Министерство образования Российской Федерации
Санкт-Петербургский политехнический университет

В.Ю. Сальников

Программирование под Windows.

Методические указания по проведению лабораторных работ по курсу «Методы программирования и прикладные алгоритмы», «Компьютерные технологии в приборостроении» и «Технологии программирования»

Санкт-Петербург
2011

ББК ??.???

УДК ???.???.?? (???.?)

Сальников В.Ю.
Программирование под Windows: Методические указания по проведению лабораторных работ. -2011г. - 22 с.

Данное руководство написано для студентов 2-го курса, изучающих дисциплину «Методы программирования и прикладные алгоритмы», «Технологии программирования» или «Компьютерные технологии в приборостроении» на кафедре ИИТ ФТК СПбГПУ.

Основная цель курса заключается в формировании навыков написания программ на языке C/C++ под Windows с использованием Win API 32 и MFC.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Программирование под Windows основывается на практических навыках грамотного использования основной движущей силы Windows – сообщениях.

В двух больших лабораторных работах курса сделана попытка по возможности полно охватить основные управляющие элементы Windows, базовые принципы управления ими, изучение основ программирования с использованием MFC, и в частности, Document-View модели.

Изучение курса требует хорошего знания языка программирования C (API) и C++ (MFC).

Автор

Лабораторная работа № 1.

Разработка программы редактирования файлов специализированного формата.

Целью данной работы является написание тестового примера с использованием Windows API 32.

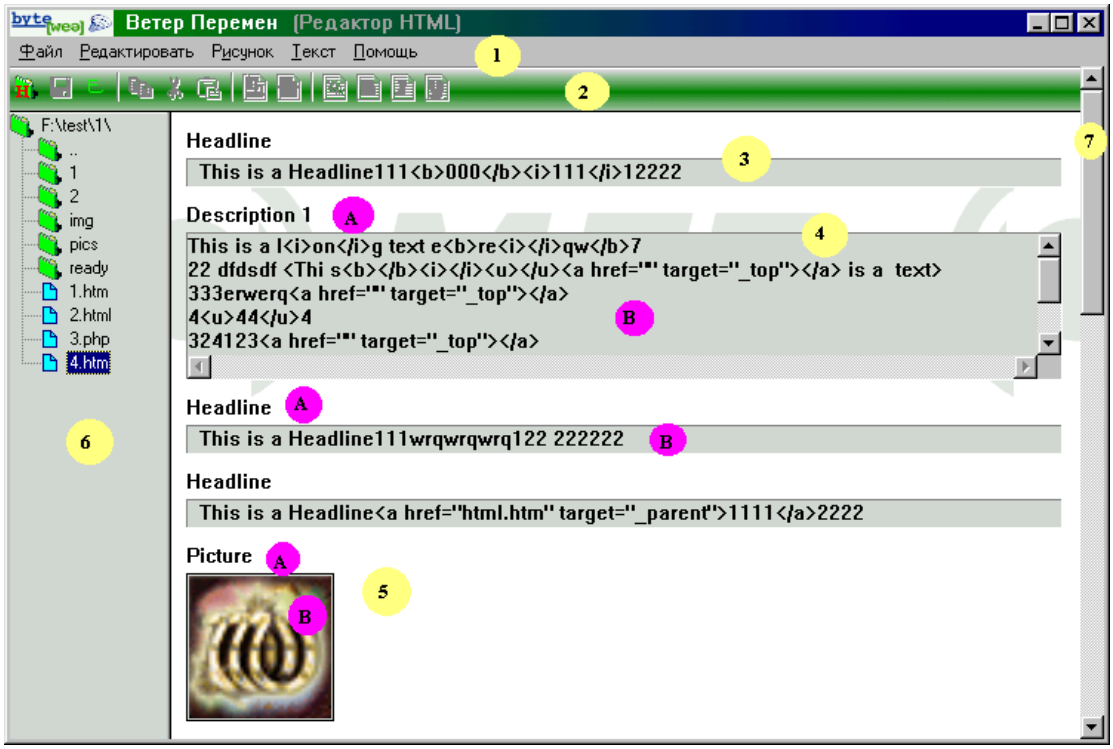
Исходные данные.

1. Программа является оконным Windows32 приложением.

2. Программа написана на языке Си++ в среде Visual C++ 6.0, VS.NET,VS2005,VS2008 или VS2010.

Требования к внешнему виду программы.

1. Типовой рекомендуемый внешний вид приложения:



На рисунке нанесены условные обозначения, описанные далее.

Размер окна по вертикали и горизонтали может меняться пользователем, но он не может быть меньше размера 400 на 200 пикселей.

Позицию и размеры окна на момент выхода из программы необходимо сохранять в реестре. При старте программы их необходимо читать из реестра (если есть) и отображать окно в соответствующей позиции и с соответствующими размерами.

2. Смысл программы сводится к поиску-открытию-редактированию-записи файлов специализированного формата.

Формат файла:

Заголовок

Элемент 1

Элемент 2

Элемент 3

…

Файл является текстовым, т.е. содержит только символы ASCII набора.

Расширение у файла может быть “txt”, “htm”, “html”, “php”.

Заголовок – строка “<!--start_bwc-->”, расположенная где-то в первой строке файла.

