - клиент-серверные приложения.

Расположение БД в значительной степени влияет на разработку приложения, обрабатывающего содержащиеся в этой базе данные.

Так, различают следующие виды приложений:

приложения, использующие локальные базы данных, называют одноуровневыми (однозвенными) приложениями, поскольку приложение и базы данных образуют единую файловую систему;

приложения, использующие удаленные базы данных, разделяют на двухуровневые (двухзвенные) и многоуровневые (многозвенные). Двухуровневые приложения содержат клиентскую и серверную части;

многоуровневые (обычно трехуровневые) приложения кроме клиентской и серверной частей имеют дополнительные части. К примеру, в трехуровневых приложениях имеются клиентская часть, сервер приложений и сервер базы данных.

Одно - и двухуровневые приложения C++ Builder могут осуществлять доступ к локальным и удаленным БД с использованием следующих механизмов:

BDE (Borland Database Engine - процессор баз данных фирмы Borland), предоставляющий развитый интерфейс API для взаимодействия с базами данных;

ADO (ActiveX Data Objects - объекты данных ActiveX) осуществляет доступ к информации с помощью OLE DB (Object Linking and Embedding Data Base - связывание и внедрение объектов баз данных);

dbExpress обеспечивает быстрый доступ к информации в базах данных с помощью набора драйверов;

InterBase Express реализует непосредственный доступ к базам данных сервера InterBase.

Механизм доступа InterBase Express, с помощью которого реализовано приложение "Organizer 1.0", ориентирован строго на работу с сервером InterBase и основан на прямом применении функций API этого сервера. Отсюда следуют все достоинства и недостатки использования этого механизма доступа. Он обеспечивает высокую скорость работы компонентов

доступа к данным. Очевидным недостатком механизма доступа InterBase является невозможность применения его для серверов баз данных, отличных от сервера InterBase SQL Server.

Технология InterBase Express строго ориентирована на работу с сервером InterBase версии не ниже 5.5. Отсюда следуют основные достоинства и недостатки этой технологии.

При реализации "Organizer 1.0" была использована утилита IB Expert, она обеспечивает более удобную работу с базами данных. Позволяет быстро и легко взаимодействовать с сервером InterBase: создавать, удалять, регистрировать и использовать базы данных.

## 1.3 Компоненты C++ Builder 6, предназначенные для работы по технологии InterBase

Компоненты C++ Builder 6, предназначенные для работы по технологии InterBase Express, расположены на странице InterBase Палитры компонентов.

Охарактеризуем кратко назначение основных из этих компонентов:

IBTable - для получения данных из таблицы или представления базы данных. Является аналогом компонента table для BDE, совместим с визуальными компонентами. Полученный с помощью этого компонента набор данных является редактируемым;

IBQuery - для получения данных с помощью SQL-запроса. Является аналогом компонента Query для BDE, совместим с визуальными компонентами;

IBstoredProc - для вызова хранимых процедур и получения набора данных на основе результатов выполнения процедуры. Соответствующий набор данных является не редактируемым. Совместим с визуальными компонентами;

IBDatabase - для установления соединения с базой данных;

IBTransaction - для управления транзакцией;

IBUpdateSQL - для создания модифицируемых наборов данных, основанных на SQL-запросах. Является аналогом компонента updateSQL для BDE. Используется совместно с компонентом iBQuery;

IBDataSet - для получения и редактирования данных. Совместим со всеми визуальными компонентами. Обеспечивает эффективный доступ к данным для просмотра и редактирования;

IBSQL - для быстрого выполнения SQL-запроса с минимальными накладными расходами. Не имеет локального буфера данных, не совместим с визуальными компонентами;

IBDatabaseInfo - для получения системной информации о свойствах базы данных, соединения и сервера;

IBSQLMonitor - для перехвата и отслеживания SQL-запросов, которые выполняют приложения пр технологии InterBase Express;

IBEvents - для обработки событий сервера InterBase;

IBExtract - для получения метаданных от сервера InterBase;

IBClientDataSet - для получения данных и применения обновлений. Использует внутренние компоненты TIBDataSet И TDataSetProvider.

Компоненты IBTable, IBQuery, IBStoredProc И IBUpdateSQL во многом похожи на свои аналоги механизма BDE.

## Глава 2. Технология разработки программного кода и интерфейса "Organizer 1.0"

## 2.1 Структура приложения "Organizer 1.0"

**"**Organizer 1.0" - программный продукт, который состоит из нескольких компонентов: планировщик задач, связанный с ним календарь и адресная книга.

