Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

ПРОГРАММИРОВАНИЕ И АЛГОРИТМИЗАЦИЯ

Методические указания
к комплексу лабораторных работ
для студентов направлений 15.03.04 (220700.62)

Курган 2015

Кафедра: «Автоматизация производственных процессов»

Дисциплина: «Программирование и алгоритмизация»
(направления 15.03.04 и 220700.62).

Составил: ст. преподаватель И.В. Скобелев.

Утверждены на заседании кафедры «27» ноября 2014 г.

Рекомендованы методическим советом университета в рамках проекта «Инженерные кадры Зауралья» «20» декабря 2013 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматизация учетных и управленческих процессов на предприятии является одной из ключевых задач, стоящих перед руководством каждого предприятия. Дисциплина «Программирование и алгоритмизация» призвана ознакомить студентов с элементарными методами и средствами программирования. Курс основан на изучении языка программирования Visual C++ фирмы Microsoft, как наиболее широко распространенного средства создания прикладного программного обеспечения. Выполняя лабораторные работы студенты должны ознакомится с использованием библиотеки классов MFC (Microsoft Foundation Class). MFC - это инструмент, который, принимая на себя большую часть черновой работы, охватывающей большую часть функциональных возможностей операционных систем Microsoft Windows, а также предоставляющих разработчику значительное количество не только очень мощных дополнительных классов, но и целые механизмы, которые, не нарушая идеологию операционной системы, существенно ее расширяют и упрощают. Классы библиотеки полностью вобрали в себя многочисленные операторы switch, которые так загромождают программы, написанные на языке С. Наряду с этим вы можете совершенно свободно смешивать вызовы библиотеки классов с прямыми вызовами Win32 API.

Целью лабораторных работ является изучение принципов построения программ в среде Visual C++ с использованием MFC. В данной среде написаны многие современные приложения используемые для автоматизации трудоемких задач управления предприятием. Самым популярным из них является система 1С: Предприятие, имеющая достаточно гибкий пользовательский интерфейс и позволяющая разрабатывать новые решения. Программирование в среде 1С имеет некоторые особенности, связанные с предметной средой. Их изучение позволяет студентам получить навыки работы в области автоматизации производственных и управленческих процессов на предприятии. Платформа 1С Предприятия специально создавалась таким образом, чтобы максимально упростить создание, модификацию и поддержку прикладных решений с точки зрения непосредственно программирования, и позволить разработчику сфокусироваться на наиболее серьезных вопросах автоматизации предметной области.

Прежде всего, следует разделить создание (или существенное развитие) сложных прикладных решений, и небольшие доработки, которые необходимо выполнять в конкретной организации. На практике специалист, не имеющий большого опыта в разработке (продвинутый пользователь, системный администратор) достаточно быстро осваивает основные приемы разработки в 1С: Предприятии и может выполнять доработку прикладного решения, и даже создавать прикладные решения, предназначенные для автоматизации различных участков деятельности предприятия [1].

Вместе с тем разработка серьезных тиражных или заказных прикладных решений, а также существенная доработка бизнес-логики при внедрении типовых решений, безусловно, требует специалиста высокого

профессионального уровня. Правильнее использовать не термин программист, а термин разработчик. При этом, в отличие от специалистов, работающих с универсальными средствами программирования, разработчики решений на платформе 1С: Предприятия являются профессионалами прежде всего в сфере автоматизации деятельности предприятия. Само средство разработки изолирует, насколько это возможно, разработчика от технических деталей, оставляя ему решение наиболее важных вопросов построения структур данных, обработки информации, построения интерфейса. Профессиональный рост специалистов направлен в сторону понимания принципов управления предприятиями, создания адекватных экономических моделей, построения сложных структур данных, реализации эффективных алгоритмов бизнес-логики, создания эргономичного интерфейса [7].

Методические указания используются для изучения в курсе «Программирование и алгоритмизация», объем и количество лабораторных работ определяется по согласованию с преподавателем.

Лабораторная работа №1
Реализация простейшей программы на VC++ с использованием графического интерфейса и библиотеки MFC (Windows Application на основе диалогового окна)

Продолжительность 2 часа.

Цель работы: Научиться разрабатывать и реализовывать простейшие программы на языке VC++. Получить практические навыки работы по использованию графического интерфейса. Научиться связывать переменные и методы с элементами диалогового окна.

Задача: вычислять стоимость поездки на автомобили, если известно цена литра бензина, расход бензина на 100 км пути и пройденный километраж.

Выполнение работы

1. Создать пустой проект Console Application и реализовать в нем программу по вычислению стоимости поездки на автомобиле, рассмотренную на лекции. (Исходными данными являются: километраж, расход топлива, цена единицы топлива).
2. Создать приложение с помощью MFC AppWizard (exe), с диалоговым окном в качестве главного. Реализовать на основе данного приложения задачу из первого пункта, рассмотренную на лекции.
3. Усовершенствовать полученную во втором пункте программу, добавив подсчет стоимости нескольких поездок.
4. Усовершенствовать полученную программу, добавив кнопку при нажатии на которую вводятся заранее определенные значения (километраж =240 км, расход топлива = 8.4 л/км, цена литра топлива =16.5).
5. Создать приложение на основе диалогового окна, производящее вычисление по формуле, предложенной преподавателем.
6. Написать отчет о проделанной работе.

Пояснения к выполнению работы

Чтобы добавить новый элемент на диалоговое окно, его следует перетащить из палитры в конструируемое окно. Нам понадобятся: Static Text, Edit box, Button. Названия элемента можно увидеть, если подвести к элементу и ненадолго задержать указатель мыши.

Чтобы изменить надпись элемента, надо щелкнуть элемент правой кнопкой мыши и выбрать **свойства** в открывшемся окне, изменить поле Caption. Для того, чтобы кириллица правильно отображалась в диалоговом окне, надо в **свойствах** этого окна установить язык Russian.

Чтобы связать переменную с элементом диалогового окна необходимо вызвать ClassWizard из меню View. Перейти на вкладку Member Variable,

выделить требуемый идентификатор элемента и нажать кнопку Add Member Variable. В открывшемся окне набрать имя переменной и тип переменной.

Чтобы связать метод с элементом диалогового окна, необходимо вызвать ClassWizard из меню View. Выбрать из списка Object iDS имя соответствующего элемента, а из списка Message соответствующее действие (например, нажатие кнопки: BN_CLICKED). Теперь дважды щелкнуть по выбранному действию. На предложение создать обработчик ответ OK.