Любой элемент имеет следующий формат:

<!--start_edit type="####" name="$$$$$" body=”%%%”-->

где:

- тип элемента, может быть одним из:

- shorttext
- longtext
- image

$$$$$ - имя элемента, может быть любым текстом в одну строку

%%% - тело элемента, зависит от типа элемента.

Существует 3 типа элементов:

А) Текст в одну строку.
Поле type=”shorttext”
Поле body содержит любую (одну) строку текста

Б) Текст в несколько строк.
Поле type=”longtext”
Поле body содержит любую текст в несколько строк

В) Картинка в формате BMP
Поле type=”image”
Поле body содержит относительный путь к файлу с картинкой, например "img\1a.bmp".

Элементы следуют друг за другом. Между элементами может находиться любой текст, который не обрабатывается программой.

Редактируемыми элементами являются тексты (body) для элементов shorttext и longtext. Остальные элементы (как и текст вне элементов) не должны меняться при редактировании и сохранении файла.

3. Программа содержит меню (желтый кружок с цифрой 1), в котором в удобной для разработчика форме представлены след. команды (в скобках указаны Hotkey комбинации клавиатуры - ускорители):

- сменить активный каталог (Ctrl+D)
- записать редактируемый файл (Ctrl+S)
- выйти из программы (Alt+F4)
- копировать выделенный текст в буфер (Ctrl+C)

- вырезать выделенный текст и поместить в буфер (Ctrl+X)
- вставить в указанное место (заменить выделенный текст) текстом из буфера (Ctrl+V)
- отменить последнюю операцию по вставке/замене (Ctrl+Z)
- вывод информации о программе и авторах

4. Программа содержит Toolbar (желтый кружок с цифрой 2) со след элементами:
- сменить активный каталог
- записать редактируемый файл
- выйти из программы
- копировать выделенный текст в буфер
- вырезать выделенный текст и поместить в буфер
- вставить в указанное место (заменить выделенный текст) текстом из буфера
- отменить последнюю операцию по вставке/замене

5. В правом поле программы отображены Элементы один за другим сверху вниз. Размер всех элементов одного формата фиксированный.

А) Для Элемента shorttext (желтый кружок с цифрой 3) это одна строка с именем элемента (лиловый кружок с буквой А) [не редактируемое] и поле с телом Элемента в одну строку (лиловый кружок с буквой В) [редактируемое].

Б) Для Элемента longtext (желтый кружок с цифрой 4) это одна строка с именем элемента (лиловый кружок с буквой А) [не редактируемое] и поле с телом Элемента в 4-6 строк (лиловый кружок с буквой В) [редактируемое].

В) Для Элемента image (желтый кружок с цифрой 5) это одна строка с именем элемента (лиловый кружок с буквой А) [не редактируемое] и поле с картинкой по ссылке, указанной в теле Элемента (лиловый кружок с буквой В) [не редактируемое].

Размер вывода тела для Элементов shorttext и longtext определяется текущим размером окна. В случае изменения горизонтальных размеров окна размер полей редактирования (лиловый кружок с буквой В) также изменяется программой автоматически.

6. В левом поле программы отображено дерево с именами файлов и папок.

Вершиной дерева является текущий каталог, который может быть изменен (выбран) через меню или toolbar (сменить активный каталог).

Папки, расположенные в каталоге демонстрируются все.

Файлы показываются только те, что задержат заголовок спец. формата.

Папки показываются отсортированными по алфавиту, и все папки расположены до файлов.

При клике мышкой на файле информация об его Элементах отображается в правой части.

При двойном клике на имя папки – она становится активным каталогом.

При двойном клике на папку “..” – родительская папка (если есть) становится активным каталогом.

Двойном клик на активный каталог воспринимается как команда меню “сменить активный каталог”.

При выходе из программы имя текущего активного каталога и активного (просматриваемого) файла сохраняются в реестре.

При старте программы информация об активном каталоге и активном файле берется из реестра (если есть).

7. Если отображение всех элементов файла не помещается по вертикали, то у окна автоматически появляется scrollbar (желтый кружок с цифрой 7). Его настройки устанавливаются программой автоматически таким образом, чтобы при его использовании можно было увидеть все элементы в правой части окна (осуществлять прокрутку содержимого).

Рекомендации по написанию программы.

1. Начальные действия.

Запустите Visual C++.

Menu -> File -> New -> Projects -> Win32 Application

Задайте Имя проекта и его расположение.

Выберите A Typical “Hello, World!” Application.

Для VS.NET или VS2005

Menu -> File -> New -> Projects -> Win32 Project

Задайте Имя проекта.

Проверьте, что в закладке Application Settings НЕ стоит галочка в пункте Empty Project.

На закладке FileView в папке Source Files вы найдете три файла:

xxx.cpp

xxx.rc

StdAfx.cpp

Файл xxx.cpp – это файл в котором представлен начальный код, сгенерированный для вас автоматически.

Файл xxx.rc – это файл с ресурсами проекта (их редактирование осуществляется на закладке ResourceView).

Файл StdAfx.cpp как и файл StdAfx.h создается и изменяется средой автоматически и у вас нет необходимости его менять.

