Курсовая работа

По дисциплине Информатика и программирование

**Автоматизация расчёта нагрузки и преподавателей ВУЗа**

**программирование база данных приложение**

**Введение**

Целью данной курсовой работы является разработка прикладной программы в среде Borland C++Builder 6.0 для реализации функции автоматизации расчёта и преподавателей ВУЗа. Для достижения цели необходимо создание гибкого интерфейса, а также осуществление следующих задач:

автоматизация ввода информации о курсах и группах, составление расписания проведения занятий для каждого преподавателя и каждой группы, реализация возможности сохранения, просмотра, изменения составленных документов, реализация возможности работы с буфером обмена, настойка внешнего вида окна, а также обеспечение защиты информации. Дополнительной задачей является представление разработанной программы как единого проекта, управляемого через пользовательский интерфейс.

В качестве среды разработки в данной курсовой работе использована визуальная среда Borland C++Builder. Данный выбор обусловлен следующими факторами:

Borland C++ Builder является средой быстрой разработки приложений. В На сегодняшний день С++ является одним из самых распространенных языков программирования. Его преимуществами являются гибкость, переносимость, универсальность. На этом языке написаны самые распространённые на сегодняшний день операционные системы, такие как Windows и Linux. Язык С++ может быть использован для решения практически любых задач.

Язык С++ не требует обязательного применения объектов в программах Это позволяет модернизировать ранее на писанные и создавать новые программы, пользуясь практически синтаксисом С и выполняя лишь более строгие требования С++ к типам: наличие в начале программы прототипов всех функций, определенных пользователей явного приведения типов для указателей на разные типы и некоторых других. Архитектура современных операционных систем становится все более и более объектно-ориентированной. При работе в таких системах не обойтись без понимания использования таких основополагающих понятий объектно-ориентированного программирования, как объекты, инкапсуляция и полиморфизм. основе систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой paзpa6oтки 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 существенно увеличивается. Система объектно-ориентированного программирования Borland C++ Builder, предназначена для операционных систем Windows. Интегрированная среда C++ Builder обеспечивает скорость визуальной разработки, продуктивность повторно используемых компонентов в сочетании с мощью языковых средств C++, усовершенствованными инструментами и разномасштабными средствами доступа к базам данных.

Профессиональные средства языка C++ интегрированы в визуальную среду разработки. C++Builder предоставляет быстродействующий компилятор с языка Borland C++, эффективный инкрементальный загрузчик и гибкие средства отладки как на уровне исходных инструкций, так и на уровне ассемблерных команд - в расчете удовлетворить высокие требования программистов-профессионалов.

С++ Builder может быть использован везде, где требуется дополнить существующие приложения расширенным стандартом языка C++, повысить быстродействие и придать пользовательскому интерфейсу качества профессионального уровня.

Профессиональные средства языка C++ интегрированы в визуальную среду разработки. C++Builder предоставляет быстродействующий компилятор с языка Borland C++, эффективный инкрементальный загрузчик и гибкие средства отладки как на уровне исходных инструкций, так и на уровне ассемблерных команд - в расчете удовлетворить высокие требования программистов-профессионалов.

Конструирование по способу "drag-and-drop " позволяет создавать приложение простым перетаскиванием захваченных мышью визуальных компонентов из Палитры на форму приложения. Инспектор объектов предоставляет возможность оперировать со свойствами и событиями компонентов, автоматически создавая заготовки функций обработки событий, которые наполняются кодом и редактируются в процессе разработки.

Свойства, методы и события - это именно те элементы языка, которые обеспечивают быструю разработку приложений в рамках объектно-ориентированного программирования. «Свойства» позволяют легко устанавливать разнообразные характеристики объектов. «Методы» производят определенные, иногда довольно сложные, операции над объектом. «События» связывают воздействия пользователя на объекты с кодами реакции на эти воздействия. События могут возникать при таких специфических изменениях состояния объектов как обновление данных в интерфейсных элементах доступа к базам данных. Работая совместно, свойства, методы и события образуют среду RAD (Rapid Application Development) быстрого и интуитивного программирования надежных приложений для Windows.

Визуальное наследование форм воплощает важнейший аспект объектно-ориентированного программирования в удобном для пользования инструменте визуального проектирования. Характеристики новой формы приложения могут быть унаследованы от любой другой существующей формы, что обеспечивает централизованную репродукцию изменений пользовательского интерфейса, облегчает контроль за кодом и уменьшает временные затраты на введение новых качественных атрибутов.

Мастер инсталляции руководит созданием унифицированных дистрибутивных пакетов для разработанных приложений.

Исходные тексты Библиотеки Визуальных Компонентов облегчают разработку новых компонентов на базе готовых примеров.

Отрытые инструменты API могут быть непосредственно интегрированы в визуальную среду системы. Вы сможете подключить привычный текстовый редактор или создать собственного мастера для автоматизации выполнения повторяющихся процедур.

