МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

В.Н. Блохин, Ю.А. Лабода, А.Г. Зыков

СООБЩЕНИЯ И ПРИЛОЖЕНИЯ WINDOWS

Учебное пособие

Санкт-Петербург

2012

В.Н. Блохин, Ю.А. Лабода, А.Г. Зыков

СООБЩЕНИЯ И ПРИЛОЖЕНИЯ WINDOWS

Учебное пособие

Санкт-Петербург

2012

1. Основные сообщения Windows

В данном пособии дан краткий обзор наиболее часто используемых сообщений Windows, которые могут обрабатываться или посылаться при выполнении приложений Windows. Все сообщения приведены в алфавитном порядке. Более подробную информацию можно получить в MSDN. Предполагается, что студенты знакомы с основными понятиями Windows: дескриптор окна (HWND), дескриптор контекста устройства (HDC), параметрами сообщений Windows WPARAM и LPARAM, параметрами окон Windows и стандартными органами управления Windows.

1.1. Назначение и обработка сообщений в Win32 application

Сообщения Windows являются эффективным способом связи между операционной системой (ОС) и приложением пользователя, а также между окнами приложения. Именно с их помощью ОС информирует соответствующее окно обо всех касающихся его событиях и осуществляется запрос или передача информации конкретному окну.
Все сообщения имеют единый формат.

Передача сообщения окну производится путем вызова соответствующей оконной процедуры, имя которой указывается при регистрации класса окна. При этом сообщения могут приходить окну либо через очередь сообщений приложения, либо минуя ее. Для получения сообщения из очереди необходимо в главной функции приложения (ГФП) организовать соответствующий цикл обработки. Пример такого простейшего цикла иллюстрирует следующий фрагмент ГФП:

```
MSG msg;
while (GetMessage(&msg, NULL, 0, 0))
{
  TranslateMessage(&msg);
  DispatchMessage(&msg);
}
return (msg.wParam); // Завершение ГФП
```

Здесь:
msg – структура, в которой будет храниться информация о полученном из очереди сообщении;
GetMessage – функция, выбирающая сообщение из очереди и заносящая информацию о нем в msg;
Эта функция вернет 0, когда получит сообщение WM_QUIT, при ошибке она возвращает -1;

TranslateMessage – функция, используемая при обработке сообщений от клавиатуры – при нажатии отображаемого символа будет сгенерировано со-

общение WM_CHAR(или WM_SYSCHAR для системной клавиши) и виртуальный код символа будет переведен в обычный, что в некоторых случаях существенно облегчает его обработку;

Функция DispatchMessage вызывает соответствующую оконную процедуру для дальнейшей обработки сообщения.

В некоторых случаях (например при обработке таблицы акселераторов) данный цикл может быть модифицирован.

Очевидно, что этот цикл завершится при выборе из очереди сообщения WM_QUIT. Отправка этого сообщения в очередь производится функцией PostQuitMessage.

Приложение также может послать сообщение какому либо окну, используя функцию PostMessage для постановки его в очередь или SendMessage для передачи его напрямую (минуя очередь сообщений). В первом случае сообщение обработается только после того, как ОС получит управление и сообщение будет выбрано из очереди, а во втором – сразу. Кроме того при использовании SendMessage обеспечивается возможность получения результата обработки данного сообщения соответствующей оконной процедурой, что очень важно при направлении сообщения стандартным органам управления (ОУ) (при этом для удобства может использоваться функция SendDlgItemMessage как подвид SendMessage).

Все сообщения можно разделить на две большие группы: общего назначения (приходят окнам различных классов и имеют префикс WM_) и специализированные (используются ОУ и имеют префикс в зависимости от типа ОУ).

1.2.Основные сообщения общего назначения

Это самая большая и наиболее часто используемая группа сообщений. Без их обработки не обходится ни одно приложение. Сообщения могут относиться как к клиентской области (КО) окна (имеют префикс WM_), так и к не клиентской области (НКО) (префикс WM_NC).

Краткий обзор основных сообщений общего назначения приведен в таблице 1.

Таблица 1.

Сообщение	**Назначение**	**WPARAM**	**LPARAM**	**Возвращаемое значение**	**Примечание**
WM_ ACTIVATE	Приходит при активизации или деактиви-	LOWORD содержит флаг активизации, а HIWORD – флаг мнии-	Дескриптор окна, которое было активным или неак-	При стандартной обработке приложение	Нестандартная обработка при перекрывающихся ок-

	зации окна.	мизации.	тивным до этого.	дол-жно вернуть 0, иначе 1.	нах край-не не же-лательна.
WM_ ACTVATEAPP	Анало-гично для окна другого прило-жения.	Содержит флаг акти-визации.	Идент-ификатор потока, чье окно было ак-тивным или неак-тивным.	Прило-же-ние дол-жно вернуть 0.	
WM_CHAR	Генери-руется функци-ей Translate Message для ото-бражае-мых симво-лов при нажатии соответ-ствую-щих кла-виш.	Код клави-ши.	Дополни-тельная информа-ция.	0	
WM_ CHARTOITEM	Посыла-ет окно-список, имею-щее стиль LBS_W ANTKE YBOAR DI NPUT роди-тельско-му окну при об-работке WM_CH AR.	LOWORD – код симво-ла. HIWORD – позиция курсора.	Дескрипт-ор окна-списка.	-1 или -2 означа-ет, что прило-жение управля-ет всеми аспекта-ми вы-бранно-го эле-мента и не тре-бует дальней-ших дей-ствий. 0 или	Позволяет организо-вать работу приложени при нажатии ка-кой-либо клавиши в окне-списке

				больше – индекс элемента, для которого окно-список должно выполнить стандартные действия для нажатой клавиши в данном элементе списка.	
WM_CLEAR	Приложение посылает это сообщение для очистки отметки окна редактора или окна COMBO BOX.	Не используется (должно быть 0).	Не используется (должно быть 0).	0	
WM_CLOSE	Приходит при закрытии окна через “крест”.	Не используется.	Не исользуется.	0	Стандартная функция обработки вызывает DestroyWindow.
WM_ COMMAND	Три источника этого сообщения: пункт меню,	Для органа управления – дескриптор окна. Иначе – 0.	0		

	акселе-ратор, орган упр-я.				
WM_COPY	Прило-жение посыла-ет это сообще-ние ре-дактору или ком-бинирo-ванному списку для ко-пирова-ния от-мечен-ного участка текста в Clipboar d .	0	0	Отсут-ствует.	Если со-общение посылает-ся комби-нирован-ному спи-ску, то оно при-ходит его элементу-редактору и не имеет эффекта, ели вы-ставлен флаг CBS_DRO PDOWNL IST.
WM_CREATE	При соз-дании окна (до его ото-бражe-ния).	Не исполь-зуется.	Содержит адрес структуры с характе-ристиками создавае-мого окна.	Для про-долже-ния соз-дания окна на-до вер-нуть 0, иначе 1.	
WM_ CTLCOLOR BTN	Прихо-дит ро-дитель-скому окну от кнопки при ее перери-совке.	HDC кноп-ки.	HWND кнопки.	Деск-рип-тор кисти фона кнопки.	Позволяет менять стандарт-ные цвета кнопки. Для PUSHBUT TON цвет тек-ста изме-нить

					нель-зя(будет изменять-ся цвет фо-кусной рамки).
WM_ CTLCOLORDLG	Анало-гично для диа-лога.	HDC диа-лога.	HWND диалога.	Деск-рип-тор кисти фона диалога.	Позволяет менять стандарт-ные цвета диалога
WM_ CTLCOLOREDIT	Анало-гично для ре-дактора	HDC редак-тора	HWND редактора	Деск-рип-тор кисти фона ре-дактора	Позволяет менять стандарт-ные цвета редактора
WM_CTLCOLORLIST BOX	Анало-гично для спи-ска	HDC спи-ска	HWND списка	Деск-риптор кисти фона списка	Позволяет менять стандарт-ные цвета списка
WM_CTLCOLOR SCROLLBAR	Анало-гично для по-лосы про-крутки	HDC поло-сы про-крутки	HWND полосы прокрутки	Деск-риптор кисти полосы про-крутки	Позволяет менять цвета по-лосы про-крутки
WM_CTLCOLORSTATIC	Анало-гично для окна static	HDC окна static	HWND окна static	Деск-риптор кисти фона ок-на static	Позволяет менять стандарт-ные цвета окна static
WM_CUT	Удаляет текущую отметку в редак-торе или комби-ниро-	0	0	0	

	ванном списке (для поля редактора) и копирует удаленный текст в clipboard				
WM_ DELETEITEM	Приходит родительскому окну от окна-редактора или комбинированного списка при их разрушении, очистке или удаления строки.	Идентификатор ОУ.	Адрес структуры типа DELETEITEMSTRUCT, содержащей дополнительную информацию.	TRUE	
WM_ DESTROY	При разрушении окна	0	0	0	
WM_ DRAWITEM	При собственной отрисовке кнопки, списка, комбинированного. списка или	Идентификатор органа управления (для пункта меню – 0).	Адрес структуры, содержащей информацию об соотв. ОУ или пункте меню	. TRUE	Должен быть предусмотрен Соответствующий флаг.

	пункта меню.				
WM_ENABLE	При изменении состояния окна	Определяет состояния окна	0	0	
WM_ ENDSESSION	Информирует о завершении сеанса Windows и приходит после обработке результата сообщения WM_QUERYEND SESSION	Флаг завершения сеанса Windows, совпадает с возвращаемым значением обработки WM_QUERY ENDSESSION)	Причина завершения сеанса Windows	0.	Смотри описание WM_QUERY ENDSESSION. Если wParam равен 1, то необходимо завершить все действия, так как после возврата приложение будет завершено. При этом не надо вызывать функции DestroyWindow и PostQuitMessage.
WM_ ERASEBKGND	Сигнализирует о разрушении фона рабочей области окна. Посылается при необходимости функци-	Дескриптор контекста устройства.	0	Если приложение само отрисовывает фон – 1, иначе 0.	Именно при обработке данного сообщения следует перерисовывать фон окна, если он отличается от указанно-

	ей BeginPaint.				го при регистрации класса.
WM_ GETFONT	Позволяет получить дескриптор фонта, используемого органом управления.	0	0	Если возвращаемое значение – NULL, то орган управления использует системный фонт.	
WM_GETICON	Позволяет получить дескриптор иконы окна.	Тип иконы (большая или маленькая).	0	Дескриптор соответствующей иконы.	
WM_GETMINMAXINFO	Приходит при изменении размера или позиции окна (в том числе при его создании).	0	Адрес структуры MINMAXINFO	0	Позволяет установить минимальный и максималь-ный размер окна, указывая требуемые значения в соответствующих полях структуры MINMAXINFO.
WM_ GETTEXT	Позволяет скопировать связан-	Количество максимально копируемых символов.	Адрес строки для копии текста.	Количество скопированных	

	ный с окном текст в буфер.			симво-лов.	
WM_ GETTEXT	Определение длины текста, связанного с окном.	0	0	Длина текста без 0-символа.	
WM_HELP	Служит для организации справки по работе сданным окном. Приходит окну, связанному с активным пунктом меню либо имеющему в данный момент времени фокус ввода, иначе – активному окну при нажатии клавиши F1.	0	Адрес структуры HELPINFO	TRUE	
WM_HSCROL	Горизонтальная про-	LOWORD – параметр прокрутки.	Дескриптор окна (ОУ) про-	0	

	крутка	HIWORD – позиция прокрутки при использовании SB_THUMBPOSITION или SB_THUMBTRACK в LOWORD.	крутки либо NULL, если приходит от полосы прокрутки.		
WM_ICONERASEBKGND	Приходит минимизированному окну, если икона определена при регистрации класса окна, и позволяет заполнить фон до отрисовки иконы	HDC иконы.	0	0	
WM_ INITDIALOG	Приходит окну диалога при его инициализации.	Дескриптор окна, которое получит фокус ввода по умолчанию.	Параметр инициализации – содержит дополнительную информацию	Если TRUE, то фокус ввода – по умолчанию; иначе пользователь устанавливает его сам с помо-	Если возвращаемое значение – FALSE и функция SetFocus не используется, то ни один из ОУ не будет иметь фокуса

				щью функции SetFocus.	ввода.
WM_ INITMENU	Приходит при инициализации меню.	Дескриптор меню.	0	0	Позволяет модифицировать меню до его отображения.
WM_INITPOPUPMENU	Аналогично для подменю или выпадающего меню.	Дескриптор подменю.	LOWORD – позиция подменю (выпадающего меню). HIWORD – флаг меню.	0	Позволяет модифицировать меню до его отображения.
WM_ KEYDOWN	Приходит при нажатии клавиши.	Виртуальный код клавиши.	Дополнительная информация.	0	
WM_KEYUP	Приходит при отпускании клавиши.	Виртуальный код клавиши.	Дополнительная информация	0	
WM_ KILLFOCUS	Окно потеряло фокус ввода.	Дескриптор окна, получившего фокус ввода (может быть 0).	0	0	
WM_LBUTTON DBLCLK	Приходит при двойном нажатии левой клавиши “мыши” в рабочей области окна.	Флаг нажатых виртуальных клавиш.	LOWORD – x, а HIWORD – y координата курсора “мыши”.	0	Это сообщение получают окна, у которых указан стиль класса CS_DBLCLKS.