На закладке FileView в папке Header Files вы найдете три файла:

xxx.h

resource.h

StdAfx.cpp

Файл xxx.h – это файл в котором представлены прототипы функций и объявления глобальных данных; начальное содержимое этого файла также генерируется для вас автоматически.

Файл resource.h – это файл с константами, установленными для ваших ресурсов, которые необходимы для доступа к ресурсам из кода вашей программы (обычно этот файл изменяется средой автоматически и у вас нет необходимости его менять)

В процессе разработки программы рекомендуется использовать навыки построения проектов, полученные в пред. семестре. Т.е. при создании нового класса или набора связанных функций рекомендуется создавать пару файлов yyy.cpp и yyy.h и включать их в проект. Таким образом вы значительно улучшите «читаемость» кода программы и тем самым облегчите себе весь процесс разработки программы.

При таком подходе при включении файла yyy.cpp в проект разместите в верхней самой первой строке файла след. инструкцию:

#include "stdafx.h"

и при построении проекта запустите Rebuild All.

В функции InitInstance добавьте вызов функции InitCommonControls, которая инициализирует библиотеку ComCtrl, т.к. у вас будут использованы как минимум два ее элемента. Так как по умолчанию библиотека Common Controls не подключается, необходимо сделать два дополнительных действия.

- в верху файла вставить след. строку:

#include “commctrl.h”

- в свойствах проекта в подпункте Linker (можно прямо в закладке Command Line) добавить файл comctl32.lib.

Эти действия заставят линкер подключить функции библиотеки Common Control в ваш проект, а в H файле представлены прототипы функций и структур Common Control для того, чтобы компилятор мог их видеть.

2. Так как программа должна обрабатывать файлы спец. формата и содержимого переменной длины, рекомендуется использовать классы со списками элементов.

Для хранения всех данных файла рекомендуется создать класс File и класс Item (элемент).

Для построения списков можно использовать либо поддержку списков внутри этих классов, либо добавить классы FileList и ItemList.

Список файлов вам пригодится для отображения элементов в дереве файлов и каталогов и он же поможет получить доступ к информации выбранного файла для отображения в правой части окна.

Список элементов поможет вам отображать все элементы в правой части окна. Логично, чтобы File содержал указатель на вершину списка его элементов.

Не забывайте также, что File должен поддерживать механизм чтения его содержимого (и формирования списка элементов) и записи.

3. Работа с файлами.

Для построения списка файлов рекомендуется использовать пару функций: _findfirst и _findnext и, соответственно структуру _finddata_t.

Не забывайте, что поиск надо осуществлять в активном каталоге.

Отбор файлов осуществляется по расширению, путем анализа поля Name в структуре _finddata_t и затем поиском заголовка в первой строке файла (рекомендуемые функции: fopen, fgets и strstr).

4. Работа с Элементами файлов.

Для составления списка элементов рекомендуется загрузить (возможно, временно) весь файл целиком и осуществлять поиск элементов там функцией strstr.

Возможен след. вариант интерпретации информации файлов:

- для каждого элемента запоминается смещение от начала файла до начала и конца элемента;
- при сохранении эти маркеры используются для записи вся информация «между» элементами и самих элементов.

5. Отображение дерева файлов и папок.

Для отображения используйте TreeView Control.

Создание: CreateWindowEx с типом WC_TREEVIEW

Возможно, вам понадобится родительское окно для отображения дерева (зависит от подхода), тогда вам нужно зарегистрировать свой класс (RegisterClass, UnregisterClass), создать окно (CreateWindowEx) и указать его как родительское для окна-дерева.

Вам понадобиться список картинок для отображения элементов дерева с иконками. Он формируется след функциями: ImageList_Create, LoadImage, ImageList_Add, DeleteObject.

Вам понадобиться обрабатывать след. сообщения (в родительском или главном окне):

WM_NOTIFY с кодом TVN_SELCHANGED (здесь нужны лишь обработки действий TVC_BYKEYBOARD и TVC_BYMOUSE, которые находятся в структуре NM_TREEVIEW, указатель на которую передается как lParam для сообщения) и NM_DBLCLK (код находится в структуре NMHDR, указатель на которую в свою очередь передается как lParam для сообщения) и, возможно, WM_SIZE.

Заметьте, что указатель на NM_TREEVIEW является также и указателем на NMHDR для окна-дерева.

Добавление элемента в дерево осуществляется посылкой сообщения (SendMessage) типа TVM_INSERTITEM окну-дереву. При этом вам понадобятся структуры TV_INSERTSTRUCT и TV_ITEM.

Изменение отображаемой для уже добавленного элемента информации (если понадобится), осуществляется функцией TreeView_SetItem.

Удаление элемента из дерева: TreeView_DeleteItem.

Удаление всех элементов из дерева: TreeView_DeleteAllItems.

Для сортировки уже добавленных элементов используйте функцию TreeView_SortChildren.

Для выделения активного элемента в дереве используйте функцию TreeView_SelectItem.

При смене активного элемента в дереве вам могут также понадобиться функции:

TreeView_GetNextItem и TreeView_Expand.

6. Отображение Элементов.

Каждый элемент представляет собой два дочерних окна: с именем и содержимым элемента.

Рекомендуется хранить:

- описатель окна;
- его положение и размеры.