Расширенная математическая библиотека содержит дополнительные унифицированные функции статистических и финансовых вычислений.

**Интерфейс Borland C++ Builder.**

В состав интерфейса С++ Builder входят:

**• Палитра компонент** содержит более 100 повторно используемых компонентов, предлагаемых для построения приложений.

**• Редактор форм** предназначен для создания интерфейса программы с пользователем.

**• Редактор кода** предназначен для написания текста программы, в частности, функций обработки событий.

**• Инспектор объектов** позволяет визуально устанавливать свойства объектов без необходимости рутинного программирования и содержит события, которые можно связывать с кодами реакции объектов на их возникновение.

**• Хранилище объектов** содержит такие объекты, как формы и модули данных, которые разделяются многими приложениями с целью уменьшения временных затрат при разработке.

C++Builder реализует визуальную методику построения приложений посредством выбора из Палитры компонентов нужных управляющих элементов . С каждым компонентом (например, кнопкой) связаны свойства, которые меняют ее вид и поведение. Любой компонент может вызывать серию событий, которые определяют его реакцию на различные воздействия[1].

**1.Основные подходы к разработке программных средств**

## В разработанном приложении интерфейс пользователя организован при помощи компонентов меню, компонентов работы с базами данных, а также при помощи механизма обмена, осуществляемого при помощи буфера обмена.

## Визуальная разработка приложений баз данных.

Разработка Систем Управления Базами Данных (СУБД) раньше всегда была очень трудоемким и медленным процессом, отягощенным необходимостью учитывать массу специфических деталей подсистем обслуживания различных баз данных на низком уровне. C++Builder принимает на себя выполнение этих рутинных операций позволяя сосредоточиться на решении основной задачи.

Все приложения СУБД, создаваемые в среде C++Builder, являются клиентами в архитектуре программного взаимодействия клиент/сервер. *Клиент* выдает запросы к серверу базы данных на получение или передачу информации. *Сервер* обрабатывает запросы от множества клиентов одновременно, координируя доступ к данным и их обновление.

Все приложения СУБД, создаваемые в среде C++Builder, основаны на компонентах пользовательского интерфейса с некоторой базой данных, которые предоставляют удивительно легкие в использовании средства разработки специальных приложений. Львиная доля времени процесса разработки уходит на визуальную установку свойств выбранных компонент. Удачно спроектированное приложение всегда обеспечивает простоту просмотра и редактирования данных пользователем, независимо от сложности структуры используемой модели данных. Данная глава с очевидностью покажет, что формы приложений СУБД для типично сложной системы в архитектуре взаимодействия клиент/сервер действительно могут быть созданы в интегрированной среде C++Builder весьма быстро и с малыми усилиями.

Воздействия на компоненты многогранны: их реакция на события обеспечивается стандартными методами, а установка значений свойств может производиться во время работы приложения. Таким образом, простое вначале приложение в процессе разработки постепенно усложняется, чтобы в конце концов выглядеть совершенно профессиональным программным изделием.

Организация доступа к базам данных

C++Builder организует доступ приложения к данным таким образом, чтобы полностью отстранить разработчика от специфики обслуживания конкретной базы данных.

Механизм BDE

Ключевой механизм BDE (Borland Database Engine), обеспечивающий работу визуальных компонент баз данных, действует как интерфейс между вашим приложением и самой базой данных. BDE реализован в виде набора системных DLL файлов. Взаимодействие компонентных объектов с BDE никак не специфицирует конкретную базу данных и не зависит от реализации обмена информацией на нижнем уровне иерархии. Именно BDE обращается в свою очередь к драйверам, специфическим для базы данных указанного типа, возвращая вашему приложению запрошенные фактические данные. BDE играет роль, аналогичную контроллеру драйверов ODBC (Open Database Connectivity) производства фирмы Microsoft, изолируя приложения от нижнего уровня взаимодействия с базой данных и увеличивая общую производительность связи за счет использования кэш-памяти. Используя BDE, вы получаете доступ ко всем локальным стандартным базам данных вашего компьютера, к источникам данных ODBC и к SQL серверам баз данных в архитектуре сетевой связи клиент/сервер.

Унифицированная технология BDE применяется во всех продуктах производства корпорации Borland: C++Builder, Borland C++, Delphi, IntraBuilder и JBuilder. Чтобы получить доступ к содержимому базы данных, приложению необходимо знать только идентификатор ее *псевдонима (alias).*

Рис. 1 изображает иерархическую структуру взаимодействия вашего приложения с базами данных.

Рис. 1. Структура взаимодействия приложений с базами данных

При добавлении компонент баз данных к форме вашего приложения соединение с BDE происходит автоматически - никакого программирования не требуется. Визуальный процесс соединения полностью находится под вашим контролем. Во время выполнения программы BDE делает необходимые запросы и получает данные, заказанные свойствами каждой используемой компоненты.[6]

Шаги создания таблицы БД при помощи DataBase Desktop представлены на рис.2,3,4,5.