WM_ LBUTTONDOWN	Приходит при нажатии левой клавиши “мыши” в рабочей области окна	Флаг нажатых виртуальных клавиш.	LOWORD – x, а HIWORD – y координата курсора “мыши”.	0	
WM_ LBUTTONUP	Приходит при отпускании левой клавиши “мыши” в рабочей области окна	Флаг нажатых виртуальных клавиш.	LOWORD – x, а HIWORD – y координата курсора “мыши”.	0	
WM_MBUTTON DBLCLK	Приходит при двойном нажатии средней клавиши “мыши” в рабочей области окна.	Флаг нажатых виртуальных клавиш	LOWORD – x, а HIWORD – y координата курсора “мыши”.	0	Это сообщение получают окна, у которых указан стиль класса CS_DBLCLKS.
WM_ MBUTTONDOWN	Приходит при нажатии средней клавиши “мыши” в рабочей области окна	Флаг нажатых виртуальных клавиш	LOWORD – x, а HIWORD – y координата курсора “мыши”.	0	
WM_ MBUTTONUP	Приходит при отпускании	Флаг нажатых виртуальных клавиш	LOWORD – x, а HIWORD – y коор-	0	

	средней клавиши “мыши” в рабочей области окна		дината курсора “мыши”.		
WM_ MENUCHAR	Приходит окну с активным меню, когда пользователь нажал клавишу не связанную с акселератором или с мнемоническим кодом.	LOWORD – код клавиши, HIWORD – флаг меню. .	Дескриптор меню	Определяет желаемое действие системы	Необходимо обрабатывать это сообщение, если акселератор используется для пункта меню, который отображается как BITMAP.
WM_ MENUSELECT	Приходит окну при выборе пункта меню.	LOWORD – идентификатор пункта меню, HIWORD – флаг меню .	Дескриптор меню.	0	
WM_ MOUSEMOVE	Приходит окну при перемещении курсора мыши в рабочей области окна.	Флаг, определяющий нажатые виртуальные клавиши.	LOWORD – горизонтальная координата курсора, HIWORD – вертикальная.	0	
	Прихо-	0	LOWORD	0	

WM_MOVE	дит окну после его перемещения..		– горизонтальная координата верхнего левого угла рабочей области, HIWORD – вертикальная.		
WM_MOVING	Приходит окну при его перемещении..	0	Аналогично.		
WM_ NCACTIVATE	Приходит окну когда его не клиентская область нуждается в изменении для индикации активного или неактивного состояния.	Новое состояние заголовка окна или иконы.	0	Если WPARAM – FALSE, возврат TRUE означает действие по умолчанию, а FALSE предотвращает деактивизацию заголовка окна или иконы. Иначе безразлично.	
WM_	Приходит окну	0	Адрес структуры	TRUE для про-	

NCCREATE	перед WM_CREATE		CREATESTRUCT	должения создания окна, иначе – FALSE.	
WM_ NCDESTROY	Информирует о разрушении НКО и приходит после WM_DESTROY.	0	0	0	
WM_ NCHITTEST	Старшее сообщение от мыши.	0	LOWORD и HIWORD содержит X и Y координаты курсора мыши в экранной системе координат.	Детализация места нахождения курсора мыши.	
WM_NCLBUTTONDBLCLK	Двойное нажатие левой клавиши мыши в НКО.	Специфицирует значение, полученное при обработке WM_NCHITTEST.	Структура типа POINT, содержащая координаты курсора мыши в экранной системе координат.	0	Должен быть предусмотрен стиль CS_DBLCLKS при регистрации класса окна.
WM_NCLBUTTONDOWN	Нажатие левой клавиши мыши в НКО.	Аналогично.	Аналогично.	0	

WM_ NCLBUTTON UP	Отпус-кание левой клавиши мыши в НКО.	Аналогич-но.	Анало-гично.	0	
WM_NCMBUT TONDBLCLK	Двойное нажатие средней клавиши мыши в НКО.	Аналогич-но.	Анало-гично.	0	Должен быть стиль ок-на CS_DBLC LKS.
WM_NCMBUT TONDOWN	Нажатие средней клавиши мыши в НКО..	Аналогич-но.	Анало-гично.	0	
WM_ NCMBUTTON UP	Отпус-кание средней клавиши мыши в НКО.	Аналогич-но.	Анало-гично.	0	
WM_NCMOUS EMOVE	При пе-ремеще-нии кур-сора мыши в НКО.	Аналогич-но.	Анало-гично.	0	
WM_NCPAINT	Прихо-дит, если НКО нужда-ется в перери-совке.	Дескриптор региона пе-рерисовки.	0		Контекст устройст-ва может быть по-лучен че-рез `GetDCEx` .
WM_NCRBUT TON DBLCLK	Двойное нажатие правой клавиши мыши в НКО.	Аналогично левой.	Анало-гично ле-вой.	0	Должен быть пре-дусмотрен стиль CS_DBLC LKS при регистра-ции клас-

					са окна.
WM_NCRBUTTONDOWN	Нажатие правой клавиши мыши в НКО.	Аналогично левой.	Аналогично левой.	0	
WM_ NCRBUTTONUP	Отпускание правой клавиши мыши в НКО.	Аналогично левой.	Аналогично левой.	0	
WM_ NEXTDLGCTL	Посылается диалоговому окну для установки фокуса ввода на ОУ.	Зависит от LPARAM: если TRUE, то идентифицирует ОУ. получившего фокус ввода, если FALSE – флаг индикации следующего или предыдущего ОУ, получившего фокус ввода. Если этот параметр 0 – фокус ввода получает следующий ОУ, иначе – предыдущий.	Флаг, определяющий трактовку WPARAM.	0	В отличие от SetFocus модифицирует рамку ОУ.
WM_PAINT	КО окна нуждается в перерисовке.	Контекст устройства	0	0	Необходимо вызывать функцию BeginPaint до отрисовки и EndPaint

					после. Если данное сообщение – результат вызова RedrawWindow, то необходимо проверить наличие региона перерисовки и вызывать BeginPaint и EndPaint только, если он не пустой.
WM_QUERYENDSESSION	Приходит при завершении сеанса WINDOWS до WM_ENDSESSION	Зарезервирован	Идентифицирует причину завершения сеанса.	0 – запретить завершение, 1 (по умолчанию)– разрешить.	После обработки этого сообщения будет послано сообщение WM_ENDSESSION.
WM_QUIT	Ставится в очередь функцией PostQuitMessage.	Код возврата, указанный PostQuitMessage.	0	Нет, так как не обрабатывается оконной процедурой.	При получении этого сообщения GetMessage прерывает обработку цикла сообщений из очереди приложения.
WM_RBUTTO	Двойное нажатие	Флаг нажатых вирту-	LOWORD – x, a	0	Это сообщение по-

N DBLCLK	правой клавиши мыши в КО.	альных клавиш.	HIWORD – у координата “мыши”.		лучают окна, у которых стиль CS_DBLCLKS.
WM_RBUTTON DOWN	Нажатие правой клавиши мыши в КО.	Флаг нажатых виртуальных клавиш.	LOWORD – x, а HIWORD – у координата курсора “мыши”.		
WM_ RBUTTONUP	Отпускание правой клавиши мыши в КО.	Флаг нажатых виртуальных клавиш.	LOWORD – x, а HIWORD – у координата курсора “мыши”.		
WM_ SETFOCUS	Окно получило фокус ввода.	Дескриптор окна, потерявшего фокус ввода.	0	0	
WM_SETFONT	Позволяет установить фонт для ОУ.	Дескриптор фонта.	TRUE указывает перерисовку ОУ.	0	
WM_SETICON	Связывает с окном новую икону.	Тип иконы (большая или маленькая).	Дескриптор иконы.	Дескриптор предыдущей иконы.	
WM_ SETREDRAW	Разрешает или запрещает перерисовку окна.	Флаг перерисовки: 1 – разрешена, 0 – запрещена.	0	0	
WM_SETTEXT	Установка текста окна.	0	Адрес строки текста.	TRUE – текст установлен.	В зависимости от типа окна текст вы-

					водится либо в заголовке либо внутри окна.
WM_ SHOW WINDOW	Приходит окну при его отображении или скрытии.	Флаг отображения.	Статус отображаемого окна.	0	
WM_SIZE	Приходит окну после изменения его размера.	Флаг изменения размера.	LOWORD и HIWORD содержат ширину и высоту КО.	0	
WM_SIZING	Приходит окну при изменении его размера	Определяет границу, по которой шло изменение.	Адрес структуры RECT, определяющей прямоугольник перемещения в экранных координатах.	TRUE	
WM_SYSCHAR	Сигнализирует о нажатии системной клавиши после ее трансляции.	Код символа.	Дополнительная информация.	0	Приходит после WM_SYS KEYDOWN, если виртуальный код может быть преобразован в обычный функцией

					Translate Message
WM_ SYSKEYDOWN	Сигнализирует о нажатии системной клавиши.	Виртуальный код клавиши.	Дополнительная информация	0	
WM_ SYSKEYUP	Сигнализирует об отпускании системной клавиши.	Виртуальный код клавиши.	Дополнительная информация	0	
WM_TIMER	Прерывание от таймера.	Идентификатор таймера.	Адрес функции обработки, указанную при инсталляции таймера функцией SetTimer.	0	Низко приоритетное сообщение.
WM_ VSCROLL	Вертикальная прокрутка.	Смотри WM_HSCROLL	Смотри WM_HSCROLL	Смотри WM_HSCROLL	
WM_WINDOWPOSCHANGED	Приходит окну после изменения его позиции.	0	Адрес структуры WINDOWPOS	0	

1.3. Специализированные сообщения

Как указывалось выше, ряд сообщений обрабатываются только ОУ конкретного класса.

В данном разделе рассматриваются сообщения и нотификации списков, редакторов, кнопок.
Под нотификацией понимается информация, передаваемая ОУ родительскому окну через сообщение WM_COMMAND (в этом случае значения WPARAM и LPARAM указываются для WM_COMMAND).

Все сообщения для стандартных ОУ обычно посылаются соответствующему окну функцией SendDlgItemMessage.

1.3.1. Сообщения окна-списка (класс Listbox)

В таблице 2 приведены наиболее часто используемые сообщения и нотификации для окна-списка.
Сообщения окна-списка имеют префикс LB_, а нотификации – префикс LBN_.
Смотрите также сообщения WM_DELETEITEM и WM_CTLCOLORLISTBOX, описанные в таблице 1.

Таблица 2.

Сообщение (нотификация)	**Назначение**	**WPARAM**	**LPARAM**	**Возвращаемое значение**	**Примеча-ние**
LB_ ADDSTRING	Добавить строку в список	0	Адрес добавляемой стоки.	Индекс (с 0) добавленной строки или LB_ERR в случае ошибки или LB_ERRSPACE при недостатке памяти.	Если список создан с LBS_SORT стилем без LBS_HASSTRINGS, система отправит родительскому окну сообщение WM_COMPAREITEM один или несколько раз

					для корректного определения места строки в списке.
LB_DELETESTRING	Удалить строку из списка.	Индекс (с 0)	0	Количество оставшихся в списке строк или LB_ERR, если индекс больше числа элементов списка.	Если это сообщение посылается окну, имеющему стиль owner-drawn без LBS_HASSTRINGS, система отправит родительскому окну сообщение WM_DELETEITEM.
LB_DIR	Заполнить список именами файлов.	Атрибуты файлов.	Адрес строки, определяющей маску добавляемых имен файлов (если “*.*”, то – все файлы).	Индекс (с 0) строки с последним именем файла или LB_ERR в случае ошибки или LB_ERRSPACE при недостатке памяти.	
	Найти строку в списке	Индекс (с 0) перед первым	Префикс для поиска без	Индекс (с 0) найденной строки или	

LB_ FINDSTRING	по заданному префиксу.	элементом, с которого начинается поиск. Если он равен -1, то поиск осуществляется по всему списку.	учета регистра.	LB_ERR в случае ошибки.	
LB_ GETCOUNT	Получить количество элементов списка.	0	0	Количество элементов списка или LB_ERR в случае ошибки.	
LB_ GETCURSEL	Получить индекс отмеченного в списке элемента.	0	0	Индекс (с 0) отмеченного в списке элемента или LB_ERR в случае ошибки..	Для определения индексов в списке с множественным выбором используйте LB_GETSELITEMS.
LB_ GETTEXT	Получить строку по заданному индексу.	Индекс элемента (с 0), из которого хотим получить строку.	Адрес строки результата.	Длина полученной строки или LB_ERR в случае ошибки.	
LB_ GETTEXTLEN	Определить длину строки по заданному индексу.	Индекс требуемого элемента (с 0).	0	Длина строки.	
LB_ GETTOP	Получить индекс	0	0	Индекс первого видимого элемента в	

INDEX	первого видимого элемента в списке.			списке	
LB_ INSERT STRIG	Вставить строку в список.	Индекс элемента (с 0), определяющий позицию , куда будет добавлена строка (если равен -1, то в конец списка).	Адрес добавляемой стоки.	Индекс вставляемой строки или LB_ERR в случае ошибки или LB_ERRSPACE при недостатке памяти.	
LB_ SETCURSEL	Выделяет указанный элемент списка.	Индекс элемента (с 0), определяющий позицию отмечаемой строки (если -1, то список не будет иметь выбора).	0	При ошибке или если индекс равен минус 1 возвращается LB_ERR.	Используется только для списков с одиночным выбором.
LB_SETSEL	Изменить статус выбора строки в списке с множественным выбором.	Флаг состояния строки (если TRUE – строка подсвечивается, если FALSE – подсветка убирается).	Индекс элемента (с 0), для которой изменяется состояние выбора, если он равен -1, то действие осуществляется для всего списка.	При ошибке возвращается LB_ERR.	Только для списка с множественным выбором
LBN_	Нотификация - извеща-	Смотрите описание WM_COM	Смотрите описание WM_CO	Смотрите описание WM_COMM	Только если список

DBLCLK	ет родительское окно о двойном щелчке мыши на строку списка.	MAND	MMAND	AND	имеет стиль LBS_NOTIFY.
LBN_KILLFOCUS	Нотификация - извещает родительское окно о потере фокуса окном-списком.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND		
LBN_SELCHANGE	Нотификация - извещает родительское окно об изменении выбора в списке.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для стиля LBS_NOTIFY и не посылается при использовании LB_SETCURSEL. Для списка с множественным выбором приходит при любом нажатии стрелок даже если состояние выбора

					не меняется.
LBN_ SETFOCUS	Нотификация - извещает родительское окно о получении фокуса оконом-списком.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	

1.3.2. Сообщения окна-редактора (класс Edit)

В отличие от окна-списка, работа с которым невозможна без использования некоторых сообщений, редактор является более самодостаточным ОУ и многие приложения используют его без использования специальных сообщений. Тем не менее авторы сочли нужным привести в таблице 3 наиболее часто используемые сообщения и нотификации для окна-редактора.
Сообщения окна-редактора имеют префикс EM_, а нотификации – префикс EN_.