Именем может быть обычный статический элемент, который создается с помощью CreateWindowEx с типом "STATIC" и с атрибутом SS_SIMPLE, его содержимое устанавливается посылкой сообщения WM_SETTEXT.

Тело Элемента shorttext отображается в окне Edit (создается CreateWindowEx с типом “EDIT” без атрибута ES_MULTILINE). Установка текста – посылка сообщения WM_SETTEXT.

Тело Элемента longtext отображается в окне Edit (создается CreateWindowEx с типом “EDIT” с атрибутом ES_MULTILINE). Установка текста – посылка сообщения WM_SETTEXT.

Изменение размеров для окон Edit (при изменении размеров окна) осуществляется функцией SetWindowPos.

Для отображения картинки рекомендуется создавать окно-Кнопку (создается CreateWindowEx с типом “BUTTON” с атрибутом BS_BITMAP и BS_FLAT).

Для загрузки картинки в BMP формате используйте LoadImage, для освобождения ресурса (после использования) DeleteObject.

Прорисовать загруженную картинку на кнопке можно посылкой кнопке сообщения BM_SETIMAGE.

Перемещение всех окон осуществляется функцией SetWindowPos.

Разрушение всех окон производится с помощью DestroyWindow.

7. Меню.

Структура меню задается в редакторе ресурсов.

Активная клавиша для навигации внутри меню задается путем размещения символа & перед соответствующей буквой. В меню это будет выглядеть как подчеркивание данной буквы.

Каждому элементу меню сопоставляется некоторый идентификатор (константа, автоматически будет помещаться resource.h). Этот идентификатор формируется автоматически, но вы можете задать любое вам удобное имя константы.

Во время работы программы, если пользователь выбирает некоторый элемент меню, функции обработки сообщений окна-хозяина будет автоматически послано сообщение WM_COMMAND с кодом-константой элемента меню. Таким образом, для задания обработки необходимых команд меню вам требуется добавить соответствующие секции в процедуре обработки сообщений. В качестве примера рекомендуется посмотреть обработку сообщений меню, которое автоматически генерируется при формировании приложения “Hello, World!”.

8. Toolbar.

Toolbar – стандартизованное средство Windows, поэтому его создание достаточно просто.

Легче всего воспользоваться функцией CreateToolbarEx для его создания. Вам также понадобится заполнить структуру TBBUTTON, указав там константы соответствующих элементов меню (т.к. в вашем случае все элементы Toolbar будут иметь соответствующий элемент меню) и в ресурсах нарисовать кнопки как элемент Toolbar в ресурсах.

В таком случае никаких дополнительных обработчиков сообщений от Toolbar не требуется.

Однако, если вы хотите создать flat Toolbar (по аналогии с тем, что на рисунке), то создавать его следует с атрибутом TBSTYLE_FLAT, а также добавить обработку в сообщение WM_NOTIFY, подтип NM_CUSTOMDRAW. Там необходимо поле dwDrawStage в структуре LPNMTBCUSTOMDRAW (указатель на которую передается в параметре lParam) проверить на тип CDDS_PREERASE и перерисовать задний план окна Toolbar (в случае на картине используется функция GradientFill) и вернуть значение CDRF_SKIPDEFAULT (что предотвратит перерисовку по умолчанию). Если вы не добавите подобный обработчик, то задний план окна Toolbar не будет перерисовываться вообще.

Кроме того, вы можете добавить текст-подсказку для каждой кнопки. Для этого необходимо создавать Toolbar с атрибутом TBSTYLE_TOOLTIPS. При этом вам также понадобиться добавить обработчику сообщения WM_NOTIFY, подтип TTN_NEEDTEXT. Там необходимо установить поле lpszText в структуре LPTOOLTIPTEXT (указатель на которую передается в параметре lParam) в желаемый текст. Индекс кнопки передается в поле hdr.idFrom той же структуры.

9. Scrollbar (полоса прокрутки).

Scrollbar так же является стандартным средством Windows, поэтому весь механизм поддержки прокрутки стандартизирован.

При создании основного окна программы укажите атрибут WS_VSCROLL.

Далее каждый раз при переходе к след. файлу используйте функцию SetScrollInfo для установки параметров полосы прокрутки. Вам понадобится заполнить структуру SCROLLINFO.

Также необходимо добавить обработчики сообщений WM_VSCROLL, подтипы SB_TOP, SB_BOTTOM, SB_LINEDOWN, SB_LINEUP, SB_PAGEDOWN, SB_PAGEUP, SB_THUMBPOSITION, SB_THUMBTRACK. В зависимости от подтипа вы определяете тип перемещения полосы прокрутки и:

- перемещаете все элементы основного окна в нужном направлении на нужную величину (см. п.6);
- установить нужную позицию самой полосы прокрутки функцией SetScrollPos;
- вызвать функцию обновления графической информации для главного окна: UpdateWindow.

Кроме того необходимо добавить симуляцию перемещения полосы прокрутки с клавиатуры. Для этого добавьте обработчик сообщений типа WM_KEYDOWN. Тип нажатой клавиши в виде виртуального кода передается в wParam. В случае кодов VK_PRIOR и VK_NEXT (PgUp и PgDwn) следует производить описанные выше манипуляции с прокруткой.

10. Работа с реестром.