Рис2 Внешний вид окна Database Desktop

Рис 3Подключение к БД

Рис4Окно выбора типа подключения

Рис 5Создание таблицы БД

**2.Характеристика визуальных компонентов среды разработки Builder 6.0, используемых в приложении**

**2.1 Компоненты главное и всплывающее меню(Main Menu, Popur Menu(Standart))**

Компоненты оформления приложений:

MainMenu создаёт на форме главное меню в виде строки с ниспадающими списками.

Содержимое пункта меню определяется свойством –Items[i]-задаёт значение пункта меню с номером i.

Значение пункта меню можно вводить программно: MainMenu->Items->Items[0]= “ ”.

В разработанном приложении данные компоненты обеспечивают удобный интерфейс между приложением и пользователем, позволяют пользователю управлять приложением (использованы на формах: «Управление приложением» и «Автоматизация расчёта нагрузки и преподавателей ВУЗа»).

**2.2 Компоненты, реализующие работу с БД**

### 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:

* Cвойство DataSet компонента DataSource идентифицирует имя компонента TDataSet. Можно присвоить значение свойству DataSet на этапе выполнения или с помощью инспектора объектов на этапе проектирования.
* Cвойство 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.

### 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).
* 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.

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.
* Insert, Edit, Delete, Append - переводят таблицу в режимы вставки записи, редактирования, удаления, добавления записи соответственно.
* Post - осуществляет физическое сохранение измененных данных.
* Cancel - отменяет внесенные изменения, не сохраненные физически.
* FieldByName - предоставляет возможность обращения к данным в полях по имени поля.

## В данном приложение компоненты, реализующие работу с БД использованы с целью обеспечения простоты просмотра и редактирования данных пользователем, независимо от сложности структуры используемой модели данных.[1]

**3.Программная реализация приложения**

**3.1 Дерево форм приложения**

Рис.1

**3.2.1Форма «Автоматизация расчёта нагрузки и преподавателей ВУЗа» является главной формой разработанного приложения, выполняет следующие функции:**

1.Возможность выбора режима работы с приложением;

1.1Переключение на режим просмотра (файлы базы данных недоступны для просмотра);

1.2Переключение на режим редактирования файлов баз данных (файлы можно просматривать и редактировать);

1.3Остановка работы приложения, выход из приложения;

1.4Поддерживается режим быстрого вызова через заранее фиксированные сочетания клавиш;

2.Защита информации (функция поддерживается только в режиме редактирования);

2.1Обеспечен ввод кода в появившееся окно ввода;

2.2Обеспечен ввод графического кода;

3Возможность управлять цветом формы при помощи пунктов меню;

4.Возможность использовать «горячих клавиш»

5.Возможность использовать всплывающие подсказки для комфорта работы пользователя.

Её внешний вид представлен на рис6

Рис6 Внешний вид формы «Автоматизация расчёта нагрузки и преподавателей ВУЗа»

Листинг модуля формы приведён в приложении А

**3.2.2Форма «Управление приложением» вызывается автоматически в процессе работы приложения при условии правильно введённого кода (в случае выбора режима редактирования) или при выборе режима просмотра.**

Форма служит для реализации следующих функций:

1.Возможность модификации базы данных;

2.Функция, использующая вычисляемое поле;

3.Функция фильтрации данных БД;

4.Функция поиска сведений БД;

5.Возможность переключения на другие формы приложения.

Её внешний вид представлен на рис7

Рис7 Внешний вид формы «Форма управления приложением»

Листинг модуля формы приведён в приложении В

**3.2.3Форма «Текстовый редактор» организует следующие функции:**

1.Создание нового документа;

2Открытие файла;

3Сохранение файла;

4Сохранение файла под указанным именем;

2.Использование буфера обмена :

2.1Функция «вырезать»;

2.2Функция «копировать»;

2.3Функция «вставить»;

3.Вызов формы «О программе» через пункт меню «Справка»;

Её внешний вид представлен на рисунке

Рис8 Внешний вид формы «Текстовый редактор»

Листинг модуля формы приведён в приложении А

**3.2.4Форма «Справка» содержит сведения о программе, её вызов инициализируется через пункт меню формы «Текстовый редактор»;**

Её внешний вид представлен на рис9

Рис9 Внешний вид формы «Справка»

Листинг модуля формы приведён в приложении А

**3.2.5Форма «Создание расписания» содержит три вкладки. Вкладка «Расписание для групп» организует следующие функции:**

1.Возможность использования механизма базы данных с целью создания расписания для групп;

2.Поддерживает функцию фильтрации данных таблицы;

Внешний вид первой вкладки представлен на рис10

Рис10 Внешний вид вкладки «Расписание для групп»

**3.2.5.1.Вкладка «Расписание для преподавателей»:**

1.Возможность выбора преподавателя из списка преподавателей с целью создания расписания.