Смотрите также сообщения WM_COPY и WM_CTLCOLOREDIT, описанные в таблице 1.

Таблица 3.

Сообщение (нотификация)	**Назначение**	**WPARAM**	**LPARAM**	**Возвращаемое значение**	**Примеча-ние**
EM_GETFIRSTVISIBLELINE	Позволяет определить, какая строка	0	0	Номер первой видимой строки (с 0) .	Только для многострочного ре-

	редактора видна первой.				дактора.
EM_GET LIMITTEXT	Определить текущий лимит для текста в окне-редакторе.	0	0	Максимальный размер текста.	
EM_ GETLINE	Позволяет скопировать указанную строку в буфер.	Номер копируемой строки (с 0).	Адрес строки результата. Первое слово буфера содержит максимальное значение копируемых символов.	Длина скопированной строки или 0, если номер строки больше числа строк редактора..	Только для многострочного редактора.
EM_ GETLINECO	Определить количество строк редактора.	0	0	Количество строк текста в редакторе или 1, если текста нет.	Только для многострочного редактора.
EM_GETSEL	Получить границы выделенного участка текста..	Стартовая позиция (может быть 0)..	Конечная позиция (может быть 0).	32-х разрядное число: LOWORD – стартовая позиция выделенного текста, HIWORD – конечная.	
EM_SCROLL	Прокрутить текст в окне.	Параметр прокрутки.	0	Если успешно, то HIWORD – TRUE, а LOWORD – количество прокручен-	Это сообщение эквивалентно WM_SCROLL.

				ных строк (может не совпадать с заданным), иначе – FALSE.	
EM_SETREADONLY	Позволяет установить или отменить режим “только для чтения”.	Режим : TRUE – установить, FALSE – отменить.	0	Не 0, если успешно, иначе -0.	При установленном режиме“только для чтения” текст в окне нельзя изменять.
EM_ SETRECT	Используется для задания форматированного прямоугольника в рабочей области редактора.	0	Адрес структуры типа RECT, содержащей размеры нового прямоугольника для вывода текста.	Отсутствует	Только для многострочного редактора. Первоначально размер прямоугольника равен размеру КО редактора и может быть изменен и в меньшую и в большую сторону.
EM_SETSEL	Изменить в окне статус от-	Номер стартовой позиции текста	Номер конечной позиции	Отсутствует	Если стартовая по-

	метки текста.	для установки отметки.	текста для установки отметки.		зиция – 0, а конечная – минус 1,то выделяется весь текст, если стартовая позиция – минус 1,то отметка снимается.
EM_UNDO	Отменить последнее действие.	0	0	Для однострочного редактора всегда TRUE, а для многострочного – TRUE, если успешно и FALSE иначе.	
EN_CHANGE	Нотификация:-извещает родительское окно об изменении текста в редакторе.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	
EN_ HSCROLL	Нотификация:-извещает родительское окно о нажатии на горизонтальную поло-	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	

	су прокрутки.				
EN_ KILLFOCUS	Нотификация:- извещает родительское окно о потере редактором фокуса ввода.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	
EN_ SETFOCUS	Нотификация:- извещает родительское окно о получении редактором фокуса ввода.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	
EN_ VSCROLL	Нотификация:- извещает родительское окно о нажатии на вертикальную полосу прокрутки.	Смотрите описание WM_COMMAND.	Дескриптор окна-редактора.	Смотрите описание WM_COMMAND	

1.3.3. Сообщения комбинированного списка (класс Combobox)

Основные сообщения для этого ОУ приведены в таблице 4.
Поскольку комбинированный список содержит элемент-редактор и элемент-список, то его
сообщения во многом повторяют сообщения для этих ОУ, но имеют префикс CB_.
Смотрите также сообщение WM_DELETEITEM , описанное в таблице 1.

Таблица 4.

Сообщение (нотификация)	**Назначение**	**WPARAM**	**LPARAM**	**Возвращаемое значение**	**Примеча-ние**
CB_ ADDSTRING	Добавить строку в список.	0	Адрес добавляемой строки.	Индекс (с 0) добавленной в список строки, если успешно, иначе - CB_ERR, или CB_ERRSPACE.	
CB_ DELETESTRING	Удалить строку из списка .	Индекс удаляемой строки (с 0).	0	Количество оставшихся строк, или CB_ERR при ошибке.	Если ОУ создан со стилем owner-drawn , но не имеет стиля CBS_HASSTRINGS, то родительское окно получит сообщение WM_DELETEITEM.
CB_DIR	Заполнить список именами файлов.	Атрибуты файлов.	Адрес строки, определяющей маску добавлямых имен файлов (“*.*” - все фай-	Индекс (с 0) строки с последним именем файла или CB_ERR в случае ошибки или CB_ERRSPACE при не-	

			лы).	достатке памяти.	
CB_ FINDSTRING	Найти строку в списке по заданному префиксу.	Индекс (с 0) перед первым элементом, с которого начинается поиск. Если он равен -1, то поиск осуществляется по всему списку.	Префикс для поиска без учета регистра.	Индекс (с 0) найденной строки или CB_ERR в случае ошибки.	
CB_ GETCOUNT	Получить количество элементов списка.	0	0	Количество элементов списка или CB_ERR в случае ошибки.	
CB_ GETCURSEL	Получить индекс отмеченного в списке элемента.	0	0	Индекс (с 0) отмеченного в списке элемента или CB_ERR в случае ошибки.	.
CB_ GETEDIT SEL	Получить стартовую и конечную позицию символов отмеченного в редакторе участка текста.	Адрес 32-х разрядного слова для получения стартовой позиции. Может быть NULL.	Адрес 32-х разрядного слова для получения конечной позиции. Может быть NULL.	32-х разрядное слово, содержащее стартовую и конечную позиции символов (в LOWORD и HIWORD соответственно).	
CB_	Получить	Индекс (с 0) получае-	Адрес строки ре-	Длина полученной стро-	Если ОУ -

GETLBTEXT	строку из элемента списка.	мой строки.	зультата.	ки или CB_ERR в случае ошибки.	OWNERDRAWN, без стиля CBS_HASSTRINGS, то по адресу указанному в LPARAM будет 32-х разрядное число, связанное с элементом списка (данные элемента).
CB_GETLBTEXTLEN	Получить длину строки для элемента списка.	Индекс (с 0) интересующей строки.	0	Длина строки.	
CB_INSERTSTRING	Вставить строку в список по заданному индексу.	Индекс (с 0) для вставляемой строки.	Адрес вставляемой строки.	Индекс (с 0) для вставленной строки или CB_ERR в случае ошибки или CB_ERRSPACE при недостатке памяти.	
CB_RESETCONTENT	Очищает список.	0	0	CB_OKAY.	Если ОУ создан со стилем owner-

					drawn , но не имеет стиля CBS_HASSTRINGS, то родительское окно получит сообщение WM_DELETEITEM для каждого элемента списка..
CB_SELECTSTRING	Выделяет соответствующий (найденный) элемент списка и копирует его в строку редактора.	Стартовая позиция для поиска (с 0), если это -1, то поиск по всему списку.	Адрес строки, содержащей префикс для поиска без учета регистра..	Индекс найденного элемента списка или CB_ERR при ошибке.	Не имеет эффекта при неудачном поиске. Если ОУ создан со стилем owner-drawn , но не имеет стиля CBS_HASSTRINGS, родительское окно получит сообщение WM_COMPAREITEM, если за-

					дан стиль CBS_SORT.
CB_ SETCURSEL	Выделяет указанный элемент списка и изменяет содержимое строки редактора.	Индекс (с 0) выделяемой строки, если равен -1, то текущая отметка снимается.	0	Индекс выделенного элемента списка или CB_ERR при ошибке или при индексе равном -1.	
CB_ SETEDITSEL	Выделяет участок текста в редакторе.	0	LOWORD содержит начальную позицию символа, а HIWORD – конечную. Используйте MAKELPARAM для формирования этого параметра.	TRUE, если успешно или CB_ERR при ошибке или установлен стиль CBS_DROPDOWNLIST.	Обе позиции задаются с 0, при этом, если первая из них равна -1, то отметка снимается, а если вторая равна -1, то выделяется весь текст. Для выделения первых 4-х символов надо задать 0 и 4.
CBN_ DBLCLK	Нотификация: извещает роди-	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только, если задан стиль

	тельское окно о двойном щелчке мыши на список.				CBS_SIMPLE .
CBN_ EDITCHANGE	Нотификация: извещает родительское окно об изменении содержимого редактора.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Эта нотификация передается после того, как система обновит экран и только, если не установлен стиль CBS_DROPDOWNLIST.
CBN_ KILLFOCUS	Нотификация: извещает родительское окно о потере фокуса.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	
CBN_ SELCHANGE	Нотификация: извещает родительское окно об изменении выбора в списке.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Не посылается, если выбор изменяется с помощью сообщения CB_SETCURSEL.
CBN_ SETFOCUS	Нотификация: извеща-	Смотрите описание WM_COM	Смотрите описание WM_CO	Смотрите описание WM_COMM	

	ет родительское окно о получении фокуса.	MAND	MMAND	AND	

1.3.4. Сообщения кнопок (класс Button)

Теперь рассмотрим основные сообщения и нотификации кнопок, обзор которых представлены н таблице 5.
Сообщения кнопок имеют префикс BM_, а нотификации – префикс BN_.
Смотрите также сообщения WM_DRAWITEM и WM_CTLCOLORBTN, описанные в таблице 1.

Таблица 5.

Сообщение (нотификация)	**Назначение**	**WPARAM**	**LPARAM**	**Возвращаемое значение**	**Примечание**
BM_ GETCHECK	Определить состояние кнопки	0	0	Состояние кнопки.	Только для кнопок, имеющих BS_AUTOCHECKBOX, BS_AUTORADIOBUTTON, BS_AUTO3STATE, BS_CHECKBOX, BS_RADIOBUTTON, или BS_3S

					TATE стиль.
BM_ GETIMAGE	Получить дескриптор иконы или картинки, связанных с кнопкой.	IMAGE_BITMAP – для картинки или IMAGE_ICON для иконы.	0	Соответствующий дескриптор или NULL.	
BM_ SETCHECK	Изменить состояние кнопки.	Новое состояние кнопки.	0		Только для кнопок типа “radio button” или “check box”.
BM_ SETIMAGE	Связать с кнопкой икону или картинку.	IMAGE_BITMAP – для картинки или IMAGE_ICON для иконы.	Соответствующий дескриптор.	Предыдущий дескриптор или NULL.	
BM_SET STYLE	Изменить стиль кнопки.	Комбинация стилей.	Флаг перерисовки (TRUE – да), (FALSE – нет).	0	
BN_ CLICKED	Нотификация: извещает родительское окно о клике на кнопку левой клавишей	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для активных кнопок.

	“мыши”				
BN_DBLCLK	Нотификация: извещает родительское окно о двойном клике на кнопку левой клавишей “мыши”	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для кнопок, имеющих стиль BS_OWNERDRAW или BS_RADIOBUTTON.
BN_DISABLE	Нотификация: извещает родительское окно о том, что кнопка стала неактивной.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для кнопок, имеющих стиль BS_OWNERDRAW
BN_KILLFOCUS	Нотификация: извещает родительское окно о потере фокуса.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для кнопок, имеющих стиль BS_NOTIFY
BN_PAINT	Нотификация: извещает родительское окно о том, что кнопка будет перерисована.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для кнопок, имеющих стиль BS_OWNERDRAW

BN_ SETFOCUS	Нотификация: извещает родительское окно о получении фокуса.	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Смотрите описание WM_COMMAND	Только для кнопок, имеющих стиль BS_NOTIFY

1.4. Пример программы ColorDialog

В заключение приводим пример программы на языке С, демонстрирующей обработку некоторых сообщений Windows, а также возможность отрисовки графического объекта, размер которого задан в мм заданным цветом с использованием цветных полос прокрутки для задания составляющих нужного цвета и синхронным отображением полученного цвета в статическом окне.