Формулы:

1. $S= 45*a+b/c+d;$
2. $S=d*(a+b+c)-d/f;$
3. $S=a-b*d+d/a;$
4. $S=r^3+a^2+b;$
5. $S=78*(a/b+c/d);$
6. $S=a^2 + b/d+c^2;$
7. $S=1/a+1/b+1/c^2;$
8. $S=67*c/d+f/a;$
9. $S=34/a+12+1/b^2;$
10. $S=(a+b+c)/f+a^2;$
11. $S=(a+b)^2/(c+d)^2;$
12. $S=1/(a+b) + 1/(a+b)^2;$
13. $S=3*(a+c+d)^2;$
14. $S=(a+1/b+1/c)^2;$

Контрольные вопросы

1. Алгоритмы. Основные понятия и определения.
2. Свойства и характеристики алгоритмов.
3. Способы описания алгоритмов.
4. В чем разница между консольным приложением и приложением на основе оконного интерфейса?
5. Какие преимущества и недостатки, по вашему мнению, имеет консольное приложение и приложение на основе оконного интерфейса?
6. Для чего служит ClassWizard.?
7. Для чего нужен AppWizard?
8. Как осуществляется конструирование графического интерфейса?
9. Как связать переменную с элементом графического интерфейса?
10. Как добавить метод, обрабатывающий реакцию на события, связанные с элементом графического интерфейса?
11. Оцените, на сколько сложно, по вашему мнению, написать программу под Windows.

Лабораторная работа №2
Создание графического интерфейса на базе диалогового окна VC++, с использованием переключателей, флажков. Реализация разветвляющихся алгоритмов

Продолжительность 4 часа.

Цель работы: Научиться разрабатывать и реализовывать программы на основе разветвляющихся алгоритмов, с использованием флажков и переключателей, а также операторов if и swish на языке VC++. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++ .

Выполнение работы

1. Создать приложение с помощью MFC AppWizard (exe), с диалоговым окном в качестве главного. Решить на основе данного приложения задачу: В гараже имеется 4 различных автомобиля (Ваз, Газель, Газ-66, Мерседес), каждый автомобиль имеет свой расход топлива на 100 км пути, а также свою стоимость топлива. Программа должна вычислять стоимость поездки для заданной машины. Использовать флажки и переключатели, операторы if или swish.
2. Усовершенствовать полученную в первом пункте программу, добавив режим заполнения данных для каждого автомобиля.
3. Усовершенствовать полученную программу, добавить грузоподъемность автомобилей и возможность указывать вес перевозимого груза. Программа должна рассчитать оптимальный вариант по стоимости перевозки груза.
4. Написать отчет о проделанной работе. В отчете алгоритм решения задачи изобразить в виде блок-схемы.

Пояснения к выполнению работы

Чтобы добавить новый элемент на диалоговое окно, его следует перетащить из палитры в конструируемое окно. Нам понадобятся: Static Text, Edit box, Button, Check box, Radio Button. Названия элемента можно увидеть, если подвести к элементу и ненадолго задержать указатель мыши.

Чтобы связать переменную с элементом диалогового окна необходимо вызвать ClassWizard из меню View. Перейти на вкладку Member Variable , выделить требуемый идентификатор элемента и нажать кнопку Add Member Variable. В открывшемся окне набрать имя переменной и тип переменной.

Чтобы добавить перемену к диалоговому окну, надо на вкладке ClassView щелкнуть правой клавишей мыши на класс диалогового окна и в открывшемся меню выбрать Add Member Variable, потом задать переменной имя и тип.

Контрольные вопросы

1. Оператор if, простой и составной. Привести примеры использования оператора if.

2. Оператор swish , простой и составной. Привести примеры использования оператора swish.
3. Чем Check box отличается от Radio Button?
4. Нарисуйте блок-схему линейного алгоритма.
5. Нарисуйте блок-схему разветвляющегося алгоритма.
6. Нарисуйте блок-схему разработанной программы. Какие виды алгоритмов в ней используются?

Лабораторная работа №3
Решение задач использующих изучаемые элементы графического интерфейса, массивы и структуры данных

Продолжительность 4 часа.

Цель работы: Научиться пользоваться списками и комбинированными полями, а также использовать структуры и массивы языка VC++. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++ .

Выполнение работы

1. Создать приложение с помощью MFC AppWizard (exe), с диалоговым окном в качестве главного. Решить на основе данного приложения задачу из лабораторной работы №3: В гараже имеется несколько различных автомобилей (например: Ваз, Газель, Газ-66, Мерседес), каждый автомобиль имеет свой расход топлива на 100 км пути, а также свою стоимость топлива. Программа должна вычислять стоимость поездки для заданной машины. Использовать списки и комбинированные поля. Для хранения данных использовать массивы или структуры данных.
2. Усовершенствовать полученную в первом пункте программу, добавив режим заполнения данных для каждого автомобиля. При создании программы иметь в виду, что количество автомобилей заранее не известно, заполнять список марок автомобилей в ходе выполнения программы. Предусмотреть возможность удаления марок авто из списка.
3. Усовершенствовать полученную программу, добавить грузоподъемность автомобилей и возможность указывать вес перевозимого груза. Программа должна рассчитать оптимальный вариант по стоимости перевозки груза.
4. Усовершенствовать полученную программу. Программа должна выполнять расчет и выводить результаты и исходные данные по щелчку на элемент списка или комбинированного поля.
5. Написать отчет о проделанной работе. В отчете алгоритм решения задачи изобразить в виде блок-схемы.

Пояснения к выполнению работы

Чтобы добавить новый элемент на диалоговое окно, его следует перетащить из палитры в конструируемое окно. Нам понадобятся: Static Text, Edit box, Button, Check box, Radio Button, List Box, Combo Box. Названия элемента можно увидеть, если подвести к элементу и ненадолго задержать указатель мыши.

Чтобы связать переменную с элементом диалогового окна необходимо вызвать ClassWizard из меню View. Перейти на вкладку Member Variable , выделить требуемый идентификатор элемента и нажать кнопку Add Member Variable. В открывшемся окне набрать имя переменной и тип переменной.

Чтобы добавить перемену к диалоговому окну, надо на вкладке ClassView щелкнуть правой клавишей мыши на класс диалогового окна и в открывшемся меню выбрать Add Member Variable, потом задать переменной имя и тип.

Заполнение полей начальными данными целесообразно делать в методе BOOL CListDlg::OnInitDialog()

При работе с List Box, Combo Box целесообразно связывать переменную с типом Control, тогда Вам станут доступны функции функции:

AddString(Строка) – добавляет строку в список.

InsertString(Номер строки,Строка), - вставляет строку в список под заданным номером. DeleteString(Номер) – удаляет строку с заданным номером.

GetCurSel() – возвращает номер выбранной строки.

GetText(Номер строки, Текстовая переменная) – помещает текст из выбранной строки в текстовую переменную. Функция для List Box.

GetLBText(Номер строки, Текстовая переменная)- помещает текст из выбранной строки в текстовую переменную. Функция для Combo Box.

Возможно, Вам пригодятся также математические функции для выполнения третьего пункта, из файла прототипов который включается в вашу программу конструкцией: #include "math.h"

ceil(a);– нахождение наименьшего целого большего числа a – тип double.

floor(a); - нахождение наибольшего целого меньшего числа a – тип double.

fmod(a,b);- нахождение остатка от деления a/b оба имеют тип double.