Для создания ключа в реестре используйте функцию RegCreateKeyEx. Здесь и далее ВСЕГДА используйте ТОЛЬКО ветку HKEY_CURRENT_USER (см. документацию).

Для определения существования ключа и получения его описателя используйте функцию RegOpenKeyEx.

Для сохранения параметров в реестре используйте функцию RegSetValueEx.

Для чтения параметров – RegQueryValueEx.

11. Редактирование текста.

Редактирование текста в окнах типа Edit осуществляется автоматически. Однако, для того, чтобы операции по копированию частей текста можно было осуществить через меню и Toolbar, требуется добавить соответствующие обработчики (константы определяются в меню). Окна Edit полностью поддерживают все операции путем обработки сообщений WM_CUT, WM_COPY, WM_PASTE, которые вы должны посылать им. В последнем случае вам понадобится проверить буфер обмена на наличие в нем текста. Используйте для этого функцию IsClipboardFormatAvailable с параметром CF_TEXT.

Кроме того, при отображении информации в окне Edit переводом строки считается сочетание CR (13) и NL (10). При чтении файла в текстовом режиме (режим открытия файла содержит флаг “t” в функции fopen) символы CR,NL заменяются при чтении одним NL. Таким образом при чтении и записи необходимо производить преобразование строк. Это удобно сделать путем создания соответствующего класса. Если же вы открываете файл в двоичном режиме (режим открытия файла содержит флаг “b” в функции

fopen) символы CR,NL так и читаются и транслировать текст при размещении в EDIT контроле не надо.

Для того, чтобы работали ускорители, необходимо направлять сообщения на соответствие ускорителям в главное окно приложения, а не в текущее (по умолчанию в цикле обработки сообщений стоит именно текущее).

Лабораторная работа № 2.
Разработка программы просмотра изображений с использованием библиотеки MFC.

По результатам данной работы студенты получат знания о написании программ под Windows с использованием библиотеки классов MFC и технологии Document-View.

Исходные данные.

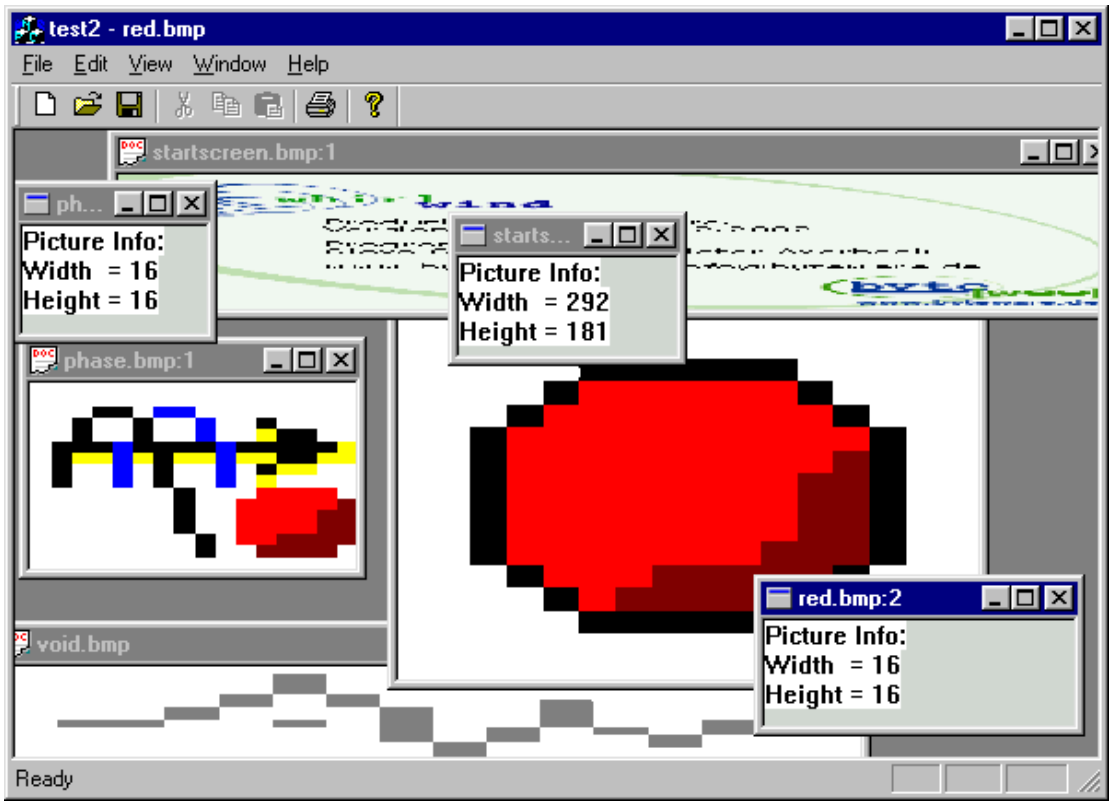
1. Программа является оконным Windows32 приложением.

2. Программа написана на языке Си++ в среде Visual C++ 6.0, VS.NET или VS2005.

3. При написании программы используются функции, классы и структуры MFC и Win API32(если понадобиться). Никаких других технологий или «надстроек» не должно быть использовано.

Требования к внешнему виду программы.