Данная программа построена в Microsoft Visual C++ 6.0 как Win32 application и содержит три основных модуля (файла):

1- ColorDialog.c содержит главную функцию приложения и функцию обработки сообщений главного окна приложения.

2- ChooseColorDlgProc.c содержит функцию обработки сообщений диалогового окна для установок размера в мм и цвета рисунка.

3 – ColorDialog.rc файл ресурсов для описания меню и диалогового окна.

Файл ColorDialog.c:

```
#include <windows.h>
#include <string.h>
#include "resource.h"
#include <stdlib.h>
#include <stdio.h>
LRESULT CALLBACK WndProc(HWND, unsigned, WPARAM, LPARAM);
WNDCLASS    wc;
HWND hWndMain;
HINSTANCE hInst;
HDC hdc;
int Size;
COLORREF Color;
```

```
 CALLBACK ChooseColorDlgProc(HWND hDlg,unsigned Message,WPARAM
wParam,LPARAM lParam);
 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{   // Главная функция приложения
   MSG msg;
 wc.style = CS_VREDRAW|CS_HREDRAW;
 wc.lpfnWndProc = WndProc;
 wc.cbClsExtra = 0;
 wc.cbWndExtra = 0;
 wc.hInstance = hInstance;
 wc.hIcon = LoadIcon(NULL,IDI_ASTERISK);
 wc.hCursor = NULL;
 wc.hbrBackground = CreateSolidBrush(RGB(255,255,255));
 wc.lpszMenuName =  "EXAMPLECOLOR";
 wc.lpszClassName = "luazw";
 if(!RegisterClass(&wc)) return FALSE;   // Регистрация класса главного окна.
 hInst=hInstance;
 // Создание главного окна
 hWndMain = CreateWindow( "luazw",
          "Пример отрисовки эллипса выбранного размера заданным цветом ",
          WS_OVERLAPPED|WS_VISIBLE|
          WS_MINIMIZEBOX|WS_MAXIMIZEBOX|
          WS_SYSMENU|WS_THICKFRAME,
          CW_USEDEFAULT,
          CW_USEDEFAULT,
          CW_USEDEFAULT,
          CW_USEDEFAULT,
          NULL,
          NULL,
           hInst,
           NULL );
 ShowWindow(hWndMain,SW_SHOW);
 UpdateWindow(hWndMain);
 while (GetMessage(&msg, NULL, 0, 0)) // Обработка цикла сообщений из
                                      //  очереди приложения
            DispatchMessage(&msg);
 return (msg.wParam);
}
 L RESULT CALLBACK WndProc(HWND hWnd, unsigned iMessage, WPARAM
 wParam, LPARAM lParam)
 {
   //    Функция обработки сообщений главного окна.
   static pdraw=0;
   HDC hdc;
   PAINTSTRUCT ps;
```

```
 RECT rc;
 static double SizeDraw;
 static double kx,ky,SizeX,SizeY;
 double x,y;
 static HBRUSH hbrEllipse;
 static COLORREF ColorDraw;
 static char buf[70];
 switch(iMessage)
 {
   case WM_CREATE:
  // При обработки сообщения о создании окна определяем коэффициенты
  // для перерасчета  размера картинки из мм в пикселы
    hdc=GetDC(0); // Получаем дескриптор контекста устройства для экрана.
    kx=(double)GetDeviceCaps(hdc,HORZRES)/(double)GetDeviceCaps(hdc,
                                        HORZSIZE);
    ky=(double)GetDeviceCaps(hdc,VERTRES)/(double)GetDeviceCaps(hdc,
                                        VERTSIZE);
    ReleaseDC(hWnd,hdc);
    break;
  case WM_COMMAND:
  switch(wParam)
  {
case IDM_COLOR: // Обработка пункта меню "Установки"
SizeDraw=Size;
   if(DialogBox(hInst,"COLOR",hWnd,ChooseColorDlgProc)) // Создаем диалог
                                                        //  установок
   {
    // Если завершили диалог по кнопке OK
     pdraw=1;
     SizeX=kx*SizeDraw;
     SizeY=ky*SizeDraw;
      ColorDraw=Color;
      EnableMenuItem(GetMenu(hWnd),IDM_DRAW,MF_ENABLED);
      DrawMenuBar(hWnd);
      if(!hbrEllipse) DeleteObject(hbrEllipse);
     }
  break;
  case IDM_DRAW:  // Обработка пункта меню "Эллипс"
SizeX=kx*Size;
SizeY=ky*Size;
hbrEllipse=CreateSolidBrush(ColorDraw);
InvalidateRect(hWnd,NULL,TRUE);
break;
case IDM_QUIT:              // Обработка пункта меню "Выход"
DestroyWindow(hWnd);   // Уничтожаем главное окно
break;
```

```
      }
 break;
    case WM_PAINT:   // Отрисовка эллипса заданного размера выбранным
                     // цветом в центре рабочей области
    if(!pdraw) return DefWindowProc(hWnd, iMessage, wParam, lParam);
    // если не было установок
    hdc=BeginPaint(hWnd,&ps);
      GetClientRect(hWnd,&rc);
      x=rc.right/2-SizeX/2.;
    y=rc.bottom/2-SizeY/2.;
    SelectObject(hdc,hbrEllipse);
    Ellipse(hdc,(int)x,(int)y,(int)(x+SizeX),(int)(y+SizeY));
      EndPaint(hWnd,&ps);
      break;

    case WM_DESTROY :  // Окно уничтожается
    if(!hbrEllipse) DeleteObject(hbrEllipse);
    PostQuitMessage(0);  // Ставим в очередь сообщений WM_QUIT
    break;
    default: return DefWindowProc(hWnd, iMessage, wParam, lParam);
 // обработка сообщений по умолчанию
   }
return 0l;
}
```

Файл ChooseColorDlgProc.c:

```
   #include <windows.h>
   #include "resource.h"
   CALLBACK ChooseColorDlgProc(HWND hDlg,unsigned Message,
                                        WPARAM wParam,LPARAM lParam)
   {
    // Функция обработки сообщений диалогового окна установок
    static HBRUSH hbrRedS,hbrGreenS,hbrBlueS;
    static HBRUSH hbrColor;
    static HBRUSH hbrDef;
    static ButtonColor;
   extern COLORREF Color;
    static HWND hWndRedS,hWndGreenS,hWndBlueS,hWndColorTR,
                hWndColorTG, hWndColorTB;
    static HWND hWndColor;
    static int npR=0,npG=0,npB=0,OldnpR=0,OldnpG=0,OldnpB=0;
    static char buf[10];
   static int OldSize=50,p;
   extern int Size;
switch (Message)
```

```
{
case WM_INITDIALOG:
ButtonColor=GetSysColor(COLOR_BTNFACE);
hbrRedS=CreateSolidBrush(RGB(255,0,0));
hbrGreenS=CreateSolidBrush(RGB(0,255,0));
hbrBlueS=CreateSolidBrush(RGB(0,0,255));
hWndRedS=GetDlgItem(hDlg,IDC_RED);
hWndGreenS=GetDlgItem(hDlg,IDC_GREEN);
hWndBlueS=GetDlgItem(hDlg,IDC_BLUE);
hWndColorTR=GetDlgItem(hDlg,IDC_COLORTR);
hWndColorTG=GetDlgItem(hDlg,IDC_COLORTG);
hWndColorTB=GetDlgItem(hDlg,IDC_COLORTB);
hbrDef=CreateSolidBrush(ButtonColor);
hWndColor=GetDlgItem(hDlg,IDC_COLOR);
npR=OldnpR;
npG=OldnpG;
npB=OldnpB;
Color=RGB(npR,npG,npB);
hbrColor=CreateSolidBrush(Color);
SetScrollRange(GetDlgItem(hDlg,IDC_RED),SB_CTL,0,255,TRUE);
SetScrollRange(GetDlgItem(hDlg,IDC_GREEN),SB_CTL,0,255,TRUE);
SetScrollRange(GetDlgItem(hDlg,IDC_BLUE),SB_CTL,0,255,TRUE);
SetScrollPos(GetDlgItem(hDlg,IDC_RED),SB_CTL,npR,TRUE);
SetScrollPos(GetDlgItem(hDlg,IDC_GREEN),SB_CTL,npG,TRUE);
SetScrollPos(GetDlgItem(hDlg,IDC_BLUE),SB_CTL,npB,TRUE);
wsprintf(buf,"%d",npR);
SetDlgItemText(hDlg,IDC_COLORTR,buf);
wsprintf(buf,"%d",npG);
SetDlgItemText(hDlg,IDC_COLORTG,buf);
wsprintf(buf,"%d",npB);
SetDlgItemText(hDlg,IDC_COLORTB,buf);
Size=OldSize;
wsprintf(buf,"%d",Size);
SetDlgItemText(hDlg,IDC_EDITSIZE,buf);
return TRUE;
case WM_CTLCOLORSCROLLBAR: // Изменяем цвета для полос пркрутки
if((HWND)lParam == hWndRedS) return (int)hbrRedS;
if((HWND)lParam == hWndGreenS) return (int)hbrGreenS;
if((HWND)lParam == hWndBlueS) return (int)hbrBlueS;
case WM_CTLCOLORSTATIC: // Изменяем цвета для статических окон
     if((HWND)lParam == hWndColor) return (int)hbrColor; // Изменяем цвет ок-
на, вкотором отображен выбранный цвет рисунка
 else if((HWND)lParam == hWndColorTR) // Изменяем цвет текста и фона для
выода красной составляющей цвета
  {
   SetTextColor((HDC)wParam,RGB(255,0,0));
```

```
    SetBkColor((HDC)wParam,ButtonColor);
    return (int)hbrDef;
   }
   else if((HWND)lParam == hWndColorTG) // Изменяем цвет текста и фона
                                        // для выода зеленой составляющей
                                        // цвета
    {
          SetTextColor((HDC)wParam,RGB(0,255,0));
          SetBkColor((HDC)wParam,ButtonColor);
          return (int)hbrDef;
                }
    else if((HWND)lParam == hWndColorTB) // Изменяем цвет текста и фона
                                         // для выода синей составляющей
                                         // цвета
          {
            SetTextColor((HDC)wParam,RGB(0,0,255));
             SetBkColor((HDC)wParam,ButtonColor);
             return (int)hbrDef;
           }
   else return 0;
case WM_HSCROLL:                    //обработка горизонтальной прокрутки
if((HWND)lParam == hWndRedS) // красная полоса
{
 switch LOWORD(wParam)
 {
   case   SB_LINERIGHT: // прокрутка вправо на 1 позицию
   if(npR >= 255) break;
   npR++;
   wsprintf(buf,"%d",npR);
   SetDlgItemText(hDlg,IDC_COLORTR,buf);
   Color=RGB(npR,npG,npB);
   DeleteObject(hbrColor);
   hbrColor=CreateSolidBrush(Color);
   SetScrollPos(GetDlgItem(hDlg,IDC_RED),SB_CTL,npR,TRUE);
   InvalidateRect(hWndColor,NULL,TRUE);
   UpdateWindow(hWndColor);
   break;

   case   SB_LINELEFT: // прокрутка влево на 1 позицию
   if(npR <= 0) break;
   npR--;
   wsprintf(buf,"%d",npR);
   SetDlgItemText(hDlg,IDC_COLORTR,buf);
   Color=RGB(npR,npG,npB);
   DeleteObject(hbrColor);
   hbrColor=CreateSolidBrush(Color);
```

```
  SetScrollPos(GetDlgItem(hDlg,IDC_RED),SB_CTL,npR,TRUE);
  InvalidateRect(hWndColor,NULL,TRUE);
  UpdateWindow(hWndColor);
  break;
  case SB_THUMBTRACK: // прокрутка с использованием
                                  //    прямоугольника на полосе прокрутки
  npR=HIWORD(wParam);
  wsprintf(buf,"%d",npR);
  SetDlgItemText(hDlg,IDC_COLORTR,buf);
  Color=RGB(npR,npG,npB);
  DeleteObject(hbrColor);
  hbrColor=CreateSolidBrush(Color);
  SetScrollPos(GetDlgItem(hDlg,IDC_RED),SB_CTL,npR,TRUE);
  InvalidateRect(hWndColor,NULL,TRUE);
  UpdateWindow(hWndColor);
  break;
}
 break;
}
if((HWND)lParam == hWndGreenS) // зеленая полоса
{
switch LOWORD(wParam)
{
case   SB_LINERIGHT: // прокрутка вправо на 1 позицию
if(npG >= 255) break;
npG++;
wsprintf(buf,"%d",npG);
SetDlgItemText(hDlg,IDC_COLORTG,buf);
Color=RGB(npR,npG,npB);
DeleteObject(hbrColor);
hbrColor=CreateSolidBrush(Color);
SetScrollPos(GetDlgItem(hDlg,IDC_GREEN),SB_CTL,npG,TRUE);
InvalidateRect(hWndColor,NULL,TRUE);
UpdateWindow(hWndColor);
break;
case   SB_LINELEFT: // прокрутка влево на 1 позицию
if(npG <= 0) break;
npG--;
wsprintf(buf,"%d",npG);
SetDlgItemText(hDlg,IDC_COLORTG,buf);
Color=RGB(npR,npG,npB);
DeleteObject(hbrColor);
hbrColor=CreateSolidBrush(Color);
SetScrollPos(GetDlgItem(hDlg,IDC_GREEN),SB_CTL,npG,TRUE);
InvalidateRect(hWndColor,NULL,TRUE);
UpdateWindow(hWndColor);
```

```
   break;
   case SB_THUMBTRACK:  // прокрутка с использованием прямоугольника
                                    //    на полосе прокрутки
   npG=HIWORD(wParam);
   wsprintf(buf,"%d",npG);
   SetDlgItemText(hDlg,IDC_COLORTG,buf);
   Color=RGB(npR,npG,npB);
DeleteObject(hbrColor);
hbrColor=CreateSolidBrush(Color);
SetScrollPos(GetDlgItem(hDlg,IDC_GREEN),SB_CTL,npG,TRUE);
InvalidateRect(hWndColor,NULL,TRUE);
UpdateWindow(hWndColor);
break;
}
break;
 }
if((HWND)lParam == hWndBlueS) // синяя полоса
{
   switch LOWORD(wParam)
   {
    case   SB_LINERIGHT: // прокрутка вправо на 1 позицию
    if(npB >= 255) break;
    npB++;
    wsprintf(buf,"%d",npB);
    SetDlgItemText(hDlg,IDC_COLORTB,buf);
    Color=RGB(npR,npG,npB);
    DeleteObject(hbrColor);
   hbrColor=CreateSolidBrush(Color);
   SetScrollPos(GetDlgItem(hDlg,IDC_BLUE),SB_CTL,npB,TRUE);
   InvalidateRect(hWndColor,NULL,TRUE);
   UpdateWindow(hWndColor);
   break;
   case   SB_LINELEFT: // прокрутка влево на 1 позицию
   if(npB <= 0) break;
   npB--;
   wsprintf(buf,"%d",npB);
   SetDlgItemText(hDlg,IDC_COLORTB,buf);
   Color=RGB(npR,npG,npB);
   DeleteObject(hbrColor);
   hbrColor=CreateSolidBrush(Color);
   SetScrollPos(GetDlgItem(hDlg,IDC_BLUE),SB_CTL,npB,TRUE);
   InvalidateRect(hWndColor,NULL,TRUE);
   UpdateWindow(hWndColor);
   break;
   case SB_THUMBTRACK: // прокрутка с использованием прямоугольника
                                    //    на полосе прокрутки
```

```
   npB=HIWORD(wParam);
   wsprintf(buf,"%d",npB);
   SetDlgItemText(hDlg,IDC_COLORTB,buf);
   Color=RGB(npR,npG,npB);
   DeleteObject(hbrColor);
   hbrColor=CreateSolidBrush(Color);
   SetScrollPos(GetDlgItem(hDlg,IDC_BLUE),SB_CTL,npB,TRUE);
   InvalidateRect(hWndColor,NULL,TRUE);
   UpdateWindow(hWndColor);
   break;
   }
   break;
 }
 break;
 case WM_COMMAND:
 switch LOWORD(wParam)
 {
  case IDOK:            //обработка нажатия кнопки OK
   DeleteObject(hbrRedS);
   DeleteObject(hbrGreenS);
   DeleteObject(hbrBlueS);
   DeleteObject(hbrDef);
   OldnpR=npR;
   OldnpG=npG;
   OldnpB=npB;
   Size=GetDlgItemInt(hDlg,IDC_EDITSIZE,&p,FALSE);
   OldSize=Size;
   EndDialog(hDlg,TRUE);
   break;
   case IDCANCEL:        //обработка нажатия кнопки CANCEL
   npR=OldnpR;
   npG=OldnpG;
   npB=OldnpB;
   DeleteObject(hbrRedS);
   DeleteObject(hbrGreenS);
   DeleteObject(hbrBlueS);
   DeleteObject(hbrDef);
   EndDialog(hDlg,FALSE);
   break;
   }
    break;
   }
   return 0;
  }
```