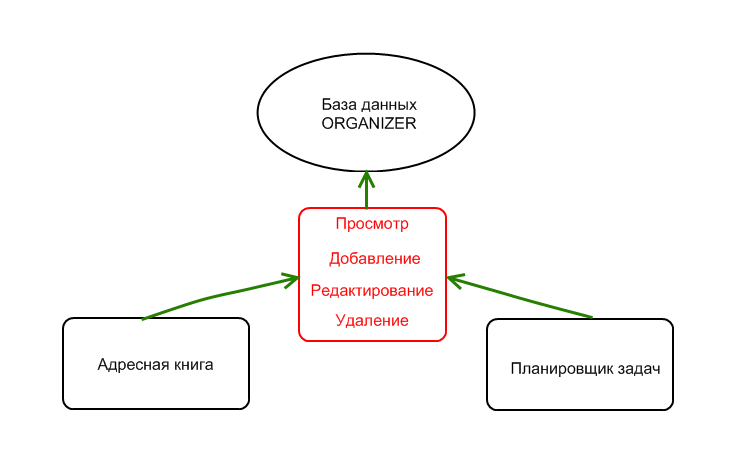


Рис.1 Структура приложения "Organizer 1.0".

В качестве основы приложения с помощью IB Expert была разработана база данных ORGANIZER, состоящая из двух таблиц: TBL\_EVENTS и TBL\_PEOPLE.

Табл.1 Структура таблицы TBL\_EVENTS

|  |  |  |
| --- | --- | --- |
| id\_events | date1 | name |
|  |  |  |

Табл. 2 Структура таблицы TBL\_ PEOPLE

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id\_people | imja | familia | otch | birth | email | icq | phone\_mob | phone\_home | phone\_work | firm | post |
|  |  |  |  |  |  |  |  |  |  |  |  |
| phone\_work | firm | post |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Таблицы между собой не связаны, т.к несут разную смысловую нагрузку и реализуют разные задачи в приложении.

Таблица TBL\_EVENTS предназначена для работы планировщика задач. Она содержит данные о введенных событиях (название, дата).

Таблица TBL\_ PEOPLE предназначена для работы адресной книги. Она содержит данные о персоне (Ф.И. О, адрес, телефон (дом, раб, сотовый), ICQ, e-mail, дата рождения, название фирмы, должность).

## 2.2 Программная структура приложения "Organizer 1.0"

В приложении используются:

главная форма с главным меню, обеспечивающим переход ко всем функциям приложения (frmMain);

формы, используемые Планировщиком задач для просмотра (frmViewEv), редактирования (frmEditEv) и добавления событий (frmAddEv);

формы, используемые Адресной книгой для просмотра (frmBook), добавления (frmAddB) и редактирования записей (frmEditB), а также форма поиска в книге по различным параметрам (frmSearchB);

модуль данных (DM), который содержит в себе компоненты DataSource, IBDataBase, IBTranzaction, IBQuery, обеспечивающие работу с базой данных.

Для реализации просмотра, добавления и редактирования информации в приложении использованы следующие компоненты С++ Builder:

*IBDatabase* Этот компонент в основном используется в приложениях, работающих на платформе InterBase. Его задачи связаны с общением с удаленным сервером, реализацией транзакций, работой с паролями.

*IBTransaction* - организует управление транзакциями.

*DataSource* В качестве источника данных всегда используется размещенный на странице Data Access компонент DataSource. Он связывается с набором данных своим свойством Data Set.

*IBQuery* - невизуальный компонент набора данных, выполняющий запросы SQL к серверу InterBase.

*DBGrid* представляет собой таблицу, в которой могут отображаться поля строк, числовые и булевы поля. Для связи этого компонента с набором данных используется только одно свойство - DataSource.

*DBNavigator -* набор кнопок для навигации по DBGrid.

## 2.3 Главная форма

На главной форме приложения представлено:

главное меню,

календарь,

навигационные кнопки, обеспечивающие быстрый доступ к адресной книге и планировщику задач соответственно,

список событий запланированных на сегодня.