Контрольные вопросы

1. Для чего используются массивы? Приведите примеры использования массивов.
2. Для чего используются структуры? Приведите примеры использования структур.
3. Как получить доступ к элементам массива? Как присвоить элементам массива начальные значения?
4. Как получить доступ к элементам структуры?
5. Чем List Box отличается от Combo Box?
6. Где присваиваются начальные значения элементам List Box, Combo Box?
7. Какие операции можно выполнять со значениями полей List Box, Combo Box?

8. Как найти минимальное или максимальное значение в массиве? Приведите примеры.

Лабораторная работа №4
Работа с файлами в VC++. Использование файлов для хранения данных

Продолжительность 4 часа.

Цель работы: Изучить приемы работы с файлами, а также способы их создания. Получить практические навыки по использованию файлов для хранения информации.

Выполнение работы

1. Создать приложение с помощью MFC AppWizard (exe), с диалоговым окном в качестве главного. Использовать решенную в лабораторной работе №4 задачу, добавив в нее возможность записи внесенных в структуру данных в файле на диске и возможность в последующем чтения данных из файла в структуру.
2. Усовершенствовать полученную в первом пункте программу, добавив режим выбора файла в стандартном диалоге Windows .
3. Усовершенствовать полученную программу, добавив к кнопкам ОК и Cancel, программный код с напоминанием о необходимости сохранения данных на диске.
4. Усовершенствовать полученную программу, добавив к методу, определяющему положение движка, вычисление оптимального варианта использования автомобиля для веса перевозимого груза, заданного положением Slider.
5. Разработать тестовый пример (т.е. задать несколько вариантов марок автомобилей разной грузоподъемности, с разным расходом бензина и разной его стоимостью). Данные тестового примера вводить из файла, как делалось в работе №5. Исследовать с работу программы с разными исходными данными.

6. Написать отчет о проделанной работе. В отчете алгоритм решения задачи изобразить в виде блок-схемы.

Пояснения к выполнению работы:

Чтобы добавить новый элемент на диалоговое окно, его следует перетащить из палитры в конструируемое окно. Нам понадобятся: Static Text , Edit box, Button, Check box, Radio Button, List Box, Combo Box. Названия элемента можно увидеть, если подвести к элементу и ненадолго задержать указатель мыши.

Чтобы связать переменную с элементом диалогового окна, необходимо вызвать ClassWizard из меню View. Перейти на вкладку Member Variable, выделить требуемый идентификатор элемента и нажать кнопку Add Member Variable. В открывшемся окне набрать имя переменной и тип переменной.

Чтобы добавить перемену к диалоговому окну, надо на вкладке ClassView щелкнуть правой клавишей мыши на класс диалогового окна и в открывшемся меню выбрать Add Member Variable, потом задать переменной имя и тип.

Файл целесообразно объявить также как и переменную, в классе диалога.

Открывать и записывать данные в файл целесообразно по нажатию на кнопку, после записи его следует закрыть. То же самое для чтения следует добавить соответствующую кнопку.

Для работы с файлами вам понадобятся функции:
<Имя >=fopen("<имя на диске>","<вид доступа>"); - открытие файла.
fclose(<Имя >); -закрытие файла.
fprintf(<Имя >,"<список форматов>",<список вводимых значений>); - форматированная запись в файл.
fscanf (<Имя >,"<список форматов>",<список вводимых значений>); - форматированное чтение из файла.
feof(<Имя >); - определяет, достигнут ли конец файла.

CFileDialog fileDialog(TRUE,NULL,"*.txt",NULL,"(*.*)|*.*|(*.txt)|*.txt|"); - объект диалога для выбора файла.
int result = fileDialog.DoModal(); - так запускается окно диалога.
result==IDOK- если файл выбран.
fileDialog.GetPathName() – функция, получающая полное имя выбранного файла имеет смысл только если файл выбран (result==IDOK).
AfxMessageBox(<Текст или текстовая переменная >,< вид окна>); - выводит окно сообщений. Возвращает значение, равное 1, если нажата кнопка ОК.

Чтобы обрабатывать сообщения, получаемые от ползунка необходимо выполнить следующие действия: Вызвать ClassWizard из меню View. На вкладке Message Maps в окошке с названием Object iDs установить имя окна приложения (например, CLab6Dlg, у Вас может быть другое имя). Из списка Message выбрать сообщение соответствующее типу Вашего ползунка (WM_VScroll для вертикально расположенного ползунка и WM_HScroll для горизонтального), и дважды щелкнуть по нему. На предложение: создать обработчик, ответить ОК. Анализировать следует сообщение SB_THUMBPOSITION.

Для работы с элемент Slider вам понадобятся функции:
SetRangeMin(число ,false); -задает минимальное значение на шкале ползунка.
SetRangeMax(число ,false); -задает максимальное значение на шкале ползунка.
Созданный вами метод обработки сообщений от ползунка
void CLab6Dlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
возвращает

nSBCode == SB_THUMBPOSITION, если движок был перемешен.
nPos – значение, которое получилось в результате перемещения.

Контрольные вопросы

1. Для чего используются файлы? Приведите примеры использования файлов.
2. Как объявить файл? Приведите примеры.
3. Как открыть файл? Приведите примеры.
4. Как записать значения в файл? Приведите примеры.
5. Как организовать стандартный диалог Windows при выборе файла?
6. Как организовать форматированное чтение данных из файла?
7. Приведите примеры чтения массивов из файла.

Лабораторная работа №5
Создание меню в Windows Application (на базе Single document приложения). Подключение команд меню к коду программы

Продолжительность 2 часа.

Цель работы: Изучить порядок создания приложения типа Single document. Научиться создавать свои пункты меню, подключать к ним программный код. Получить практические навыки в написании программ с развитым меню.

Выполнение работы

1. Создать приложение с помощью MFC AppWizard (exe), выбрать тип предложения Single document.
2. Перейти на вкладку Res, найти ресурс IDR_MAINFRAME – меню и перейти в редактор меню. С помощью средств визуального редактора меню создать новый пункт меню и к нему несколько подпунктов.
3. Добавить к каждому пункту созданного Вами меню программный код, который позволяет выводить в окне документа надписи соответствующие выбранному пункту меню (например: **ВЫБРАН 1 пункт меню** или **ВЫБРАН 2 пункт меню** и т.д.).
4. Добавить возможность программной блокировки и разблокировки отдельных пунктов меню, а также возможности изменения надписей в пункте меню, установки галочек и точек напротив надписей (использовать полученную на лекции информацию).
5. Добавить кнопки на панель инструментов и привязать к ним команды меню, использовать ресурс Toolbar. Сделать клавиши - акселераторы, используя ресурс Accelerator.
6. Добавить пункт в меню, который позволяет нам ввести данные о автомобилях из файла, созданного в лабораторной работе №5. Выводим

полученные данные в окно программы (использовать материал лекции и предыдущих лабораторных работ).

7. Написать отчет о проделанной работе. В отчете алгоритм решения задачи изобразить в виде блок-схемы.