2.Автоматическое вызов формы, содержащей расписание для преподавателя на основе данных таблицы с вкладки «Расписание для групп» Её внешний вид представлен на рис11

Рис11 Внешний вид вкладки «Расписание для преподавателей»

**3.2.5.2.Вкладка «Справка» служит для реализации следующих функций:**

1.Выбор раздела справки;

2.Загрузка файла данных, содержащего текст справки;

3.6Форма «Расписание» служит для отображения расписания для преподавателей. Форма создаётся автоматически при нажатии на кнопку в процессе работы приложения. Содержит таблицу, в которую автоматически записываются данные. Данные, содержащиеся в таблице уничтожаются при закрытии формы.

Внешний вид вкладки представлен на рис12

Рис12 Внешний вид вкладки «Справка»

Листинг модуля формы «Создание расписания» приведён в приложении А

**4.Тестирование приложения**

Протестируем разработанное приложение на возможность реализации перечисленных функций с целью проверки работоспособности приложения.

**4.1 Тест 1**

При тестировании приложения на возможность изменения внешнего вида окна получены следующие результаты (рис. 13)

Рис.13Выбор цвета формы приложения из перечня возможных цветов

Изменение цвета формы происходит в момент нажатия кнопки пункта меню «Закраска формы».(рис14,15)

Рис.14Результат изменения цвета1

Рис 15Результат изменения цвета2

**4.2Тест2**

Протестируем приложение на возможность выбора режима работы с базами данных.

Рис16 Выбор режима работы

После выбора режима необходимо нажать на кнопку «Работать!».

В режиме просмотра сведения, содержащиеся в базе данных остаются недоступными для пользователя.(рис16,17)

Рис 17Работа в режиме просмотра

При выбранном режиме редактирования нажатие кнопки «Работать!» даст следующий результат:

На форме появится ещё одно окно ввода, служащее для ввода пароля, а также ещё одна кнопка «Подтвердите правильность пароля».

Кроме того, высветится сообщение следующего вида (рис 18)

Рис18 Сообщение о необходимости ввода пароля

Рис19 Организация защиты информации в процессе работы в разработанном приложении.

При правильном вводе пароля нажатие на кнопку «подтвердите правильность пароля спровоцирует следующую реакцию приложения (рис19)

Рис 20Реакция на правильный ввод пароля

На форме высветится новое сообщение с другим запросом (рис

Рис.20Запрос на ввод графического кода

В том случае, если в окно ввода пароля на момент нажатия кнопки находится сообщение, не являющееся паролем, программа оповестит об этом пользователя при помощи сообщения следующего вида (рис21)

Рис21 Сообщение об ошибке при вводе пароля

В случае ошибки требуется снова ввести пароль, иначе приложение не допустит работы с ним.

Для ввода графического кода предусмотрены некоторые атрибуты, вызываемые через меню рабочей формы.

Код распознаётся по параметрам: цвет линии (рис22), тип линии (рис23), цвет заливки компонента Image, который в данном приложении предназначен для рисования.

Рис22 выбор цвета линии

Рис 23выбор типа линии

Рис 24 Ввод неверного графического кода

При несоответствии выбранных параметров условиям(рис , заданным программно появляется сообщение (рис25)

Рис25 Сообщение об ошибке

При необходимости вновь ввести графический код можно воспользоваться кнопкой «Очистить», отвечающей за очистку компонента Image (рис 26)

Рис 26Очистка компонента Image

В случае правильного задания графического кода, нажатие на кнопку «Добро пожаловать!» инициирует появление другой формы (рис27)

Рис 27Реакция приложения на ввод верного графического кода

4.3Тест3

Протестируем появившуюся форму на возможность модификации баз данных, а также на реализацию функции защиты информации, содержащейся в таблицах базы данных.

Для этого попытаемся ввести какие-либо сведения в одну из ячеек таблицы, затем переместимся на другую ячейку данной таблицы (рис28)

Рис28 Попытка модификации базы данных

Если проделать все вышеперечисленные действия, появится сообщение (рис29)

Рис29 Предупреждение нежелательной модификации БД

Далее при нажатии на кнопку OK модифицированная ячейка сохраняет своё состояние, в обратном случае данные таблицы приобретают первоначальный вид.

**4.4 Тест4**

Протестируем разработанное приложение на возможность создания вычисляемых полей в рамках используемых баз данных.

В данном случае вычисляемым поле будет поле «Возраст», которое рассчитывается на основе содержания поля «Год рождения» (рис30)

Рис 30 Использование вычисляемого поля

**4.5Тест5**

Протестируем приложение на возможность поиска по заданному ключу, а также фильтрации данных в соответствии с заданным условием.

Выберем пункт меню «Поиск», на форме появится панель поиска (рис31) В число параметров поиска включены поле, по которому необходимо осуществить поиск, и запись, которую требуется найти.