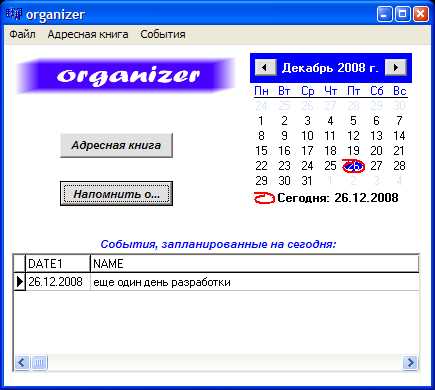


Рис. Главная форма

## 2.3.1 Структура Главного меню

Файл

Адресная книга (осуществляет переход на форму, отображающую записи Адресной книги - frmBook)

Напоминалка (осуществляет переход на форму, отображающую записи о событиях frmViewEv)

Выход (закрывает приложение)

Адресная книга

2.1 Просмотреть (осуществляет переход на форму, отображающую записи Адресной книги - frmBook)

2.2 Найти человека (осуществляет переход на форму поиска персоны по различным введенным данным - frmSearchB)

2.3 Добавить новую запись (осуществляет переход на форму, добавляющую записи в Адресную книгу - frmAddB)

3. События

3.1 Просмотр событий (осуществляет переход на форму, отображающую записи о событиях frmViewEv)

3.2 Создать напоминание (осуществляет переход на форму, добавляющую записи в Планировщик задач - frmAddEv)

## 2.3.2 Календарь

В качестве календаря для приложения был взят компонент C++ Builder TMonthCalendar, который находится во вкладке Win32.

Компонент TMonthCalendar (Календарь) используется для быстрого выбора даты с помощью мыши. Сразу после размещения в форме календарь готов к работе.

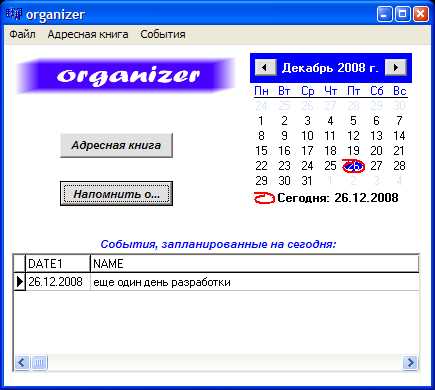


Рис.3. Календарь на форме

В красном кружке выделена текущая дата (она же подробнее указана в нижней части календаря), синим подсвечена дата, выбранная с помощью мыши или дата, устанавливаемая в календаре по умолчанию. Слева можно отображать номера недель в году (свойство WeekNumbers). С помощью кнопок в верхней части календаря можно перемещаться по месяцам. Чтобы вернуться к месяцу с текущей датой, достаточно щелкнуть на ней в нижней части календаря.

Кроме информативной, календарь выполняет также одну из основных функций приложения. При двойном щелчке мыши на календарь вызывается форма добавления нового напоминания, что упрощает работу пользователя с программой.

## 2.3.3 Список текущих событий

Реализован при помощи компонента DBGrid. Он связан с ресурсом данных DS1 и запросом на выборку qr4, находящимися в модуле данных DM.

При каждом запуске приложения выполняется функция FormCreate, которая выглядит следующим образом:

void \_\_fastcall TfrmMain:: FormCreate (TObject \*Sender)

{

MonthCalendar1->Date = Date ();

DM->qr4->Close ();

DM->qr4->SQL->Clear ();

DM->qr4->SQL->Text="SELECT DATE1, NAME FROM TBL\_EVENTS where DATE1=: CURRENT\_DATE";

DM->qr4->ParamByName ("CURRENT\_DATE") - >AsDate = Date ();

DM->qr4->Open ();

DM->qr4->FetchAll ();

}

Здесь запрос qr4 производит выборку данных из таблицы TBL\_EVENTS, значение столбца DATE1 которых совпадает с текущей датой.

## 2.4 Планировщик задач

## 2.4.1 Форма управления планировщиком задач

Планировщик задач вызывается с помощью главного меню: "События", "Просмотр событий". Появляется форма (frmViewEv), которая позволяет просматривать, добавлять, редактировать и удалять записи.

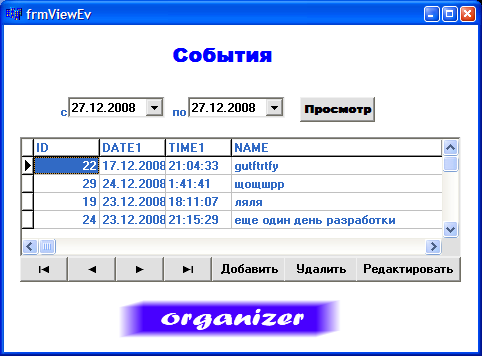


Рис.4 Форма управления планировщиком задач.