Пояснения к выполнению работы

Чтобы вывести в окно вида текстовую строку, можно поступить следующим образом:

1.В классе документа (например CLab7Doc) объявить переменную типа CString (например, CString S1;) . В конструкторе документа (CLab7Doc::CLab7Doc()) инициализировать строку.

```
CLab7Doc::CLab7Doc ()
{       S1="";}
```

2.В методе OnDraw(CDC* pDC) класса вида добавить строчку, выводящую значение на экран. pDC->TextOut(0,0,pDoc->S1); как в лабораторной работе №1.

3. С помощью ClassWizard создать обработчик для пункта меню. Обратите внимание, чтобы в окошке Class Name стоял класс вида (например, CLab7View).

4. В обработчике написать необходимую реакцию на выбор пункта меню.

Чтобы заблокировать или разблокировать отдельные пункты меню следует сделать обработчик для сообщения UPDATE_COMMAND_UI обычным образом с помощь ClassWizard.

pCmdUI->Enable(k); если к= 0, тогда заблокировано, к=1 разблокировано.

pCmdUI->SetText(S2); S2- текст, который выводится в меню.

pCmdUI->SetRadio(k); если к= 0,нет точки; к=1, есть точка.

pCmdUI->SetCheck(k); если к= 0,нет галочки; к=1, есть галочка.

Контрольные вопросы

1. Для чего нужно меню? Как создать проект с меню?
2. Как выводится текст в окно программы, какой метод для этого используется? Приведите примеры.
3. Как сделать метод для пункта меню?
4. Как сделать пункт меню не активным? Приведите примеры.
5. Как поставить галочку или точку у пункта меню?
6. Как программно изменить надпись в пункте меню?
7. Как сделать кнопку для пункта меню?
8. Как сделать акселераторы для пункта меню?

Лабораторная работа №6
Вызов диалоговых окон из меню в Windows Application (на базе Single document приложения). Решение задач с использованием различных элементов графического интерфейса

Продолжительность 4 часа.

Цель работы: Изучить включение в приложение Single document своих диалоговых окон. Научиться передавать значения между диалоговыми окнами и главным окном приложения. Изучить работу модального и немодального диалога. Получить практические навыки в разработке программ на основе MFC.

Выполнение работы

1. Создать приложение с помощью MFC AppWizard (exe), выбрать тип предложения Single document. Сделать свой пункт в меню с названием MyDial1, из которого будем вызывать диалоговое окно.
2. Создать диалоговое окно, выполнив команду Insert | Recourse и нажать кнопку NEW. В диалоговом окне сделать поле для ввода знаний и переменную для нее, как это делалось в работах 2-6.
3. Добавить необходимый код для запуска Вашего окна как **модального** из пункта меню и вывода текста, введенного в окне диалога в окно программы (т.е. изучить методы обмена информацией между модальным диалоговым окном и основным окном программы). Использовать информацию, полученную на лекции.
4. Добавить еще один пункт в меню с названием, например, MyDial2 . По аналогии с пунктом 2 добавить еще одно диалоговое окно с полем для ввода значений и кнопкой (это заготовка для **немодального** диалогового окна).
5. Добавить необходимый код для запуска немодального диалогового окна и обмена информацией между немодальным диалоговым окном и основным окном программы.
6. Написать отчет о проделанной работе. В отчете алгоритм решения задачи изобразить в виде блок-схемы.

Пояснения к выполнению работы

Использовать знания, полученные в работе №7 для создания пунктов меню.

Чтобы создать новый ресурс, надо в меню выполнить команду Insert | Recourse, выбрать диалоговое окно, и нажать кнопку NEW. Чтобы привязать новый ресурс к коду программы нужно создать для него класс, используя ClassWizard, ответив OK на предложение создать класс, и дать имя классу, например: Dil1.

В классе View – в методе для пункта меню назначаем классу соответствующую переменную, например: Dil1 d1;.

Для запуска окна используем метод d1.DoModal();.

Для получения значения переменной из модального окна надо ей присвоить значения в этом модальном окне. Например, с помощью кнопок OK и Cancel. Потом получить значение, например: pDoc->S1=d1.m_tx1;.

Для работы с немодальным окном, прежде всего надо при создании поставить галочку в свойствах Visible, чтобы окно стало видимым.Потом надо

создать указатели в диалоговом окне - указатель на класс вида, а в классе вид - указатель на класс диалогового окна.

Например, в классе диалогового окна:

Diln(CView* pView);

CView* m_pView;

Создать конструктор

Diln::Diln(CView* pView)

{ m_pView = pView;}

И метод для вызова окна

BOOL Diln::Creat()

{return CDialog::Create(Diln::IDD);}

В классе вида указатель на диалог**Diln* m_pDiln;**

В конструкторе создать объект диалога**m_pDiln = new Diln(this);**

В деструкторе его уничтожить**delete m_pDiln;**

В методе для меню создаем окно**m_pDiln->Creat();**

Для корректного удаления окна в кнопке диалогового окна создаем сообщение **WM_END** которое посылаем виду. **m_pView->PostMessage(WM_END);** его надо объявить в файле Diln.h следующим образом **#define WM_END WM_USER + 10**

И вставляем в таблицу реакций на сообщения вида свое сообщение **ON_MESSAGE(WM_EN, OnEnd1)**

А затем и саму функцию обработчик сообщений

LONG CLab8View::OnEnd1(UINT wParam, LONG lParam)

{m_pDiln->DestroyWindow();

return 0;}

Передача значения и вывод его в окно вида осуществляется с помощью вызова метода **m_pView->Invalidate();** из диалогового окна, а получение параметра в методе OnDraw(CDC* pDC) с помощью указателя на диалоговое окно.

pDoc->S1=m_pDiln->m_tx2;

Контрольные вопросы

1. Как создать диалоговое окно в проекте Single document?
2. Что значит модальный диалог? Приведите примеры.
3. Что значит не модальный диалог? Приведите примеры.
4. В чем отличие при построении модального и немодального диалога с точки зрения программиста?
5. Как передать параметр из модального окна и где его можно получить?
6. Как передать параметр из немодального окна и где его можно получить?
7. Для чего нужен указатель на класс вида в диалоговом окне?
8. Для чего нужен указатель на класс диалогового окна в классе вида?
9. Для чего передавать сообщение WM_END при закрытии окна?
10. Для чего мы делали функции OnEnd1(UINT wParam, LONG lParam)?

Лабораторная работа №7

Доступ к полям базы данных в Windows Application (на базе Single document приложения). Класс CFormView в качестве класса вида

Продолжительность 2 часа.

Цель работы: Получить начальные представления о базах данных и доступе к полям данным с помощью ODBC. Научиться обрабатывать полученную информацию. Получить практические навыки работы с базами данных.