Рис 31Остановка указателя на строке с найденной записью

Для осуществления фильтрации выберем пункт меню «Сортировка».рис(32,33)

Рис32 Режим фильтрации включен

Рис 33Режим фильтрации отключён

Для удобства по отношению к GroupBox1, содержащей компоненты фильтрации, возможно применение функции «Скрыть», вызываемой через Popur Menu правым щелчком мыши по компоненту (рис34,35)

Рис 34 Вызов функции «Скрыть»

Рис 35Результат применения функции «Скрыть»

**4.6Тест6**

Протестируем функцию переключения между формами, реализуемую при помощи одноимённого пункта Главного меню формы управления приложением.

При выборе пункта «Создание файла» получим следующий результат (рис36)

В случае выбора пункта «Переключение режима» появится окно главной формы (рис37)

Рис36 Открытие формы «Текстовый редактор»

Рис 37 Переключение на главную форму

**4.7Тест7**

Протестируем функциональные возможности формы «Текстовый редактор».

Вторая группа кнопок на панели инструментов отвечает за сохранение и создание новых файлов (рис38,39).

Рис 38 Открытие файла

Рис39 Результат открытия файла

Возможно сохранение содержимого окна при нажатии на кнопку «Сохранить как» (рис40)

Рис 40 Сохранение файла под выбранным именем

Вторая группа кнопок отвечает за редактирование введённого текста.

Кнопки поддерживают функции соответственно «вырезать», «копировать», вставить».(рис41)

Рис41 Внесение фрагмента текста в буфер обмена

Последняя кнопка организует вызов формы, содержащей сведения о программе (рис 42)

Рис 42 Внешний вид окна справки

Самая первая кнопка служит для закрытия окна редактора.

Параллельно может осуществляться сохранение данных, содержащихся в окне (рис43)

Рис43 Событие на нажатие кнопки «Выход»

**4.8Тест8**

Протестируем форму «Создание расписания» на возможность реализации вложенных функций. Вкладка «Расписание для групп» формы реализует функции ввода(рис 44)и фильтрации (рис45) данных таблицы БД

Рис44 Реализация функции ввода

Рис45 Реализация функции фильтрации данных

Вкладка «Расписание для преподавателей» реализует функцию

Выбора преподавателя из общего списка (рис46,47,48), а также инициирует появление формы, содержащей расписание для данного преподавателя(рис.49,50)

Рис46 список преподавателей

Рис 47Осуществление выбора 1

Рис 48Осуществление выбора 2

Рис 49Расписание для преподавателя а

Рис 50Расписание для преподавателя б

При тестировании вкладки «Справка» получены следующие результаты (рис51,52,53)

Рис51 Реализация функции «справка»1

Рис 52Реализация функции «справка»2

Рис53 Реализация функции «справка»3

**Заключение**

В ходе разработки приложения была достигнута цель данной курсовой работы, а именно: реализации функции автоматизации расчёта и преподавателей ВУЗа. Поставленные в данной курсовой работе задачи были решены в ходе тестирования полученного приложения.

Готовая программа способна облегчить работу пользователя за счёт экономии времени и усилий, однако приложение не совершенно. Гибкость выбранной визуальной среды позволяет вносить в приложения необходимые коррективы. Удобный интерфейс позволяет создать необходимые условия для комфортной работы с приложением.

**Список использованных источников**

1. "Работа с локальными базами данных в C++ Builder 5", Аpхангельский .

2. "Введение в Borland C++ Builder", Елманова.

3.Джарод Холингвэрт, Дэн Баттерфилд, Боб Сворт, Джэйми Оллсоп C++Builder 5. Руководство разработчика.

4.Borland C++ Builder 5. Энциклопедия программиста. Калверт Ч., Рейсдорф К., "ДиаСофт" - 2001, 944 стр.

5.http://www.codenet.ru/

6."Язык C++ Builder 5. Справочное пособие", Архангельский А.

7."C++ Builder 5: 70 компонентов ввода/вывода инф.", Аpхангельский А.

8. http://www.interface.ru

9. Borland C++ Builder and MS SQL 2000 developer

**Приложение А**

Листинг формы «Автоматизация расчёта нагрузки и преподавателей ВУЗа.

#ifndef KurH

#define KurH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ExtCtrls.hpp>

#include <Menus.hpp>

#include "CGRID.h"

#include "PERFGRAP.h"

#include <Mask.hpp>

#include <jpeg.hpp>

#include <ComCtrls.hpp>

#include "Word\_2K\_SRVR.h"

#include <OleServer.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published:// IDE-managed Components

TMainMenu \*MainMenu1;

TButton \*Button1;

TMenuItem \*N2;

TMenuItem \*N4;

TMenuItem \*N5;

TCColorGrid \*CColorGrid1;

TPanel \*Panel1;

TComboBox \*ComboBox1;

TButton \*Button2;

TEdit \*Edit1;

TImage \*Image1;

TColorBox \*ColorBox1;

TLabel \*Label1;

TMenuItem \*N1;

TMenuItem \*N3;

TMenuItem \*N6;

TMenuItem \*N7;

TButton \*Button3;