Файл ColorDialog.c:

```
#include "resource.h"
//////////////////////////////////////////////////////////////////////////
//
// Menu
//

EXAMPLECOLOR MENU DISCARDABLE
BEGIN
    MENUITEM "&Установки",              IDM_COLOR
    MENUITEM "&Эллипс",                 IDM_DRAW, GRAYED
    MENUITEM "&Выход",                  IDM_QUIT
END


//////////////////////////////////////////////////////////////////////////
//
// Dialog
//

COLOR DIALOG DISCARDABLE  0, 0, 236, 137
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_VISIBLE |
    WS_CAPTION |
    WS_SYSMENU
CAPTION "Текущие установки"
FONT 12, "Times New Roman"
BEGIN
    DEFPUSHBUTTON   "Выбор",IDOK,68,120,40,12
    PUSHBUTTON      "Возврат",IDCANCEL,132,120,40,12
    SCROLLBAR       IDC_RED,20,12,140,10,WS_TABSTOP
    SCROLLBAR       IDC_GREEN,20,42,140,10,WS_TABSTOP
    SCROLLBAR       IDC_BLUE,20,72,140,10,WS_TABSTOP
    RTEXT           "255",IDC_COLORTR,5,12,10,10
    RTEXT           "0",IDC_COLORTG,5,42,10,10
    RTEXT           "10",IDC_COLORTB,5,72,10,10
    LTEXT           "Цвет",IDC_COLORT,190,10,30,10
    LTEXT           "",IDC_COLOR,170,20,50,60
    LTEXT           "Размер в мм:",IDC_SIZET,50,90,50,10
    EDITTEXT        IDC_EDITSIZE,100,90,20,10
END
```

На рисунке 1.1 представлен вид диалогового окна для задания текущих установок.

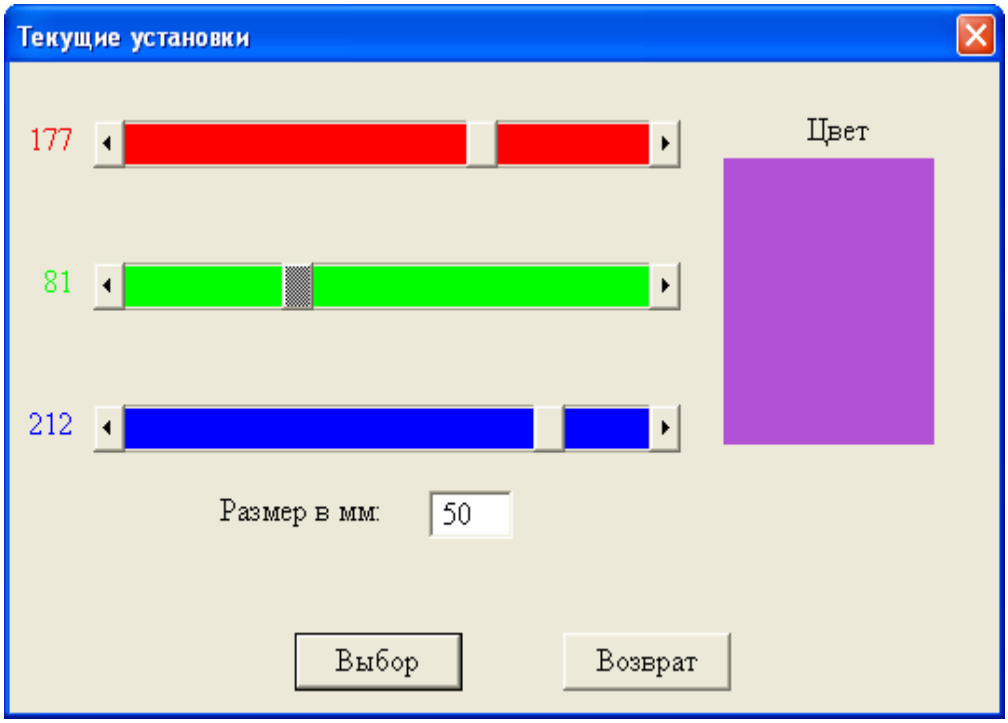


Рис. 1.1. Вид диалогового окна для задания текущих установок

2. Создание программ с использованием мастера генерации приложений AppWizard и библиотеки MFC

Мастер AppWizard используется для генерации основы исходных файлов будущей программы. Аналогично , мастер ClassWizard генерирует основную часть кода для определения производных классов и обработчиков сообщений, настройки MFC, управления диалоговыми окнами, а также выполнения других задач. Они позволяют создавать программы с большим набором сложных компонентов. К таковым относятся панели инструментов, строки состояния, контекстная справка, объекты OLE, средства доступа к базам данных и даже законченные меню с частично или полностью функционирующими командами открытия и сохранения файлов, печати, предварительного просмотра печати и выполнения других задач. После генерации основного кода программы с помощью мастеров остается добавить собственный код, определяющий логику работы программы. Преимущества такого подхода состоит в том, что вы не только используете уже написанный код MFC, но и освобождаетесь от необходимости создавать средства визуального интерфейса вручную, обеспечивая соответствие этого интерфейса требованиям Microsoft. Отметим, что функции MFC содержат вызовы функций Win32API, сохраняя возможность использования этого интерфейса, путем обращения к SDK-функциям, если в этом возникает необходимость.

Для начала, рассмотрим процесс создания простого графического приложения, выводящего графики основных тригонометрических функций в задаваемом произвольном диапазоне.

С помощью иконы запустить приложение MSDEV.EXE (рис. 2.1).

В пункте меню File выбрать пункт New, в открывшемся окне, выбрать в качестве создаваемого приложения “MFC AppWizard (exe)”, задать место расположения создаваемого проекта (Location) и имя самого проекта (Project name), после чего нажать кнопку OK (рис. 2.2).

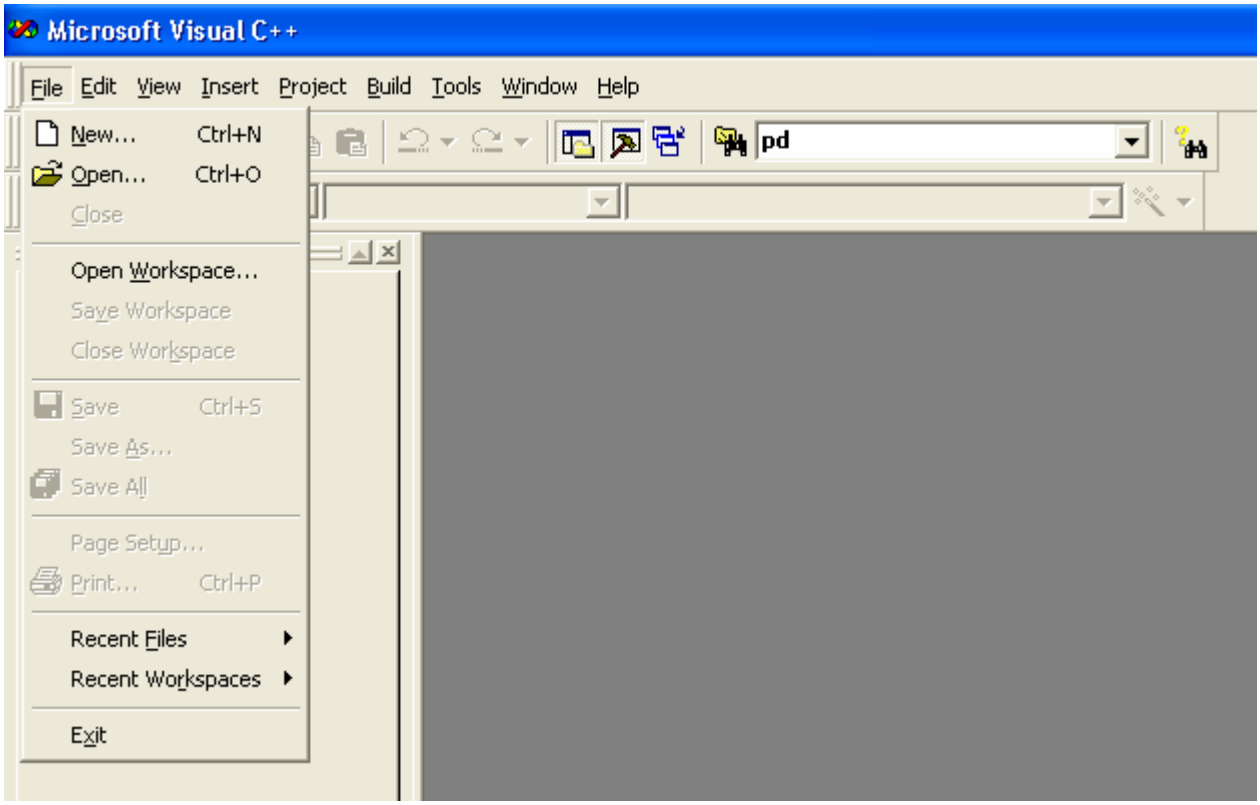


Рис. 2.1. Изначальное окно приложения MSDEV.EXE.