На форме представлено содержимое таблицы TBL\_EVENTS, с помощью DBGrid. Под таблицей расположены навигационные клавиши, позволяющие перемещаться по записям таблицы вперед и назад. Они реализованы при помощи компонента DBNavigator. Он связывается с базой данных ORGANIZER через свойство DataSet.

Функциональная клавиша "Удалить" удаляет всю текущую запись в DBGrid с помощью функции btnDelClick:

void \_\_fastcall TfrmViewEv:: btnDelClick (TObject \*Sender)

{

if (Application->MessageBox ("Вы действительно хотите удалить эту запись??","Подтвердите удаление",MB\_OKCANCEL + MB\_ICONQUESTION) ! = IDCANCEL)

{

try

{

DM->qr5->Close ();

DM->qr5->SQL->Clear ();

DM->qr5->SQL->Text = "DELETE from TBL\_EVENTS where id=: id";

DM->qr5->ParamByName ("id") - >AsInteger=DM->qr1->FieldByName ("id") - >AsInteger;

DM->qr5->Prepare ();

DM->qr5->ExecSQL ();

DM->IBTransaction1->Commit ();

DM->qr1->Close ();

DM->qr1->SQL->Clear ();

DM->qr1->SQL->Text = "SELECT \* FROM TBL\_EVENTS";

DM->qr1->Open ();

DM->qr1->FetchAll ();

}

catch (. .)

{

DM->IBTransaction1->Rollback ();

ShowMessage ("Error");

}

}

}

При нажатии кнопки "Удалить" функция вызывает диалоговое окно пользователя MessageBox, под названием "Подтвердите удаление", вопросом пользователю "Вы действительно хотите удалить эту запись?" и кнопками "Да" и "Отмена".

При нажатии кнопки "Да" функция выполняет SQL-запрос к таблице TBL\_EVENTS, удаляющий запись с текущим ID.

При нажатии "Отмена" функция возвращается на форму управления планировщиком задач.

На форме также представлена возможность просмотра записей таблицы TBL\_EVENTS за определенный промежуток времени. Для этого на форму помещены два компонента DataTimePicker, задающие этот промежуток по датам. Нажатие кнопки "Просмотр" реализует следующая функция:

void \_\_fastcall TfrmViewEv:: btnViewClick (TObject \*Sender)

{

DM->qr1->Close ();

DM->qr1->SQL->Clear ();

DM->qr1->SQL->Text="SELECT DATE1, NAME FROM TBL\_EVENTS where DATE1 between: d1 and: d2";

DM->qr1->ParamByName ("d1") - >AsDate=DateTimePicker1->Date;

DM->qr1->ParamByName ("d2") - >AsDate=DateTimePicker2->Date;

DM->qr1->Open ();

DM->qr1->FetchAll ();

}

После нажатия кнопки "Просмотр" выполняется SQL-запрос, который делает выборку тех записей в таблице TBL\_EVENTS, значения столбца которых находятся в промежутке между значениями DataTimePickerов, которые посылаются в запрос в качестве параметров.

Функциональные клавиши "Добавить" и "Редактировать" вызывают формы добавления и редактирования событий соответственно.

## 2.4.2 Форма добавления нового напоминания

Форма вызывается как из главного меню, так и с главной формы приложения нажатием кнопки "Напомнить о…", а также из формы управления планировщиком задач.

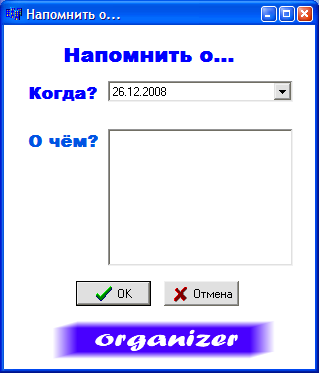


Рис.5. Форма добавления нового напоминания

На форме помещены 2 компонента: DateTimePicker1 и Memo1. после ввода соответствующей информации и нажатии на кнопку "ОК" вызывается функция BitBtn1Click:

void \_\_fastcall TfrmAddEv:: BitBtn1Click (TObject \*Sender)

{

try

{

DM->qr1->SQL->Clear ();

DM->qr1->SQL->Add ("INSERT into tbl\_events (DATE1,TIME1,NAME) values (: par1,: par2,: par3)");

DM->qr1->ParamByName ("par1") - >AsDate=DateTimePicker1->Date;

DM->qr1->ParamByName ("par2") - >AsTime=DateTimePicker1->Time;

DM->qr1->ParamByName ("par3") - >AsString= (Memo1->Text);

DM->qr1->ExecSQL ();

DM->IBTransaction1->Commit ();

ShowMessage ("successfull");

}

catch (. .)