1. Типовой рекомендуемый внешний вид приложения:



2. Основное назначение программы:

- отображение BMP файлов в двух форматах по выбору: как рисунок и как информация о рисунке (реальный размер);
- в случае просмотра рисунка его размер должен изменяться автоматически с изменением окна просмотра;
- количество одновременно открытых файлов не ограничено;
- если файл уже открыт и производится попытка его повторного открытия, то соответствующий файл вновь не открывается, а окно просто перемещается на передний план.

Рекомендации по написанию программы.

1. Начальные действия.

Запустите Visual C++.

Menu -> File -> New -> Projects -> MFC AppWizard (exe)

Задайте Имя проекта и его расположение.

Выберите:

- Multiple Documents (step 1 of 6)
- Document/View architecture support (step 1 of 6)
- No database support (step 2 of 6)
- No compound document support (step 3 of 6)
- No Automation (step 3 of 6)
- Not an ActiveX control (step 3 of 6)
- Docking Toolbar (step 4 of 6)
- Initial Status bar (step 4 of 6)
- Printing and Print preview (step 4 of 6)
- NO Context-sensitive Help (step 4 of 6)
- 3D Controls (step 4 of 6)
- NO MAPI (step 4 of 6)
- NO Windows Socket (step 4 of 6)
- Toolbar looks Normal (step 4 of 6)
- MFC Standard project style (step 5 of 6)
- Do not change any file names (step 6 of 6)

В средах VS.NET, VS2005 последовательность действий практически идентична.

2. Создание класса для поддержки BMP файлов.

Добавление чего-то, что вносит существенные изменения, требует создания дополнительного класса. В нашем случае это поддержка BMP файлов: загрузка и отображение.

Для добавления классов удобно использовать ClassWizard:

- запускаем ClassWizard (Menu -> View -> ClassWizard);
- нам нужен новый класс, поэтому выбираем Add Class ->New…;
- называем его MyBMP;

- в качестве базового нам подходит самый простой класс CObject, однако такого в списке нет и приходится выбирать наипростейший, например, CStatic.

В VS.NET и VS2005 ClassWizard отсутствует в явном виде, но функциональность вся тем не менее поддерживается. В частности добавить класс здесь можно в окне Solution Explorer в закладке Class View выбрать корневой элемент (название проекта) и в меню по правой кнопке мыши выбрать элемент “Add” -> “Add Class…”, далее в Categories выбрать MFC, в правой части окна MFC Class. Далее Open, задаем имя класса и в качестве базового выбираем CObject.

В результате в проекте появятся два файла MyBMP.cpp и MyBMP.h

Теперь «упростим» наш класс, т.е. уберем все лишнее:

- в качестве базового класса укажем CObject вместо CStatic;
- удалим карту сообщений и оставим только Конструктор и деструктор.

Теперь при любой попытке войти в ClassWizard, он будет ругаться, что не может найти информацию о данном классе и предложить найти ее вручную или исключить класс из списка классов, контролируемых Wizardom. Нам подходит последнее, т.к. все, что надо мы и вручную добавим.

Проблема не появилась бы, если использовать среду Visual C++ 7.0, т.к. там есть CObject класс в качестве непосредственного базового (так и есть).

Другим путем создания класса (впрочем с теми же трудностями) является использование закладки ClassView в проекте. Там можно также использовать контекстное меню Add Class.

Заметьте также, что добавление данного класса можно было осуществить и вручную, а не с использованием ClassWizard и результат был бы таким же.

3. Наполнение MyBMP класса и первый тест

Первым делом убедимся в работоспособности нашего класса. Для этого добавим в него функцию прорисовки:

```
BOOL Draw(CDC *pDC,RECT *pRect);
```

Типы параметров должны быть вам знакомы. Первый – класс контекста дисплея и второй – прямоугольная область (где осуществлять прорисовку).

В качестве тела (в cpp файл) добавим прорисовку прямоугольника (пока):

```
pDC->FillSolidRect(0,0,100,100,0xFF0000); return TRUE;
```

Класс документа должен содержать структуру нашего объекта, коим является BMP рисунок. Так как поддержка BMP у нас возложена на MyBMP класс, то логично добавить в класс формата документа (…Doc) объект класса MyBMP:

```
MyBMP ozbmp; // в публичную секцию (не забудьте также про h файл)
```

Теперь у нас две задачи. В классе Doc осуществить загрузку информации, а в классе View отображать ее корректно. В этом основной принцип Documrnt/View модели. С загрузкой пока подождем, а прорисовку

добавим. Нам необходим класс …View. Там есть метод OnDraw (вы можете найти его напрямую или через ClassWizard).

Там уже есть код получения объекта документа – это тот объект, который описан в Doc классе. Добавим прорисовку:

```
pDoc->ozbmp.Draw(pDC,NULL);
```

Проверьте, теперь в каждом окне при запуске программы будет прорисован квадрат синего цвета 100 на 100.

4. Уточнение области прорисовки.

Теперь займемся определением области вывода. Проблема в том, что вывод может осуществляться как в окно, так и на внешнее растровое устройство, например принтер. Пути «прохождения» сообщений для прорисовки в этих случаях различны. Попробуем это решить.