Выполнение работы

1. Создайте приложение с помощью MFC AppWizard (exe), выбрать тип предложения Single document. Создать метод OnChar(UINT nChar, UINT nRepCnt, UINT nFlags) для чтения нажатых клавиш. Выводить полученные значения точно так же как делалось в работе №5 или №6 .
2. Усовершенствовать программу, добавив возможность обработки нажатия клавиши ENTER (обработка символа ‘\r’) для перевода на новую строку. Текст выводится в несколько строк.
3. Добавьте два пункта меню. По команде с одного из них текст выводится в центр окна, по команде с другого с левого края окна.
4. Добавьте пункт меню, который будет обрабатывать реакцию на нажатие клавиши мыши. Теперь текст будет выводиться в то место, куда щелкнете мышкой. Остальные пункты при этом должны продолжать работать.
5. Дополните программу возможностью убирать набранный текст (т.е. обработку нажатия клавиши Backspace –символ '\b')
6. Создайте с помощью приложение Microsoft Office Excel модель базы данных. Например, таблицу с полями данных о наводнениях в Кургане и Звериноголовском. Поля yr –год, ku – максим уровень у г. Кургана, kd – дата, zu – максимальный уровень у Звериноголовского, zd – дата. Заполнить таблицу данными и назвать ее tab1.
7. С помощью средств операционной системы сделаем этот файл источником данных ODBC следующим образом Пуск ->Панель управления->Администрирование -> источники данных (ODBC)-> Вкладка User DNS-> Файлы Excel->Configure-> Выберите созданную в первом пункте книгу.
8. Создайте приложение с помощью MFC AppWizard (exe), выбрать тип предложения Single document. На втором шаге поставьте точку напротив пункта Database view with file support и дальше, нажав на кнопку Data Source, выберите пункт ODBC в окошке Файлы Excel, потом - свою таблицу, созданную в первом пункте.
9. В созданном приложении сделайте поля редактирования для отображения значений из базы данных и свяжите их с полями базы данных с помощью Class Wizard.

10. Добавьте возможность выбора значения с самым высоким уровнем наводнения и его отображения в полях.
11. Добавьте возможность выбора и отображения в поле списка годов с уровнем наводнения превышающем заданное значение, вводимое в поле редактирования.
12. Напишите отчет о проделанной работе. В отчете алгоритм решения задачи изобразите в виде блок-схемы.

Пояснения к выполнению работы

Чтобы обрабатывать нажатие или отпускание клавиш, надо с помощью ClassWizard сделать метод обработчик сообщения WM_Char в классе вида, и добавлить в него свой код.

```
void CLab9View::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
      //*Здесь Ваш код
}
```

Конец строки, убирание последнего введенного символа осуществляются обычным образом: путем сравнения полученного символа с заданными (например '\b' '\r').

Размеры выводимой строки и размеры области в окне узнаем с помощью классов CRect и CSize.

Их можно использовать следующим образом.

```
GetWindowRect(&rect);
      x=rect.Width()/2;
      y=rect.Height()/2;
      size=pDC->GetTextExtent(pDoc->s1[ki]);
      x-=size.cx/2;
      y-=size.cy/2;}
```

Обработчик нажатия левой клавиши. Надо сделать метод обработки сообщения WM_OnLButtonDown

void CLab9View::OnLButtonDown(UINT nFlags, CPoint point)

point.x; point.y; - координаты точки щелчка мышью.

Для успешного выполнения работы необходимо использовать навыки, приобретенные при выполнении предыдущих работ, и знания, полученные на лекции.

Для выполнения работы Вам понадобится умение работать с программой Microsoft Office Excel на самом минимальном уровне. Некоторые знания по Microsoft Windows. Эти знания Вы должны были приобрести в курсе информатики.

Для выполнения работы также понадобятся функции работы с базами данных.

m_pSet->MoveFirst(); - перейти на первую запись вазы данных.

m_pSet->MoveNext(); - перейти на следующую запись.

m_pSet->Move(к); - перейти на к записей вперед.

m_pSet->MoveLast(); - перейти на последнею запись.

m_pSet->MovePrev(); - перейти на предыдущую запись.
m_pSet->SetAbsolutePosition(k); перейти на запись с номером к.
m_pSet->IsEOF()==1 – определить, достигнут ли конец таблицы

Как пользоваться массивами, как вводить данные в структуры, как выводить данные в списки, Вы должны были изучить в предыдущих лабораторных работах. Используйте полученные знания.

Контрольные вопросы

1. Что надо сделать, чтобы выводить в окно приложения символы введенные с клавиатуры?
2. От куда мы получаем символы генерируемые при нажатии клавиш?.
3. Как сделать ввод в несколько строчек?
4. Как вывести текст по центру окна?
5. Как определить размеры области окна и размеры текстовой строки?
6. Как задать реакцию на манипуляции с мышью?
7. Как убрать последний введенный с клавиатуры символ?
8. Что такое базы данных и для чего они используются?
9. В чем преимущества и недостатки организации хранения данных в виде базы данных?
10. Как получить доступ к полям базы данных?
11. Как отобразить данные из базы данных в программе?

Лабораторная работа №8
Создание сетевых приложений с использованием средств MFC. Использование класса CSocket

Продолжительность 6 часа.

Цель работы: Изучить использование стандартного класса CSocket для создания сетевых приложений, работающих с транспортными протоколами в современных компьютерных сетях. Научиться создавать клиентские и серверные части сетевых приложений.

Выполнение работы

1 часть: Создание клиента

1. Создайте приложение с помощью MFC AppWizard (exe) с диалоговым окном в качестве главного, на этапе создания приложения выберите приложение с поддержкой sockets. На диалоговом окне предусмотрите поля ввода типа Edit box для имени клиента, IP адреса, и текста, передаваемого серверу сообщения. Предусмотрите еще одно поле типа Edit box для отображения ответа от сервера и сделать его многострочным. Предусмотрите на форме две кнопки: одну для установки связи с сервером, другую для передачи ему сообщений.
2. Добавьте в проект класс MySocket как производный от стандартного класса CSocket с помощью ClassWizard. Добавьте во вновь образованный

класс виртуальные функции для передачи сообщения о разрыве связи OnClose, функцию для приема сообщений от сервера OnReceive и указатель для связи с формой диалогового окна.

3. В класс формы диалогового окна добавьте необходимый код для работы кнопок установки связи с сервером и кнопки передачи сообщений. Объявите все необходимые переменные для реализации работы программы клиента.
4. Оттранслируйте, исправьте ошибки и проверьте работоспособность программы, как на одном компьютере с сервером, так и в сети.