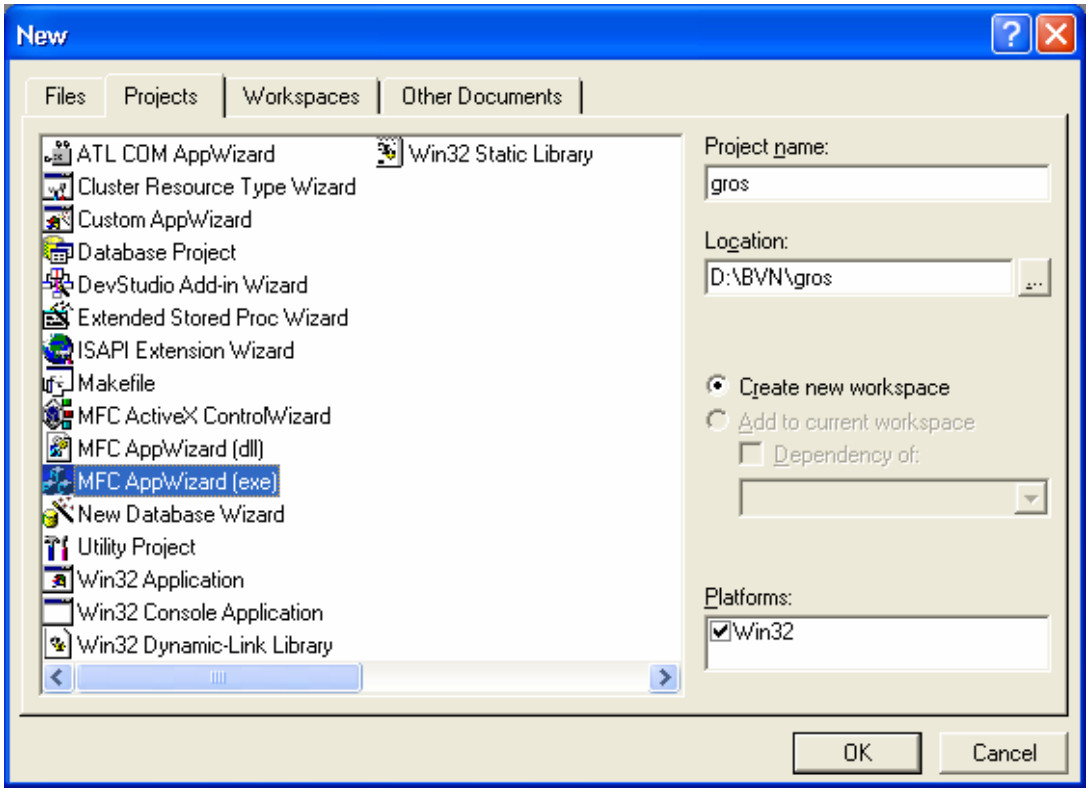


Рис. 2.2. Выбор проекта будущего приложения.

В директории D:\Bvn\CPP будет создан проект с именем gros (файл для открытия в будущем этого проекта – gros.dsw, находящийся в директории D:\Bvn\CPP\gros). Непосредственное создание проекта состоит из шести шагов, которые представляются своими окнами.

Шаг 1. В открывшемся окне у становить переключатель “Single document” и выбрать кнопку Next>.

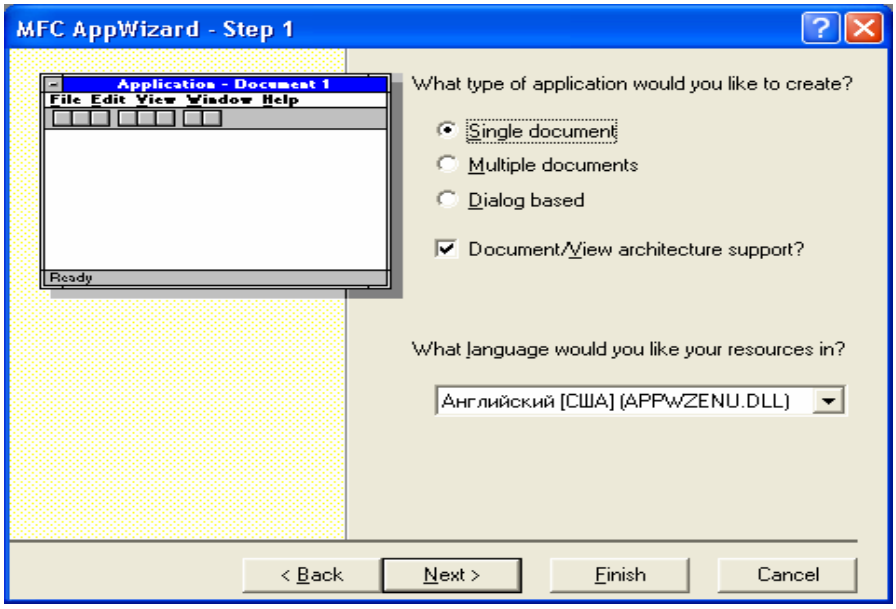


Рис.2.3. Шаг 1 создания проекта.

На шаге 2 выбрать Next>.

Шаг 3. Установить переключатели, как указано ниже и выбрать Next>.

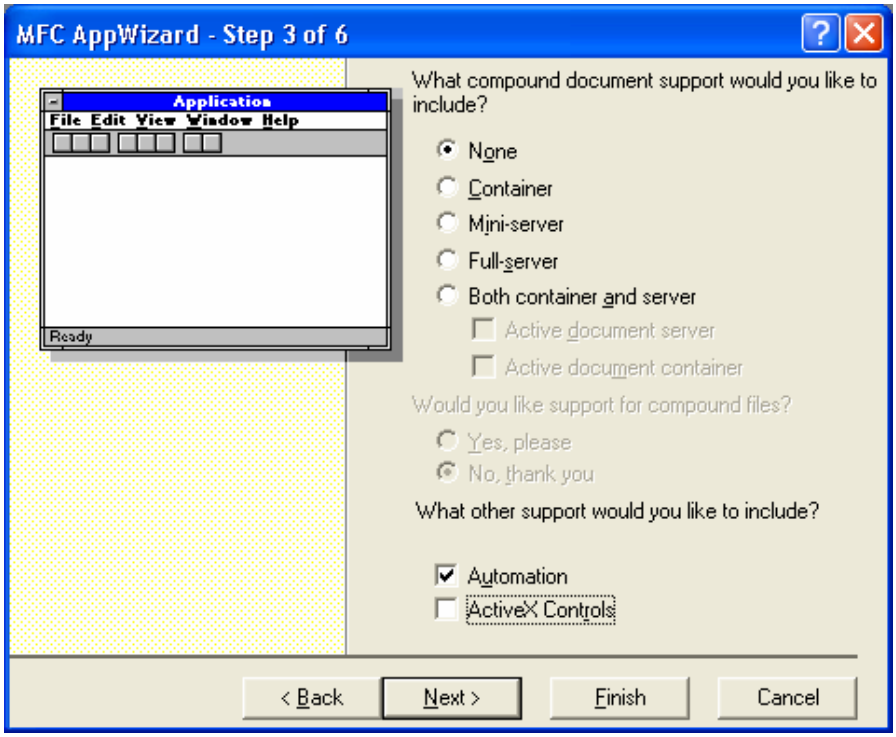


Рис. 2.4. Шаг 3 создания проекта.

Шаг 4. Сбросить все переключатели и выбрать NEXT>.

Рис. 2.5. Шаг 4 создания проекта.

Шаг 5. Установить переключатели, как указано ниже и выбрать Next>.

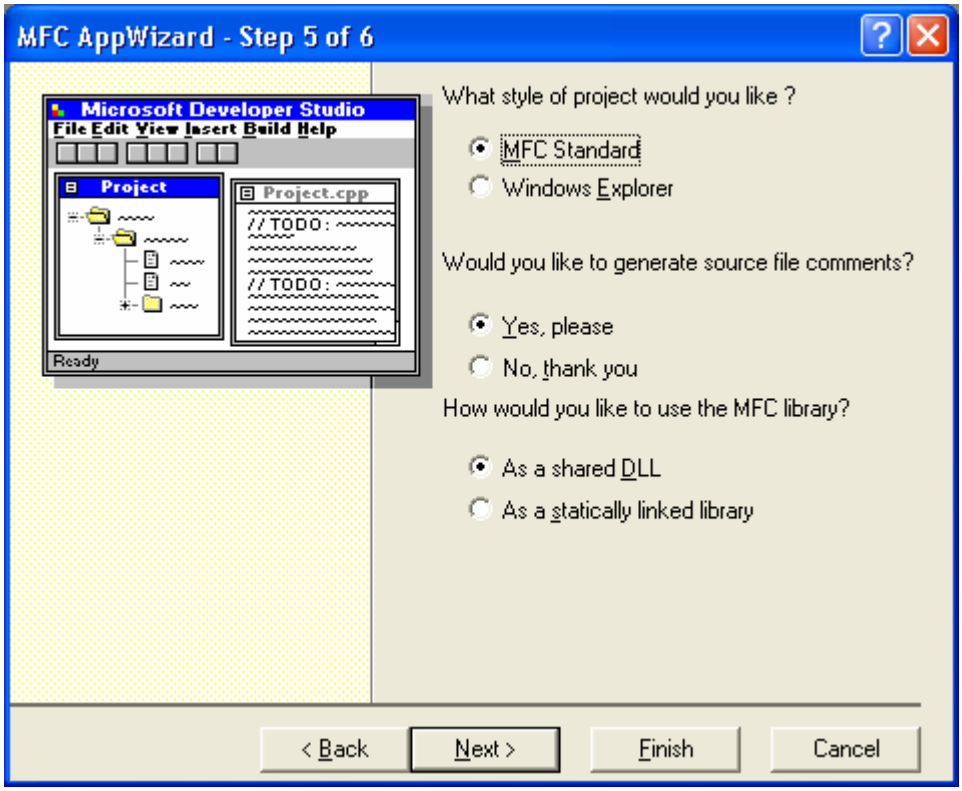


Рис. 2.6. Шаг 5 создания проекта.

Шаг 6. Выбрать Finish

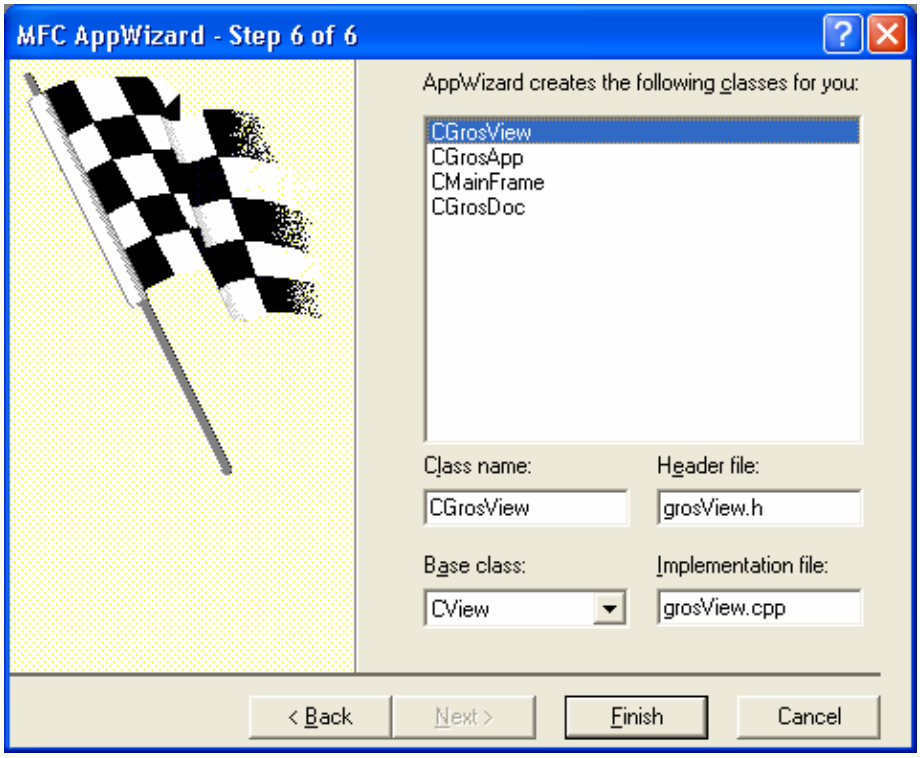
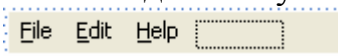


Рис. 2.7. Шаг 6 создания проекта.

В качестве базового класса выбирается CView, поскольку создается приложение, которое в любой, но одной и той же, координатной сетке должно строить выбираемые графики для четырех основных тригонометрических функций.

В результате будет создан файл grosView.cpp, который будет содержать заготовки функций созданных мастером AppWizard. Окно содержащее этот файл представлено ниже (рис. 2.8).

Создание (корректировка) меню: после выбора опции Resource будет отображено окно, представленное на рис. 2.9. После двойного щелчка левой клавишей «мыши» по IDR_MAINFRAME , будет отображено меню, которое было создано по умолчанию:



После щелчка правой клавишей «мыши» по пункту меню будет отображено всплывающее меню, вид которого показан на рис. 2.10.

При выборе пункта Properties будет отображено окно, позволяющее задавать (корректировать) название пункта меню и для конечного пункта меню соответственно сообщение, которое связано с данным пунктом меню. Например, для открывающегося пункта меню График (рис. 2.11.) и его корректировки (рис. 2.12) или для конечного пункта меню Sin (рис. 2.13) в открывающемся пункте меню График.