TButton \*Button4;

void \_\_fastcall N5Click(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall N8Click(TObject \*Sender);

void \_\_fastcall N9Click(TObject \*Sender);

void \_\_fastcall N2Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Image1MouseDown(TObject \*Sender,

TMouseButton Button, TShiftState Shift, int X, int Y);

void \_\_fastcall CColorGrid1Change(TObject \*Sender);

void \_\_fastcall N4Click(TObject \*Sender);

void \_\_fastcall N10Click(TObject \*Sender);

void \_\_fastcall N1Click(TObject \*Sender);

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall N6Click(TObject \*Sender);

void \_\_fastcall N3Click(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

private:

int fbMouseDown;// User declarations

public:// User declarations

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

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

#include <vcl.h>

#pragma hdrstop

#include "Kur.h"

#include "Unit4.h"

bool OK;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CGRID"

#pragma link "PERFGRAP"

#pragma link "Word\_2K\_SRVR"

#pragma resource "\*.dfm"

TForm1 \*Form1;

TForm4 \*Form4;

//---------------------------------------------------------------------------

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

: TForm(Owner)

{

fbMouseDown=0;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N5Click(TObject \*Sender)

{

Form1->Close();

}

//---------------------------------------------------------------------------

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

{

int i,j;

for (i=0;i<Image1->Width;i++)

{for (j=0;j<Image1->Height;j++)

Image1->Canvas->Pixels[i][j]=clWhite;}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N8Click(TObject \*Sender)

{

Image1->Canvas->Pen->Style=psDot;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N9Click(TObject \*Sender)

{

Image1->Canvas->Pen->Style=psClear;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N2Click(TObject \*Sender)

{

CColorGrid1->Width=90;

CColorGrid1->Height=90;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{ if (ComboBox1->ItemIndex==0)

{

Form4=new TForm4(Application);

Form4->Table1->Active=false;

Form4->Table2->Active=false;

Form4->Show();

}

else

{Application->MessageBox("Защита конфиденциальности информации","Введите пароль",MB\_OK);

Edit1->Visible=true;

Button3->Visible=true;}}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Image1MouseDown(TObject \*Sender,

TMouseButton Button, TShiftState Shift, int X, int Y)

{

Image1->Canvas->LineTo(X,Y);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CColorGrid1Change(TObject \*Sender)

{

Image1->Canvas->Brush->Color=CColorGrid1->BackgroundColor;

Image1->Canvas->Rectangle(0,0,265,385);

Image1->Canvas->Pen->Width=1;

Image1->Canvas->Pen->Color=CColorGrid1->ForegroundColor;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N4Click(TObject \*Sender)

{

Form1->Color=ColorBox1->Selected;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N10Click(TObject \*Sender)

{

Image1->Canvas->Pen->Style=psSolid;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N1Click(TObject \*Sender)

{

Image1->Canvas->Pen->Style=psDot;

}

//---------------------------------------------------------------------------

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

{

Edit1->Visible=false;

Button3->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

if (Edit1->Text!="пэ-81")

Application->MessageBox("Пароль введён неверно","Сообщение об ошибке",MB\_OK);

else

{Application->MessageBox("Введите графический код","Добро пожаловать!",MB\_OK);}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N6Click(TObject \*Sender)

{

Image1->Canvas->Pen->Style=psDash;

}

//---------------------------------------------------------------------------

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

{

Image1->Canvas->Pen->Style=psDot;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

if (Image1->Canvas->Pen->Style==psDash)

{Form4=new TForm4(Application);

Form4->Show();}

else

Application->MessageBox("","Попробуйте ещё раз",MB\_OK);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Image1Click(TObject \*Sender)

{}

Листинг формы «Форма управления приложением»

#include <vcl.h>

#pragma hdrstop

#include "Unit4.h"

#include "Unit2.h"

#include "Kur.h"

#include "Unit3.h"

#include "uni.h"

#include "Unit7.h"

#include "Unit8.h"

#include "Unit5.h"

bool Success;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CGRID"

#pragma resource "\*.dfm"

TForm4 \*Form4;

TForm1 \*Form1;

TForm3 \*Form3;

//---------------------------------------------------------------------------

\_\_fastcall TForm4::TForm4(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Button1Click(TObject \*Sender)

{

GroupBox1->Visible=true;

Edit2->Visible=false;

GroupBox1->Caption="Èñõîäíûå äàííûå";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N11Click(TObject \*Sender)

{

Form4->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Table1BeforePost(TDataSet \*DataSet)

{

if (MessageBox(0,"Âû äåéñòâèòåëüíî õîòèòå ñîõðàíèòü èçìåíåíèÿ?","Ïðåäóïðåæäåíèå î íåæåëàòåëüíîé ìîäèôèêàöèè ÁÄ",MB\_YESNO)==IDNO)

Table1->Cancel();

else

{Table1->Post();

Table1->Append();}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Table1CalcFields(TDataSet \*DataSet)

{

Table1Voz->Value=2009-Table1God->Value;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N7Click(TObject \*Sender)