2 часть: Создание сервера

5. Создайте приложение с помощью MFC AppWizard (exe), с диалоговым окном в качестве главного, на этапе создания приложения выберите приложение с поддержкой sockets. На диалоговом окне предусмотрите два поля типа List Box: одно - для отображения соединений с клиентами, второе - для сообщений, получаемых от клиентов. Дальше предусмотрите поле типа Edit box для сообщения, передаваемого от сервера клиентам, и предусмотрите две кнопки: одна отвечает за передачу выбранному клиенту, вторая за передачу всем клиентам.
6. Добавьте в проект класс CClientSocket как производный от стандартного класса CSocket с помощью ClassWizard. Добавьте во вновь образованный класс виртуальные функции для передачи сообщения о разрыве связи OnClose , функцию для приема сообщений от сервера OnReceive и указатель для связи с формой диалогового окна.
7. Добавьте в проект класс CListeninngSocket как производный от стандартного класса CSocket с помощью ClassWizard. Добавьте во вновь образованный класс виртуальную функцию OnAccept(int nErrorCode) для реакции на подключение очередного клиента и указатель для связи с формой диалогового окна.
8. Добавьте в проект класс CPool, в котором и будем производить всю работу, связанную с созданием сокетов и передачей информации между сервером и клиентами. Добавьте в него необходимые для обработки и передачи информации функции. BOOL Send(int index,CString data); void ProcessRead(CClientSocket *pSocket,CString msg); int ProcessClose(CClientSocket *pSocket); int ProcessAccept(); BOOL Send2All(CString data); BOOL Init(void *phost,int port);.
9. Добавьте в класс диалогового окна функцию для отображения сообщений от клиентов UpdateWiew(), а также методы для передачи сообщений.
10. Протестируйте и отладьте сервер вместе с клиентами, работающими на одном компьютере и в сети. При работе в сети необходимо указывать реальные IP адреса.
11. Напишите отчет о проделанной работе. В отчете алгоритм решения задачи изобразите в виде блок-схемы.

Пояснения к выполнению работы

Для реализации той части программы, которая, собственно, и отвечает за все взаимодействие с сетью, нам понадобится три класса: - CClientSocket, CListeningSocket и CPool. Первый создается для каждого нового клиента, подсоединяющегося к серверу. CListeningSocket, как видно из названия, - "слушающий" сокет. Его основная и единственная задача - отвечать на запросы о подключении новых клиентов. CPool хранит в себе все создаваемые сокеты, управляет ими и осуществляет связь с остальной частью программы. Теперь рассмотрим каждый класс подробно.

```
class CClientSocket : public CSocket
{
public:
  CClientSocket(CPool* ppool);
  virtual void OnReceive(int nErrorCode);
  virtual void OnClose(int nErrorCode);
protected:
  CPool* pPool;
};
```

Этот класс - наследник CSocket, который, в свою очередь, является наследником CAsyncSocket. Последние два класса принадлежат библиотеке MFC. В них выполнена вся черная работа по упаковке WinSock API в объектно-ориентированную обертку. Заголовки классов находятся в файле afxsock.h, а особо любопытные могут заглянуть в их исходный текст sockcore.cpp, находящийся в том же каталоге, что и остальные исходные материалы библиотеки. Строго говоря, основная нагрузка лежит на CAsyncSocket, и можно было бы свой класс наследовать от него, а не от CSocket, но, несмотря на то, что классы выглядят очень похожими, в их устройстве, а значит и использовании есть существенные различия.

Некоторые функции, принадлежащие CAsyncSocket, такие, как Receive, Send, Accept, всегда возвращают ошибку с кодом WSAEWOULDBLOCK. Это связано с тем, что они не дожидаются своего полного выполнения, и поэтому результат их вызова приходится определять в других местах программы. В CSocket те же самые функции ожидают, пока все необходимые действия не завершатся, и лишь тогда возвращают управление программе. Сокет с такими свойствами называется блокирующим. Для взаимодействия в локальной сети такие задержки в большинстве случаев не будут заметны. Но при использовании Интернета время ожидания может оказаться слишком большим. Тогда существует два варианта: один - воспользоваться CAsyncSocket и самостоятельно создать механизм обработки подтверждений. Его можно реализовать следующим образом: после вызова упомянутых функций устанавливаем таймер и, если по прошествии некоторого времени не получаем уведомление об удачном выполнении затребованных действий, констатируем неудовлетворительный результат. Другой - выделить для CSocket отдельный поток.

Теперь вернемся к нашему классу. Единственная необходимая его переменная - указатель на объект типа CPool, т.е. указатель на хозяина. Его инициализация совершенно необходима, поэтому конструктор по умолчанию заменяем на следующий:

```
CClientSocket::CClientSocket(CPool* ppool)
{
  pPool=ppool;
}
```

Кроме того, нам потребуется переопределить две виртуальные функции. Кстати, ClassWizard поддерживает классы сокетов, так что требуется просто выбрать название нужной функции из списка. Вообще, если им пользоваться, то про сокеты он еще много чего напишет, предложит переопределить копирующий конструктор и т.д., но на это творчество можно спокойно закрыть глаза.

```
void CClientSocket::OnReceive(int
  nErrorCode)
{
char buff[4096];
  CString buffer;
int nRead;
nRead=Receive(buff, 4096);
switch (nRead)
{
case 0:
    Close();
    break;
case SOCKET_ERROR:
    AfxMessageBox ("Error occurred");
    Close();
  break;
default:
  buff[nRead]='\0';
  buffer=buff;
pPool->ProcessRead(this,buffer);
  }
  CSocket::OnReceive(nErrorCode);
}
```

Эта функция вызывается средой, чтобы сообщить сокету о получении новых данных. Их мы можем получить, используя функцию Receive, которой в качестве параметров передается указатель на массив памяти с записанными принятыми байтами и размером массива. В случае успешного выполнения функция возвращает все полученные байты. При возникновении ошибки возвращается SOCKET_ERROR. Кроме того, получение нулевого значения

трактуется как сигнал об обрыве соединения, такое может произойти, если передающий сокет был закрыт во время транзакции. Для того чтобы известить о произошедшем "вышестоящие органы", вызываем ProcessRead(this), передавая себя самого через аргумент.
И последняя функция:

```
void CClientSocket::OnClose(int
  nErrorCode)
{
  pPool->ProcessClose(this);
CSocket::OnClose(nErrorCode);
}
```

Она вызывается при закрытии текущего соединения. Переопределять ее нужно только для того, чтобы сообщить о закрытии CPool. Обратите внимание на вызов функций, принадлежащих предкам. Microsoft настаивает, что это необходимо. На самом деле функции пустые, но не исключено, что в последующих версиях там может появиться какой-либо код.

Думаю, никто не будет спорить, что приведенный класс никак нельзя назвать «тяжелым» как с точки зрения понимания, так и количества строк реализации. Но CListeningSocket еще «легче».

```
class CListeningSocket : public CSocket
{
public:
CListeningSocket(CPool* ppool);
  virtual void OnAccept(int nErrorCode);
protected:
CPool* pPool;
}
```

Класс также хранит указатель на своего владельца и имеет такой же конструктор, какой и CClientSocket. Для добавления функциональности достаточно переопределить всего одну функцию:

```
void CListeningSocket::OnAccept(int
  nErrorCode)
{
pPool->ProcessAccept();
CSocket::OnAccept(nErrorCode);
}
```

Она вызывается системой при запросе нового соединения. В отличие от предыдущего класса информирующая функция ProcessAccept() не передает указатель this. Но это и не требуется, поскольку объект данного типа существует в единственном экземпляре.