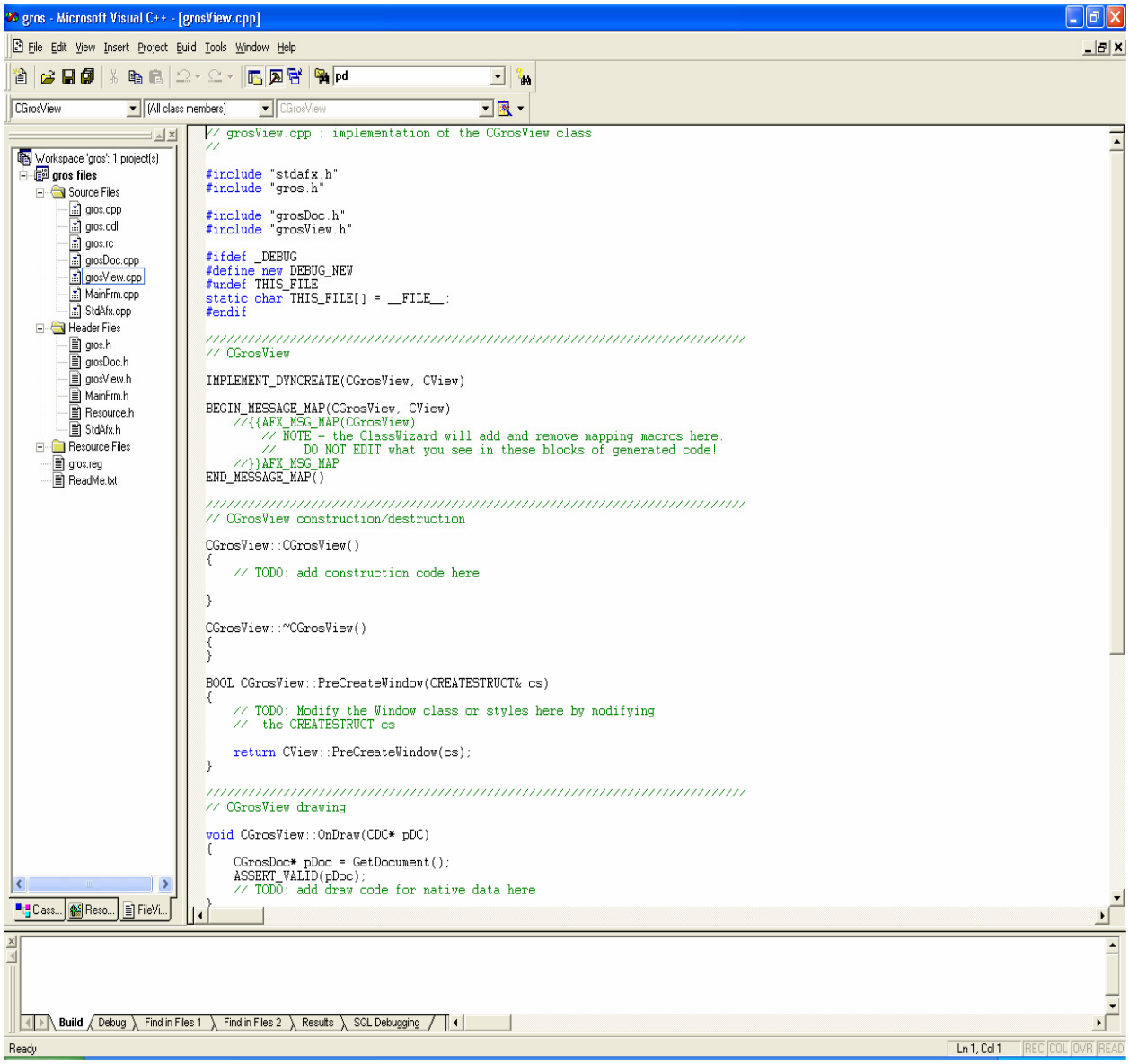


Рис. 2.8. Вид окна с функцией *Имя_проекта*View.cpp после построения приложения.

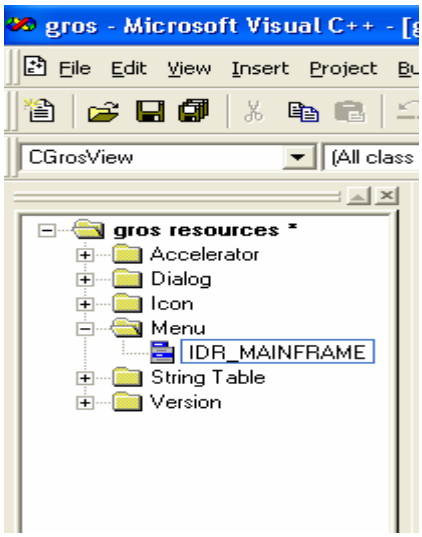


Рис. 2.9. Изначальное окно для корректировки ресурсов.

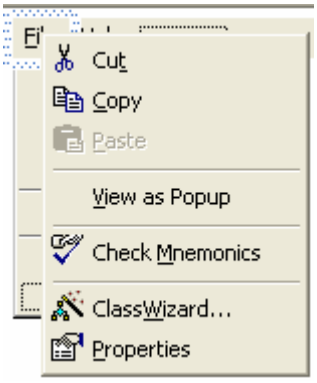


Рис. 2.10. Всплывающее меню для корректировки меню приложения.

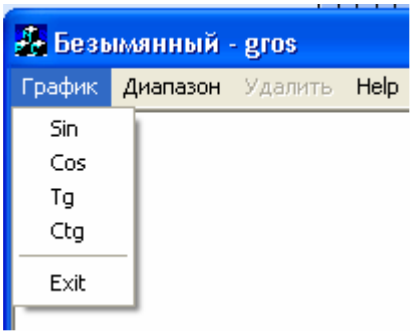


Рис. 2.11. Пример окна при выборе открывающегося пункта меню.

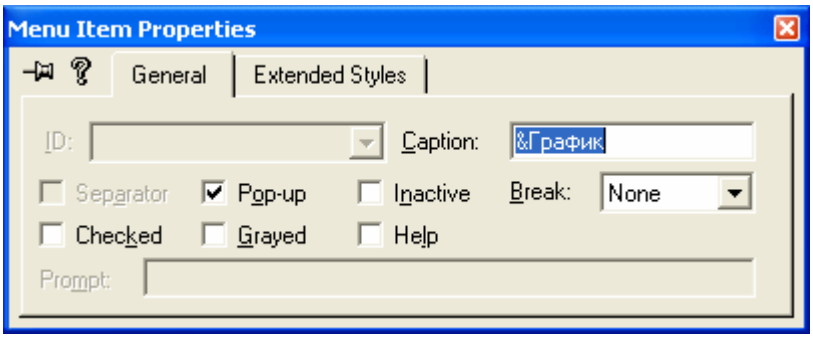


Рис. 2.12. Пример окна для корректировки открывающегося пункта меню.

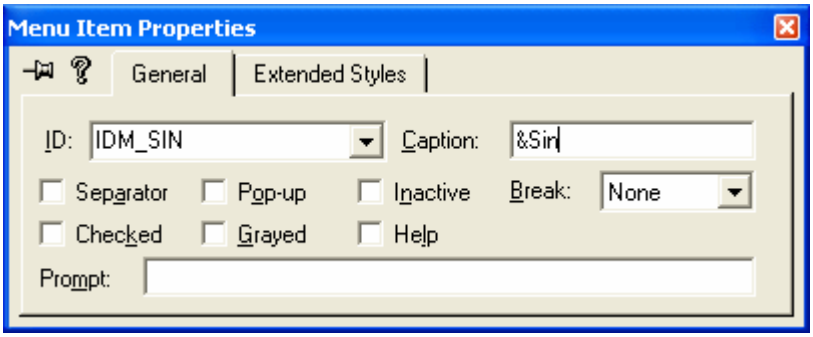


Рис. 2.13. Пример окна для корректировки конечного пункта меню.

Для добавление функции, отвечающей за выбор пункта меню, необходимо выбрать пункты меню View → ClassWizard. В открывшемся окне (рис. 2.14) выбрать в списке Object IDs идентификатор соответствующего пункта меню и нажать клавишу Add Function. Соответственно, будет отображено окно (рис. 2.15), содержащее имя соответствующей добавляемой функции. После нажатия в нем на клавишу ОК, необходимо нажать на клавишу ОК и в предыдущем окне (рис. 2.14).

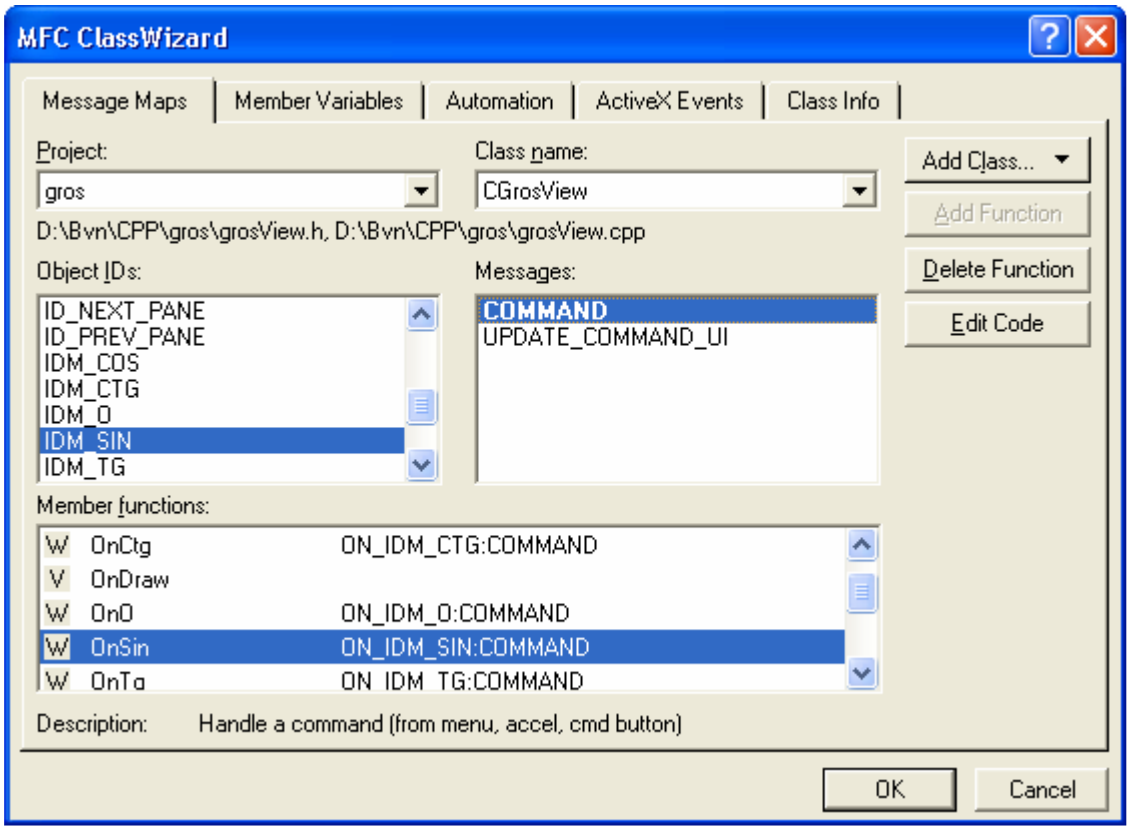


Рис. 2.14. Окно для добавления функции, связанной с пунктом меню.

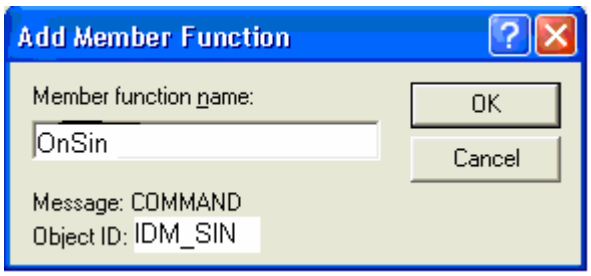


Рис. 2.15. Окно для подтверждения имени добавленной функции.

Естественно будет добавлена пустая функция, вид которой приведен ниже. Ее фактическое наполнение дело разработчика приложения.

```
void CGrosView::OnSin()
{
        // TODO: Add your command handler code here

}
```

Для создания окна диалога выбираем вкладку Resource и в открывшемся окне (рис. 2.16) отмечаем тип ресурса Dialog.

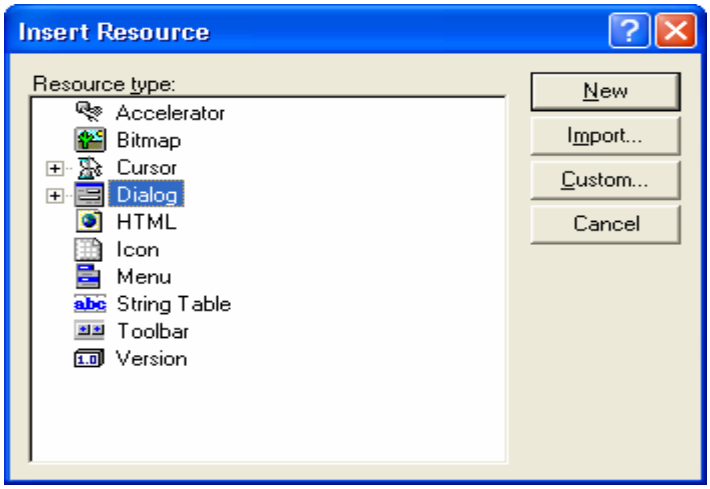


Рис. 2.16. Начальное окно добавления и корректировки окна диалога.