{ Form1=new TForm1(Application);

Form1->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::O1Click(TObject \*Sender)

{

Panel1->Visible=1;

ProgressBar1->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Edit1Change(TObject \*Sender)

{int i;

Timer1->Enabled=true;

ProgressBar1->Visible=true;

TLocateOptions Options;

Success=Table1->Locate(ComboBox1->Text,Edit1->Text,Options);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Timer1Timer(TObject \*Sender)

{

if (Edit1->Text!=""&&!Success)

Application->MessageBox("íå íàéäåíî íè îäíîé çàïèñè",MB\_OK);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N6Click(TObject \*Sender)

{

Form3=new TForm3(Application);

Form3->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Timer2Timer(TObject \*Sender)

{if (Edit1->Text!="")

ProgressBar1->StepIt();

if (ProgressBar1->Position==80)

{Timer1->Enabled=false;

ProgressBar1->Visible=false;}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N18Click(TObject \*Sender)

{

Form8=new TForm8(Application);

Form8->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::FormCreate(TObject \*Sender)

{

GroupBox1->Visible=false;

ProgressBar1->Visible=false;

Table1->FilterOptions>>foNoPartialCompare;

Table1->FilterOptions<<foCaseInsensitive;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N19Click(TObject \*Sender)

{

GroupBox1->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N20Click(TObject \*Sender)

{

Panel1->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N8Click(TObject \*Sender)

{

Form6=new TForm6(Application);

Form6->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N1Click(TObject \*Sender)

{

GroupBox1->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::CheckBox1Click(TObject \*Sender)

{

if (!CheckBox1->Checked)

Table1->Filtered=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Edit2Change(TObject \*Sender)

{float m;

String StrFiltr;

if (CheckBox1->Checked)

{Table1->Filtered=false;

Table1->Refresh();

StrFiltr=Edit2->Text+"\*";

if (ListBox1->Items->Strings[ListBox1->ItemIndex]=="FIO")

{Table1->Filter="FIO='"+StrFiltr+"'";}

else

{if (ListBox1->Items->Strings[ListBox1->ItemIndex]=="God")

Table1->Filter="God='"+FloatToStr(m)+"'";

else {if (ListBox1->Items->Strings[ListBox1->ItemIndex]=="Pol")

Table1->Filter="Pol='"+StrFiltr+"'";

else {if (ListBox1->Items->Strings[ListBox1->ItemIndex]=="Spec")

Table1->Filter="Spec='"+StrFiltr+"'";

else

Table1->Filter="Group='"+StrFiltr+"'";}}}}

{

Table1->Filtered=true;}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::N13Click(TObject \*Sender)

{

GroupBox1->Visible=false;

}

//---------------------------------------------------------------------------

Листинг формы «О программе»

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

//---------------------------------------------------------------------

#pragma resource "\*.dfm"

TForm7 \*Form7;

//---------------------------------------------------------------------

\_\_fastcall TForm7::TForm7(TComponent\* AOwner)

: TForm(AOwner)

{

}

//---------------------------------------------------------------------

void \_\_fastcall TForm7::OKButtonClick(TObject \*Sender)

{

Form7->Close();

}

//---------------------------------------------------------------------------

Листинг формы «Текстовый редактор»

#include <vcl.h>

#pragma hdrstop

#include "uni.h"

#include "Unit3.h"

//---------------------------------------------------------------------

#pragma resource "\*.dfm"

TForm3 \*Form3;

//---------------------------------------------------------------------

\_\_fastcall TForm3::TForm3(TComponent\* AOwner)

: TForm(AOwner)

{

}

//---------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton1Click(TObject \*Sender)

{

if (OpenDialog1->Execute())

{Memo1->Lines->LoadFromFile(OpenDialog1->FileName);

StatusBar1->Panels->Items[0]->Text=OpenDialog1->FileName;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton3Click(TObject \*Sender)

{

if (Memo1->Lines->Count>0)

{

if (MessageBox(0,"Ñîõðàíèòü ñîäåðæèìîå îêíà ðåäàêòèðîâàíèÿ? ","Ïîäòâåðäèòå ñîõðàíåíèå",MB\_YESNO)==IDYES)

{

SpeedButton4Click(Sender);

}

}

Memo1->Clear();

StatusBar1->Panels->Items[0]->Text="Áåç èìåíè";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton5Click(TObject \*Sender)

{

if (StatusBar1->Panels->Items[0]->Text=="Áåç èìåíè")

SpeedButton4Click(Sender);

else

{Memo1->Lines->SaveToFile(StatusBar1->Panels->Items[0]->Text); }

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton2Click(TObject \*Sender)

{

SpeedButton3Click(Sender);

Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton4Click(TObject \*Sender)

{

if (SaveDialog1->Execute())

Memo1->Lines->SaveToFile(SaveDialog1->FileName);

StatusBar1->Panels->Items[0]->Text=SaveDialog1->FileName;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton6Click(TObject \*Sender)