Как можно увидеть, вся деятельность описанных классов связана в основном с информированием CPool о том, что с ними происходит важного. Рассмотрим, наконец, сам класс CPool. Он не имеет предков и наделен широкими полномочиями.

```
class CPool
```

```
{
public:
  CPool();
  virtual ~CPool();
  BOOL Init(void *phost, int port);
  BOOL Send2All(CString data);
  void ProcessClose(CClientSocket *pSocket);
  void ProcessRead(CClientSocket* pSocket,CString msg);
  int ProcessAccept();
protected:
  CListeningSocket *m_pSocket;
  CPtrList connectionList;
  void *pHost;
};
```

Для хранения клиентских сокетов, используем CPtrList. Это достаточно удобно и в нашем случае эффективнее, чем, скажем, CPtrArray. Для связи с остальными частями программы CPool имеет указатель на некоторый внешний объект pHost.

Конечно, в классе объявлен указатель на слушающий сокет; как и другие его "двоюродные братья", этот объект также будет создан динамически. Перейдем к функциям.

```
BOOL CPool::Init(void *phost,int port)
{
  pHost=(CChildView*)phost;
  m_pSocket = new CListeningSocket(this);
  if (m_pSocket->Create(port))
  {
    if(m_pSocket->Listen())
      return TRUE;
  }
  return FALSE;
}
```

Нет смысла записывать производимые здесь действия в конструктор, так как можно к этому моменту не знать номера порта, а кроме того, до начала активных действий с сокетом необходимо вызвать AfxSocketInit(), функцию, которая может вернуть ошибку. Если вам не хочется самим набирать эту часть кода, воспользуйтесь компонентом из Component Gallery. Он добавит инициализацию сокетов в функцию InitInstance() класса приложения. То же самое будет сделано, если в AppWizard попросить поддержку сокетов. Запуск сокета на прослушивание "эфира" осуществляет функция Listen(), а номер порта, на котором сокет будет это делать, передается через функцию Create().

Следующая функция

```
int CPool::ProcessAccept()
{
CClientSocket* pSocket =
```

```
  new CClientSocket(this);
if (m_pSocket->Accept(*pSocket))
{
connectionList.AddTail(pSocket);
   pHost->UpdateView();
}
else
   delete pSocket;
  return 0;
}
```

Добавляет созданный CClientSocket в хранилище (функция AddTail) и сигнализирует о необходимости обновить информацию об имеющихся подключениях. Вообще все вызовы функций объекта pHost не относятся непосредственно к механизму взаимодействия сокетов и добавлены исключительно ради придания жизни исходному коду. Передача соединения от слушающего сокета клиентскому происходит в m_pSocket->Accept(*pSocket).
Обработка отключения клиента реализована в

```
void CPool::ProcessClose(CClientSocket *pSocket)
{
POSITION pos = connectionList.Find(pSocket);
connectionList.RemoveAt(pos);
delete pSocket;
  pHost->UpdateView();
}
```

На функции ProcessRead, по идее, должна лежать основная нагрузка по разбору полученного, но, поскольку обмен происходит только строковыми данными, здесь больше и делать нечего.

```
void CPool::ProcessRead(CClientSocket *pSocket,CString msg)
{
  pHost->UpdateView();
  pHost->editbox.AddText(": "+msg);
}
```

В большинстве реальных приложений может оказаться необходимой возможность посылки одного сообщения всем подключенным на данный момент клиентам. Эту задачу выполняет функция Send2All(CString data).

```
BOOL CPool::Send2All(CString data)
{
  if(data.GetLength()==0)
    return FALSE;
  for(POSITION pos=connectionList.
    GetHeadPosition();pos != NULL;)
  {
CClientSocket* pSock = (CClientSocket*)
  connectionList.GetNext(pos);
    pSock->Send(data,data.GetLength());
```

```
  }
  return TRUE;
}
```

Из connectionList поочередно берется указатель на клиентский сокет, и, чтобы послать данные, вызывается функция Send, которой необходимо передать указатель на источник данных и его размер в байтах. В приведенном выше коде data - это переменная типа CString. Здесь стоит отметить, что если объем данных больше, чем можно передать за один раз, то исходный блок будет разбит и передан по частям. В этом случае функция OnRecieve вызывается несколько раз, а при использовании CAsyncSocket будет вызвана еще и OnOutOfBandData, извещающая о фрагментации данных. Максимальный размер блока в SDK и прочих документациях указывать не любят. Еще наполним содержанием деструктор, где освободим всю выделенную память.

```
CPool::~CPool()
{
  while(!connectionList.IsEmpty())
  {
  CClientSocket*pSocket=(CClientSocket*)
    connectionList.RemoveHead();
  if(pSocket!=NULL)
  {
    pSocket->Close();
    delete pSocket;
  }
  }
  m_pSocket->Close();
  delete m_pSocket;
}
```

Теперь, когда мы рассмотрели все необходимые классы, проследим за последовательностью действий, выполняемых при подключении нового клиента. При получении требования на соединение вызывается функция CListeningSocket::OnAccept, она, в свою очередь, вызывает ProcessAccept, принадежащую CPool. В ProcessAccept создается новый объект класса CClientSocket. С этого момента он и отвечает за взаимодействие с клиентом, а слушающий сокет снова готов обрабатывать запросы о новых подключениях. Вот, пожалуй, и все, что касается сервера.

Действия, выполняемые на стороне клиента, совсем просты. Но, на всякий случай, приведу и их. Для наглядности в класс сокета я добавил взаимодействие с окном диалога. Даже с конкретной привязкой к реализации программы код практически не увеличился в объеме. Переопределяем уже знакомые функции. Единственное поле данных - указатель на объект класса, производного от CDialog. Это конкретная реализация pHost из предыдущих классов.

```
class MySocket : public CSocket
{
```

```
public:
  MySocket(CClientDlg *pDlg)
    {m_pDlg = pDlg;};
  virtual void OnClose(int nErrorCode);
  virtual void OnReceive(int nErrorCode);
  CClientDlg *m_pDlg;
};
void MySocket::OnClose(int nErrorCode)
{
  m_pDlg->GetDlgItem(IDC_CONNECT)->
    EnableWindow(TRUE);
  delete this;
  CSocket::OnClose(nErrorCode);
}
void MySocket::OnReceive(int nErrorCode)
{
  char st[4096];
  int r=Receive(st, 4096);
  st[r]='\0';
  m_pDlg->SetDlgItemText(IDC_ANSWER,st);
  CSocket::OnReceive(nErrorCode);
}
```

IDC_ANSWER и IDC_CONNECT - идентификаторы соответствующих ресурсов, первый - окно редактирования, второй - кнопка, нажатием на которую клиент пытается соединиться с сервером. Если это получается, то кнопка блокируется. Это действие реализуется следующим образом:

```
pSocket = new MySocket(this);
pSocket->Create();
if(pSocket->Connect("127.0.0.1", port))
{
  GetDlgItem(IDC_CONNECT)->EnableWindow(FALSE);
  pSocket->Send(Name,Name.GetLength());
}
else
  delete pSocket;
```

Функции Connect необходимы следующие два параметра: LPCTSTR lpszHostAddress - адрес узла сети, он может быть числовым ip-адресом (я использовал значение 127.0.0.1, так как оно представляет собой адрес локального компьютера) или буквенным, т. е. DNS-адресом, например «someserver.ru». Второй параметр - номер порта. При разрыве соединения кнопка снова активизируется. Что мы и делаем в OnClose().