{

DM->IBTransaction1->Rollback ();

ShowMessage ("Error");

}

Она выполняет подстановку значений DateTimePicker1и Memo1 при помощи параметров в запрос qr1 для добавления новой записи в таблицу TBL\_EVENTS.

## 2.4 3 Форма редактирования выбранного напоминания

Форма редактирования выбранного напоминания вызывается при помощи нажатия кнопки "Редактировать" на форме управления планировщиком задач.

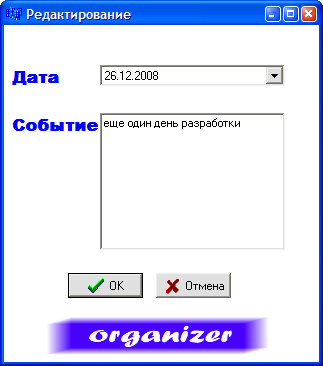


Рис.6. Форма редактирования выбранного напоминания

Чтобы форма выводила значения компонентов DateTimePicker1и Memo1 была написана функция FormShow:

void \_\_fastcall TfrmEditEv:: FormShow (TObject \*Sender)

{

DM->IBQuery1->Close ();

DM->IBQuery1->SQL->Clear ();

DM->IBQuery1->SQL->Text = "SELECT \* FROM TBL\_EVENTS where id=: par";

DM->IBQuery1->ParamByName ("par") - >AsInteger=DM->qr1->FieldByName ("ID") - >AsInteger;

DM->IBQuery1->Open ();

DM->IBQuery1->FetchAll ();

// ShowMessage (DM->IBQuery1->FieldByName ("DATE1") - >AsString);

DateTimePicker1->Date = DM->IBQuery1->FieldByName ("DATE1") - >AsDateTime;

Memo1->Text = DM->IBQuery1->FieldByName ("NAME") - >AsString;

}

А процесс сохранения измененной записи описывается функцией BitBtn1Click:

void \_\_fastcall TfrmEditEv:: BitBtn1Click (TObject \*Sender)

{

// DM->IBTransaction1->StartTransaction ();

try

{

DM->qr2->Params->Clear ();

DM->qr2->Close ();

DM->qr2->SQL->Clear ();

DM->qr2->SQL->Text="UPDATE TBL\_EVENTS a SET a. name=: par22,date1=: par1 where id=: par3";

DM->qr2->ParamByName ("par1") - >AsDateTime= DateTimePicker1->DateTime;

DM->qr2->ParamByName ("par22") - >AsString= Memowhat->Text;

DM->qr2->ParamByName ("par3") - >AsInteger=DM->qr1->FieldByName ("id") - >AsInteger;

DM->qr2->Prepare ();

DM->qr2->ExecSQL ();

DM->IBTransaction1->Commit ();

frmEditEv->ModalResult = mrOk;

}

catch (. .)

{

DM->IBTransaction1->Rollback ();

ShowMessage ("Ошибка при обновлении данных");

}

}

Она выполняет обновление таблицы TBL\_EVENTS новыми значениями DateTimePicker1и Memo1 запросом qr2.

## 2.5 Адресная книга

## 2.5 1 Форма управления адресной книгой

Адресная книга вызывается из помощи главного меню "Файл"-"Адресная книга" или "Адресная книга"-"Просмотреть" или нажатием кнопки "Адресная книга" на главной форме.

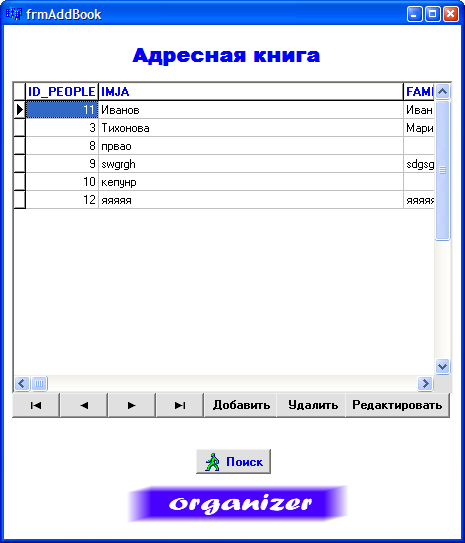


Рис.7. Форма управления адресной книгой

Адресная книга позволяет просматривать, добавлять, удалять и редактировать записи таблицы TBL\_PEOPLE.