Добавим в публичную секцию класса View прямоугольник:

```
RECT m_rectDraw;
```

В методе прорисовки класса View чуть поменяем код:

```
GetClientRect(&m_rectDraw);
pDoc->ozbmp.Draw(pDC, &m_rectDraw);
```

и, соответственно в методе прорисовки класса MyBMP:

```
pDC->FillSolidRect(pRect,0xFF0000);
```

Проверим. Теперь при смене габаритов окна прямоугольник всегда принимает нужный размер. Однако, если вы попытаетесь просмотреть как рисунок будет выглядеть на странице принтера (Menu->File->Print Preview), то увидите, что там рисуется лишь маленький прямоугольник и он не масштабируется на всю страницу. Займемся этим.

При печати на принтере (и при просмотре макета страницы) используется метод OnPrint, обрабатывающий соответствующее сообщение от Windows. У нас в классе View такого метода нет. Добавим его. Запустите ClassWizard, выбирите класс View, в списке Messages выберите OnPrint. Теперь нажмите кнопку AddFunction. Достаточно просто.

В VS.NET и VS2005 это следует делать так. Переходим в обзор Class View, выбираем наш класс, открываем его и там находим элемент Maps. Это карты класса MFC. Нам нужна карта сообщений MESSAGE (она там единственная). Выберите ее. Теперь вам надо перейти к реализации карты в исходном файле (участок кода в *.cpp файле-реализации класса между макросами BEGIN_MESSAGE_MAP и END_MESSAGE_MAP). Обычно это можно сделать двойным кликом на элементе MESSAGE. Ваша задача расположить курсор ввода внутри кода карты сообщений. Тогда в тулбаре окна Properties появится кнопка Messages. Нажмите на нее и выберите сообщение, обработчик которого надо добавить/изменить. Однако, если обработчик сообщения уже реализован в базовом классе, вам следует искать обработчик в списке методов базового класса, а не обработчиков сообщений. Для этого нажмите другую кнопку в тулбаре окна Properties. Она называется Overrides. В списке найдите нужный метод и в опциях выберите <Add> … В

нашем случае найдите метод OnPrint и выберите опцию <Add> OnPrint. В ваш код будут добавлен соответствующий метод.

До добавления нами функции, тем не менее, какая-то обработка была – мы ведь видели, что прямоугольник рисуется. В данном случае мы подменили функцию базового класса нашей собственной и должны либо вызвать функцию базового класса, либо написать свою. Последнее нам подходит, т.к. тело функции достаточно просто. Установим нужный размер поля вывода и вызовем функцию прорисовки:

```
m_rectDraw = pInfo->m_rectDraw;
OnDraw(pDC);
```

Вызов же базовой функции закоментируем.

Однако, в этом случае мы получаем m_rectDraw здесь, а не в функции прорисовки, поэтому и там необходимо внести изменения:

```
if(pDC->IsPrinting()){ pDoc->ozbmp.Draw(pDC, &m_rectDraw); }
else{ GetClientRect(&m_rectDraw); pDoc->ozbmp.Draw(pDC, &m_rectDraw); }
```

5. Загрузка и отображение BMP файлов.

Добавим функцию загрузки BMP файла в наш класс MyBMP:

```
BOOL ZBmp::Load(const char *szFileName);
```

параметр – имя файла для загрузки (с путем).

Для подгрузки и дальнейшего отображения BMP файла удобно поступать так:

- загрузить BMP файл с помощью LoadImage (вам уже знакомо);
- преобразовать хэндл в класс CBitmap с помощью метода CBitmap::FromHandle();
- запросить совместимый контекст дисплея CreateCompatibleDC();
- выбрать объект CBitmap в качестве активного SelectObject();
- использовать функцию StretchBlt для масштабирования CBitmap до нужных размеров вывода.

Кроме метода Load вам понадобиться изменить и метод Draw класса MyBMP. Сделайте это самостоятельно. Добейтесь эффекта автоматического масштабирования картинки под окно вывода.

Последнее, что необходимо – вызвать функцию подгрузки в нужном месте. Для этого существует метод Serialize. Нам нужна его часть, отвечающая за загрузку, а не запись. Добавим туда соответствующий код:

```
ozbmp.Load(ar.GetFile()->GetFilePath());
```

6. Добавление второго типа окна просмотра.

Document/View модель позволяет добавлять любое количество просмотрщиков (View) к любому документу (Doc). Добавим дополнительный View класс для отображения технической информации о BMP файле (размер по X и по Y).

Для этого легче всего скопировать cpp и h файлы уже существующего класса View, добавить их в проект и изменить там везде имя класса на новый, скажем View2. Того же эффекта можно добиться путем добавления класса с

помощью ClassWizard, однако это может потребовать больше усилий. Выберите любой вариант и добавьте класс самостоятельно. Добейтесь компиляции проекта без ошибок. Не забудьте, что, если вы копируете вручную, то вы должны заменять имя класса не только к коде C++, но и в спец. Макро и спец. Комментариях.

Теперь нам необходимо добавить еще одну константу в ресурсах. В строковых ресурсах скопируйте текущую строку с описание типа ресурса (IDR_…TYPE) под новым именем (IDR_…TYPE2). В тексте представлена спец. Последовательность, которая определяет тип документа, его расширение, его название и т.д. Подробную информацию о формате данной строки вы можете найти в Помощи по методу GetDocString класса CdocTemplate. Попробуйте варьировать значения строки, чтобы типы отображения были различны.