Контрольные вопросы

1. Для чего нужны Socket?
2. Как добавить класс работающий с Socket?

3. Как настроить класс CSocket для получения сообщений с сервера ?
4. Как передать сообщение из класса CSocket в диалоговое окно?
5. Как передать сообщение серверу?
6. Как получить сообщение от сервера, какие операторы отвечают за это?

Лабораторная работа №9
Знакомство с платформой 1С: Предприятие. Создание конфигурации, объектов конфигурации: справочников, документов, регистров, отчетов.

Продолжительность 2 часа.

Цель работы: изучить способы создания баз данных, состав конфигурации и задание имен в интерфейсе, связь между базой на диске и в интерфейсе конфигурации.

Задание: создать учебную базу данных, для последующей работы в ней. создать в ней объект типа справочник, создать в справочнике реквизиты разных типов. Создайте объект типа перечисление, создайте необходимые формы и заполните данными справочник в режиме 1С: Предприятие. создать в , базе один или несколько документов, разработать для них формы интерфейса с пользователем, а также макеты печатных форм для вывода бумажных документов

Выполнение работы:

1. Создать на диске папку в которой будет находится ваша конфигураций 1С: Предприятие..
2. Запустить 1С: Предприятие и выполнить все необходимые шаги для создания своей конфигурации.
3. Запустить 1С: Предприятие в режиме конфигуратора и ознакомится с перечнем возможных объектов.
4. Запустить полученную конфигурацию в режиме 1С: Предприятие.
5. Изменить в конфигураторе свойство: «Основной режим запуска», запустить в режиме 1С: Предприятие.
6. Создайте объект конфигурации – подсистема (например: финансы, сотрудники, контакты и т.д.). Запустите в режиме 1С: предприятие.
7. Создать объект типа справочник назвать в соответствии с предложенным вариантом и отнести его к подходящей по смыслу подсистеме.
8. Создать необходимый набор реквизитов, используя базовые типы данных языка.
9. Создать объект типа перечисление.
10. Создать реквизиты в качестве типа, в которых используется другой объект конфигурации.
11. Создать таблицу и табличные реквизиты.
12. Создать форму для интерфейса в режиме 1С: Предприятие.

13. Заполнить справочник данными.
14. Создать документы заданного вида как объект конфигурации .
15. Создать необходимое количество реквизитов разных типов.
16. Создать таблицу и табличные реквизиты.
17. Создать объект конфигурации журнал для учета документов.
18. Проверить работу созданных объектов.
19. Создать обработчик события в документе.
20. Сконструировать печатную форму документа.
21. Дополнить печатную форму своими данными
22. Создать журнал для регистрации документов созданных в предыдущей лабораторной работе.
23. Создать регистр накопления для документа созданного в предыдущей лабораторной работе, например отражающий движение денежных средств и материалов.
24. Создать несколько документов, посмотреть как в регистрах отражается проведение документов.
25. Создать отчет отражающий полученные денежные средства, за определенные периоды.
26. Создать отчет произведенных затрат по периодам агентам.
27. Создать отчет по полученной прибыли и оборотам..
28. Создать отчет по движению товаров в натуральном выражении.
29. **Написать отчет о проделанной работе.**

Пояснения к выполнению работы:

Для успешного выполнения работы рекомендуется сначала ознакомится с учебным пособием [7].

Контрольные вопросы:

1. Для чего на форме запуска 1С две кнопки «Конфигуратор» и «1С Предприятие»?
2. Как добавить новую конфигурацию в список?
3. Как узнать в какой папке находится та или иная конфигурация из списка?
4. Как открыть конфигурацию для редактирования?
5. Для чего служит режим 1С: Предприятие?
6. Для чего служит режим «Конфигурации»?
7. Как присвоить имя конфигурации?
8. Как соотносятся между собой имя конфигурации, имя в интерфейсе запуска и имя папки на диске в котором находится база данных?
9. Как создать новый объект конфигурации стандартного типа?
10. Как создать новый объект конфигурации стандартного типа?
11. Для чего нужны объекты типа подсистема?
12. Что такое справочник в конфигурации?
13. Для чего используются объекты типа справочник?

Порядок оформления отчета

Отчет оформляется по результатам выполнения лабораторной работы на стандартных листах бумаги формата А4 с помощью редактора Word. Оформление отчета является важной частью лабораторной работы. При работе над отчетом студент должен осмыслить ход выполнения работы и научиться документировать полученные результаты.

Отчет должен содержать блок-схемы алгоритмов, оформленные по ГОСТу, необходимые структуры данных и распечатки программных кодов, разработанные студентом.

В отчете могут быть приведены тестовые данные, использовавшиеся студентом для проверки правильности работы программы. По результатам проверки программы должны быть сделаны выводы. Если в работе получились не те результаты, которые ожидались, в отчете должны быть отмечены причины, по которым не удалось получить желаемые результаты.

Допускается оформление одного отчета по всему курсу, выполненных в течение семестра лабораторных работ.

Пример оформления титульного листа для лабораторной работы приведен в Приложении 1.

Список рекомендуемой литературы

1. Давыдов В.Г. Программирование и основы алгоритмизации: Учебное пособие. - М.: Высшая школа , 2003.

2. Холзнер С. Microsoft Visual C++6 с самого начала. – СПб.: Питер, 2005.

3. Либерти Джесс. Освой самостоятельно C++ за 21 день.- М.: Издательский Дом "Вильяме", 2001. - 814 с.

4. Радченко М.Г. 1С:Предприятие 8.2 практическое пособие разработчика. Примеры и типовые решения / М.Г. Радченко, Е.Ю. Хрусталева - М.: ООО « 1С-Паблибшинг», 2009.

5. 1С:Предприятие 8.2 Руководство разработчика. Ч.1, Ч.2. Москва. Фирма «1С». 2009.

6. http://1c-uroki.ru/

7. Скобелев И.В., Грибанов К.Н., Фалев В.В. Объектно-ориентированное программирование автоматизированных систем управления на платформе 1С: Предприятие 8.2. Учеб. пособие. - Курган: Издательство Курганского гос. ун-та, 2013.- 107 с.

8. Профессиональная разработка в среде 1С Предприятие 8. / под ред. Радченко М.Г. - « 1С-Паблибшинг», СПб.: Питер, 2006.

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматизации производственных процессов

ОТЧЕТ
по лабораторным работам
по дисциплине
«Программирование и алгоритмизация»

Выполнил:
студент гр. 214 Иванов И.И.

Проверил:
преподаватель Скобелев И.В.

Курган 2014

Скобелев Игорь Вадимович

ПРОГРАММИРОВАНИЕ И АЛГОРИТМИЗАЦИЯ

Методические указания
к комплексу лабораторных работ
для студентов направлений 15.03.04 (220700.62)

Авторская редакция

Подписано в печать 18.03.15	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ.л. 2,0	Уч.-изд. л. 2,0
Заказ 63	Тираж 25	Не для продажи

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.