Функции удаления, добавления и редактирования реализованы точно так же как и в форме управления планировщиком задач, за исключением того, что в запросах используется таблица TBL\_PEOPLE и ее соответствующие поля.

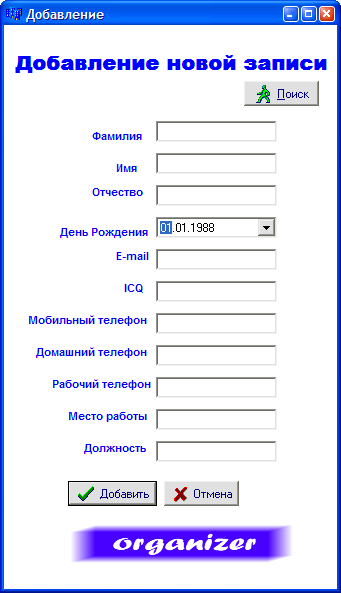


Рис.8. Форма добавления новой записи в Адресную книгу

На формах просмотра и добавления новой записи Адресной книги имеется кнопка "Поиск", нажатие на которую вызывает форму поиска по параметрам.

## 2.5.2 Форма поиска по параметрам

Форма поиска по параметрам вызывается из главного меню "Адресная книга" "Найти человека" или из форм просмотра и добавления записей Адресной книги нажатием кнопки "Поиск".

Поиск человека в адресной книге можно производить по трем параметрам:

по фамилии

по дате рождения

по месту работы

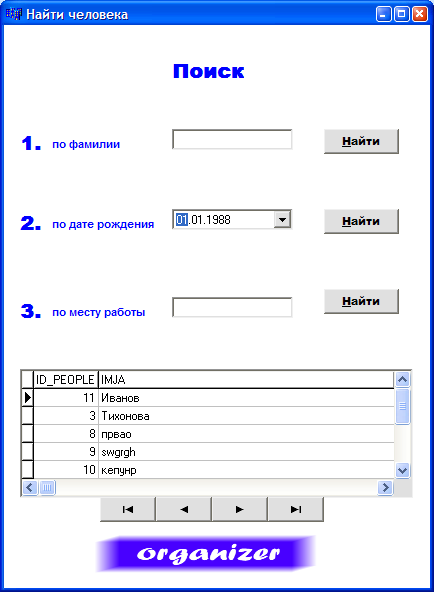


Рис.9. Форма поиска по параметрам

Для каждого поиска по параметрам предусмотрена отдельная кнопка, нажатие которой выполняет функцию, реализующую выборку записей таблицы TBL\_EVENTS отвечающие ключевым словам поиска.

Поиск по фамилии реализует следующая функция:

void \_\_fastcall TfrmSearchB:: btn1Click (TObject \*Sender)

{

DM->qr123->Close ();

DM->qr123->SQL->Clear ();

DM->qr123->SQL->Text="SELECT \* FROM TBL\_PEOPLE where FAMILIA=: par1";

DM->qr123->ParamByName ("par1") - >AsString=Edit1->Text;

DM->qr123->Open ();

DM->qr123->FetchAll ();

}

## Заключение

В результате проделанной курсовой работы был создан электронный органайзер "Organizer 1.0". Кроме функций классического органайзера он может использоваться как база клиентов, программа записной книжки для хранения контактов людей и предприятий, заметок, событий, праздников, юбилеев. Электронный ежедневник содержит удобный планировщик задач с напоминалкой. Каждая персона может сопровождаться такой информацией как Ф.И. О, адрес, телефон (дом, раб, сотовый), ICQ, e-mail, дата рождения, название фирмы, должность. В приложении есть функция быстрого поиска по адресной книге по нескольким параметрам, что существенно сокращает время для поиска нужной информации. В планировщике задач есть возможность просмотра событий за определенный промежуток времени, что поможет еще лучше спланировать свой рабочий график.

В будущем, возможно, будет разработана более усовершенствованная вервия. Например, можно расширить возможности адресной книги, если реализовать загрузку фотографий для каждого контакта. А также дополнить планировщик задач загрузкой и выгрузкой документов MSOffice.

## Список использованных источников

1. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 1. Язык C++. - М.: Бином-Пресс, 2002 г. - 544 с.: ил.
2. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты. М.: Бином-Пресс, 2002 г. - 528 с.: ил.
3. Хомоненко А.Д., Ададуров С.Е. Работа с базами данных в C++ Builder. - СПб.: БХВ-Петербург, 2006. - 496 с: ил.
4. http://it. kgsu.ru/