В основном файле проекта найдите место, где формируется шаблон данного Document/View взаимодействия и добавьте туда еще один шаблон для того же документа:

```
CMultiDocTemplate* pDocTemplate2;
pDocTemplate2 = new CMultiDocTemplate(
IDR_…TYPE2,
RUNTIME_CLASS(C…Doc),
RUNTIME_CLASS(CChildFrame), // custom MDI child frame
RUNTIME_CLASS(C…View2));
AddDocTemplate(pDocTemplate2);
```

здесь многоточие означает часть имени классов, зависящее от имени вашего проекта.

Теоретически теперь будут работать оба вьюера. Проверить это легко. При запуске программы она спросит у вас какой тип вьюера использовать для отображения файла.

В дальнейшем нам понадобится получать доступ к зарегистрированным шаблонам. Для этого следует шаблонные переменные сделать не локальными, а членами класса приложения. Перенесите их описание в класс C…App:

```
CMultiDocTemplate* pDocTemplate;
CMultiDocTemplate* pDocTemplate2;
```

Последний необязательный шаг – добавление иконки для нового вьюера. В ресурсах добавьте иконку с идентификатором, соответствующем нашему новому типу вьюера.

7. Отображение текстовой информации.

В новом классе-вьюере достаточно лишь сменить вызов функции в функции перерисовки с Draw на DrawText. Все остальное может оставаться в нем так же как и в первом вьюере (не зря же мы его писали).

Понятно, что необходимо добавить эту функцию в MyBMP класс:

```
BOOL DrawText(CDC *pDC, RECT *pRect);
```

Здесь вам понадобиться вывести текст (форматированный). Для этого вам могут понадобиться:

- CDC::GetTextMetrics()
- wsprintf()
- CDC::DrawText()

8. Последние шаги.

Запретим открытие пустого окна при старте и, напротив, будем открывать окно, если файл указан в командной строке. Найдите след. строку в главном файле проекта:

```
if (!ProcessShellCommand(cmdInfo)) return FALSE;
```

закомментируйте его и поместите туда другой код:

```
if (m_lpCmdLine[0] != '\0'){
// open an existing document
OpenDocumentFile(m_lpCmdLine);
}
// Enable drag/drop open
m_pMainWnd->DragAcceptFiles();
```

Попутно мы подключили поддержку Drag&Drop. Просто, верно?

Добавим поддержку в меню переключения с одного вьюера на другой. В редакторе ресурсов скопируйте меню под именем IDR_…TYPE и установите имя нового меню в IDR_…TYPE2. Как видите константа меню соответствует типу вьюера (см. параграф 6).

Теперь в обоих этих меню добавьте по два элемента:
Menu->View->Show Picture с константой ID_VIEW_PIC и
Menu->View->Show Info с константой ID_VIEW_TEXT

Теперь надо добавить функции – реакции на новые элементы меню.

В ClassWizard выберите класс CMainFrame, в поле Object IDs найдите ID_VIEW_PIC, в поле Messages выберите COMMAND, нажмите Add Function и добавьте функцию OnViewPic. Аналогичные действия проведите для второго элемента и добавьте функцию OnViewText.

Теперь несколько «нечеткая» последовательность.

Заполните ваши обработчики новых команд меню так:

```
void CMainFrame::OnViewPic(){
CreateOrActivateFrame(theApp.pDocTemplate,RUNTIME_CLASS(CTest2View)
);
}
void CMainFrame::OnViewText(){
CreateOrActivateFrame(theApp.pDocTemplate2,RUNTIME_CLASS(CTextBVie
w));
}
```

Как видите – они только вызывают некую функцию, которую также необходимо вручную добавить в класс CMainFrame:

```
void CMainFrame::CreateOrActivateFrame(CDocTemplate* pTemplate,
```

```
CRuntimeClass* pViewClass)
{
    CMDIChildWnd* pMDIActive = MDIGetActive();
    ASSERT(pMDIActive != NULL);
    CDocument* pDoc = pMDIActive->GetActiveDocument(); ASSERT(pDoc != NULL);
    CView* pView;
    POSITION pos = pDoc->GetFirstViewPosition();
    while (pos){
        pView = pDoc->GetNextView(pos);
        if (pView->IsKindOf(pViewClass))
        {
            pView->GetParentFrame()->ActivateFrame();
            return;
        }
    }
    CMDIChildWnd* pNewFrame=(CMDIChildWnd*)(pTemplate->CreateNewFrame(pDoc, NULL));
    if (pNewFrame == NULL)  return;     // not created
    ASSERT(pNewFrame->IsKindOf(RUNTIME_CLASS(CMDIChildWnd)));
    pTemplate->InitialUpdateFrame(pNewFrame, pDoc);
}
```

Если вы разберете текст функции, то увидите, что основное ее назначение – избежать повторного открытия окна для файла, который уже открыт в одном из окон. Разберитесь с этим самостоятельно.

И последний штрих. Как видите здесь используется переменная theApp, однако она не видна от сюда. Сделаем ее видимой – добавим ее описание (она уже создается в главном файле проекта) прямо за описанием класса – в h файл:

extern C…App theApp;