{

Memo1->CutToClipboard();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton7Click(TObject \*Sender)

{

Memo1->CopyToClipboard();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton8Click(TObject \*Sender)

{

Memo1->PasteFromClipboard();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton9Click(TObject \*Sender)

{

Form7=new TForm7(Application);

Form7->Show();

}

//---------------------------------------------------------------------------

Листинг формы «Создание расписания»

#include <vcl.h>

#pragma hdrstop

#include "Unit7.h"

//---------------------------------------------------------------------

#pragma resource "\*.dfm"

TForm6 \*Form6;

//---------------------------------------------------------------------

\_\_fastcall TForm6::TForm6(TComponent\* AOwner)

: TForm(AOwner)

{

}

//---------------------------------------------------------------------

void \_\_fastcall TForm6::CancelBtnClick(TObject \*Sender)

{

Table1->Cancel();

Form6->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::OKBtnClick(TObject \*Sender)

{

Table1->Post();

Form6->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::Button1Click(TObject \*Sender)

{

Table1->Bof;

while (Table1->Eof)

{Table1->Next();

}

TForm\* GridForm=new TForm((TComponent \*)NULL);

GridForm->Caption="Расписание";

GridForm->Width=300;

GridForm->Height=300;

TStringGrid \*Grid=new TStringGrid(GridForm);

Grid->ColCount=Table1->FieldCount;

Grid->RowCount=Table1->FieldCount;

Grid->Parent=GridForm;

Grid->Align=alClient;

GridForm->Show();

Grid->FixedCols=1;

Grid->FixedRows=1;

Grid->Cells[0][1]="8.20";

Grid->Cells[0][2]="10.00";

Grid->Cells[0][3]="11.45";

Grid->Cells[0][4]="13.20";

Grid->Cells[0][5]="15.00";

Grid->Cells[0][6]="16.45";

Grid->Cells[0][7]="18.00";

Grid->Cells[0][8]="19.45";

Grid->Cells[1][0]="Понедельник";

Grid->Cells[2][0]="Вторник";

Grid->Cells[3][0]="Среда";

Grid->Cells[4][0]="Четверг";

Grid->Cells[5][0]="Пятница";

Grid->Show();

GridForm->Handle;

String StrFiltr;

Table1->Filtered=false;

Table1->Refresh();

StrFiltr=ListBox1->Items->Strings[ListBox1->ItemIndex]+"\*";

Table1->Filter="FIO1='"+StrFiltr+"'";

Table1->Filtered=true;

int i;

while (!Table1->Eof)

{for (i=1;i<Table1->FieldCount;i++)

{if (Table1Time->Value==Grid->Cells[0][i])

Grid->Cells[1][i]=Table1Group->Value+Table1N1->Value;}

Table1->Next();

}

Table1->Filtered=false;

Table1->Bof;

Table1->Filter="FIO2='"+StrFiltr+"'";

Table1->Filtered=true;

while (!Table1->Eof)

{for (i=1;i<Table1->FieldCount;i++)

{if (Table1Time->Value==Grid->Cells[0][i])

Grid->Cells[2][i]=Table1Group->Value+Table1N2->Value;}

Table1->Next();}

Table1->Filtered=false;

Table1->Bof;

Table1->Filter="FIO3='"+StrFiltr+"'";

Table1->Filtered=true;

while (!Table1->Eof)

{for (i=1;i<Table1->FieldCount;i++)

{if (Table1Time->Value==Grid->Cells[0][i])

Grid->Cells[3][i]=Table1Group->Value+Table1N3->Value;}

Table1->Next();}

Table1->Filtered=false;

Table1->Bof;

Table1->Filter="FIO4='"+StrFiltr+"'";

Table1->Filtered=true;

while (!Table1->Eof)

{for (i=1;i<Table1->FieldCount;i++)

{if (Table1Time->Value==Grid->Cells[0][i])

Grid->Cells[4][i]=Table1Group->Value+Table1N4->Value;}

Table1->Next();}

Table1->Filtered=false;

Table1->Bof;

Table1->Filter="FIO5='"+StrFiltr+"'";

Table1->Filtered=true;

while (!Table1->Eof)

{for (i=1;i<Table1->FieldCount;i++)

{if (Table1Time->Value==Grid->Cells[5][i])

Grid->Cells[5][i]=Table1Group->Value+Table1N5->Value;}

Table1->Next();}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::Button2Click(TObject \*Sender)

{

RichEdit1->Lines->LoadFromFile(ListBox2->Items->Strings[ListBox2->ItemIndex]+".txt");

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::Edit1Change(TObject \*Sender)

{

String StrFiltr;

Table1->Filtered=false;

Table1->Refresh();

StrFiltr=Edit1->Text+"\*";

Table1->Filter="Group='"+StrFiltr+"'";

if (CheckBox1->Checked)

Table1->Filtered=true;

}

//---------------------------------------------------------------------------