Нажимаем кнопку New. Будет открыто пустое окно для создаваемого диалога (рис. 2.17).

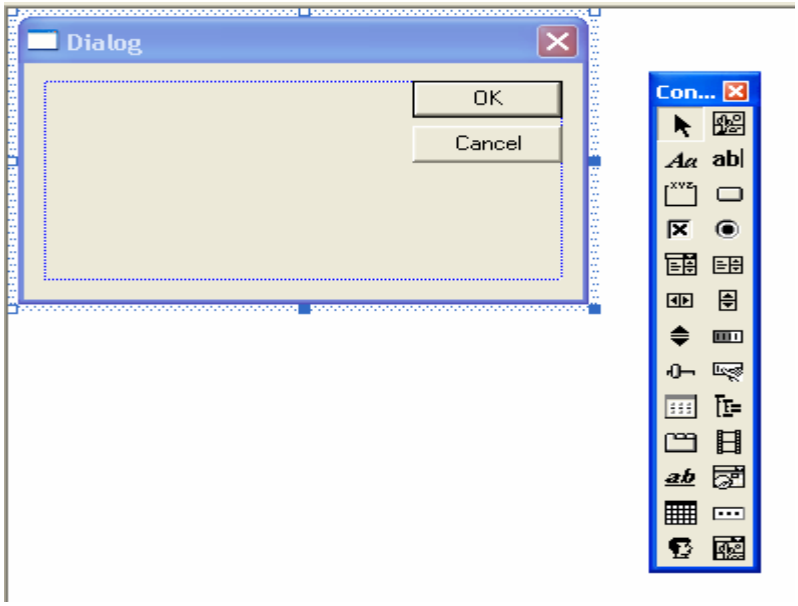


Рис. 2.17. Пустое окно для создания будущего окна диалога.

С помощью «мыши» перетаскиваем в нужное место окна диалога необходимые органы управления и определяем их размеры. Например, для ввода значений диапазона по оси X, окно может иметь вид, представленный на рис. 2.18.

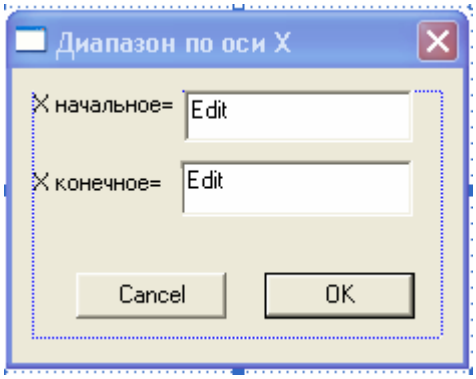


Рис. 2.18. Вид окна диалога для ввода диапазона по оси X.

Затем щелкаем правой клавишей мыши по созданному окну диалога и в открывшемся окне выбираем пункт MFC APP WIZARD, чтобы создать class, связанный с этим окном. Будет выведено окно представленное на рис. 2.19.

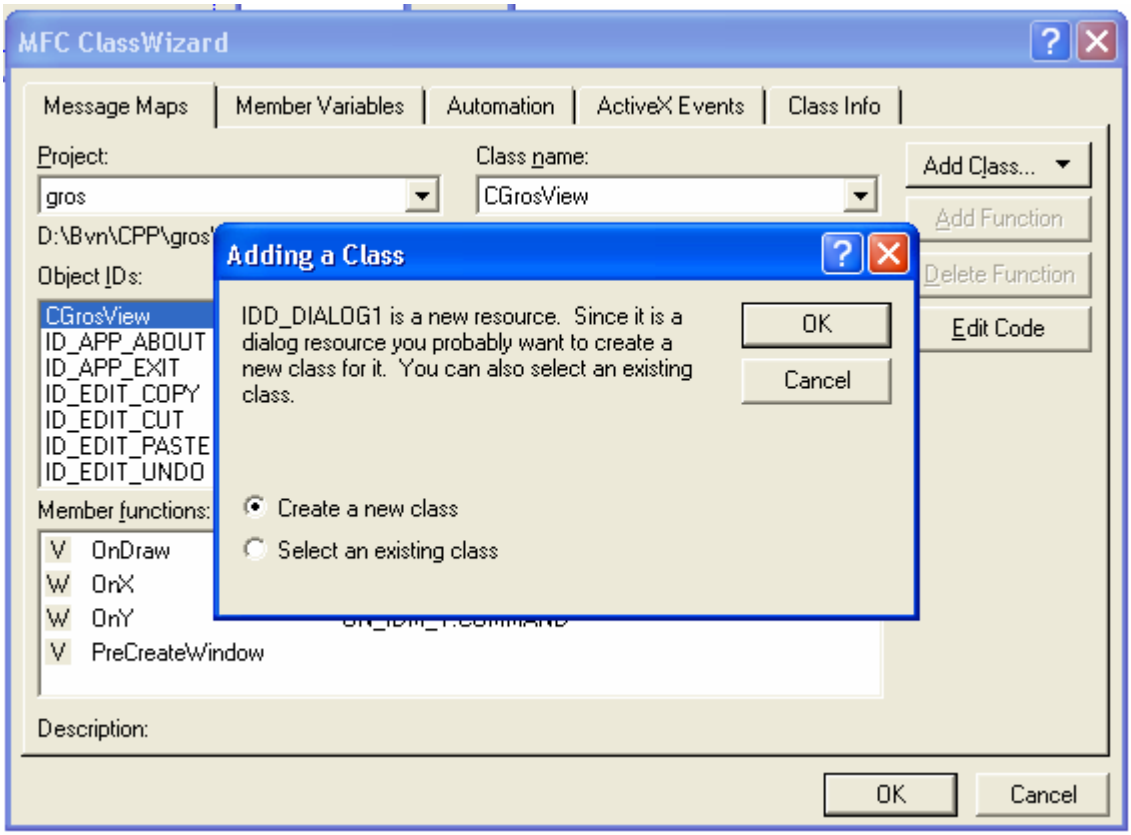


Рис. 2.19. Окно создания соответствующего класса для нового окна диалога.

Нажимаем клавишу OK и в открывшемся окне (рис. 2.20) в поле Name задаем имя класса.

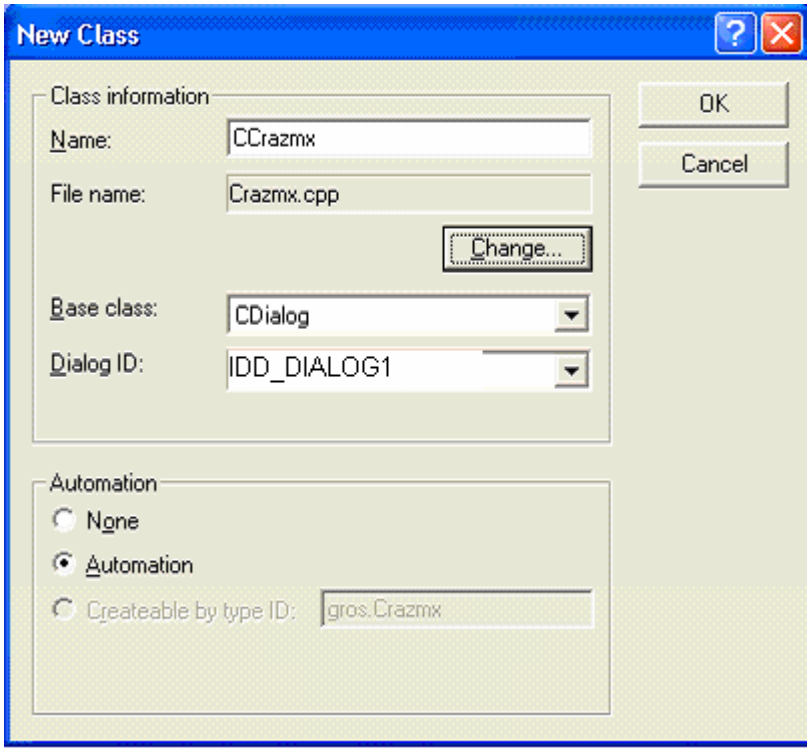


Рис. 2.20. Окно для задания имени класса для нового окна диалога.

Для добавления функции, которая отвечает за действия, которые должны быть выполнены при выборе соответствующего пункта меню, необходимо выбрать пункт меню View, а в нем пункт ClassWizard и в открывшемся окне (рис. 2.21) в списке Object Ids выбрать идентификатор, назначенный данному пункту меню, а в списке Messages строку COMMAND. После этого нажать клавишу Add Function.

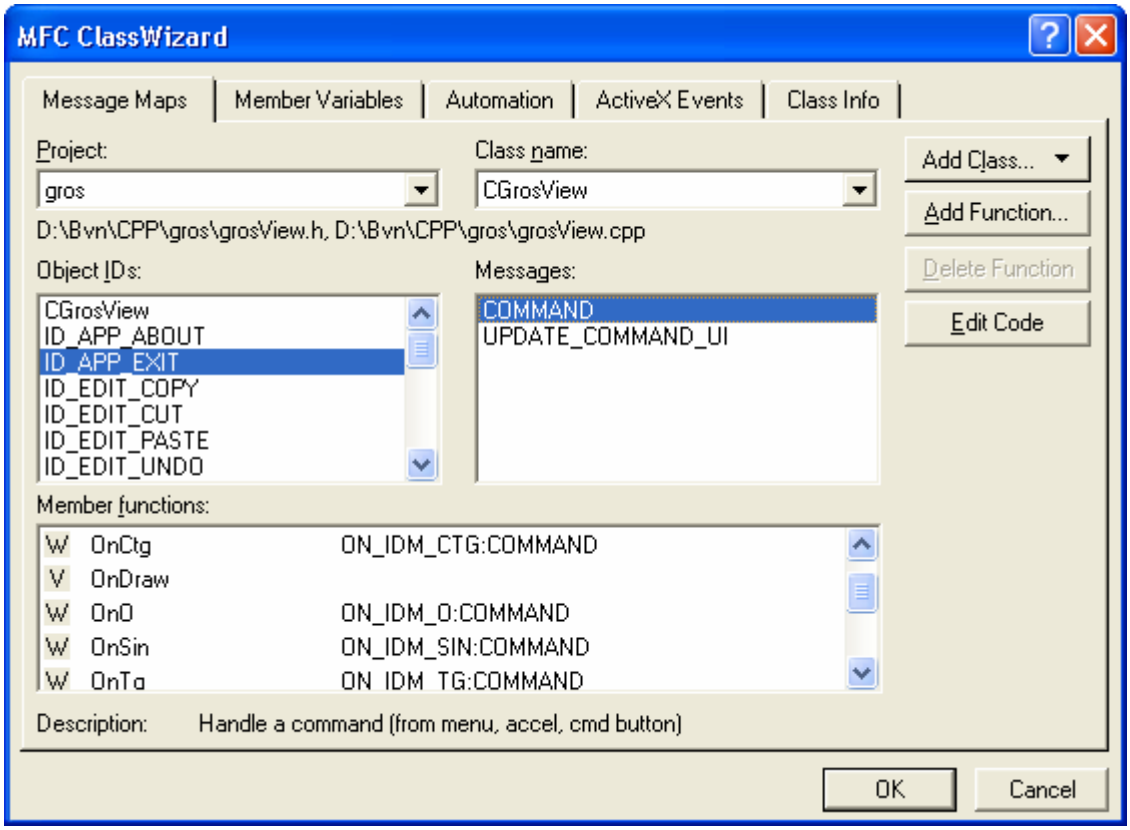


Рис. 2.21. Окно для определения функции, связанной с новым окном диалога.

Откроется окно (рис. 2.22), в котором будет определено имя добавленной функции. В этом окне необходимо нажать клавищу ОК.

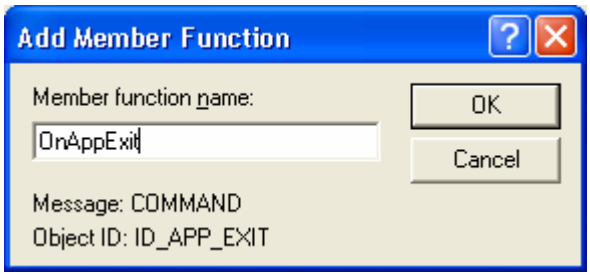


Рис. 2.22. Окно для подтверждения имени созданной функции для нового окна диалога.

Наполнение созданной функции окна диалога соответствующим кодом дело разработчика данного приложения.

Для окончательного создания приложения необходимо создать необходимые окна диалога. Это окно определяющее диапазон изменения X и окно задающее min и max значения Y, а также окно, позволяющее выбрать цвет для

отображения выбранного графика. В файле grosView.cpp определить функции, задающие последовательность действий при выборе того или иного пункта меню. В нашем случае – последовательность операторов выводящих выбранный график, а также операторы, отображающие окна диалогов для задания соответствующих диапазонов изменения X и Y, а также выбора соответствующих цветов графиков.

Пример окна будущего приложения, которое может выглядеть следующим образом (рис. 2.23), приведен ниже. Все зависит от выбранного диапазона изменения X и предельных отображаемых значений для Tg(x) и Ctg(x), а также от цветов отображения для выбранных в приведенном примере функций Sin(x) и Tg)x).

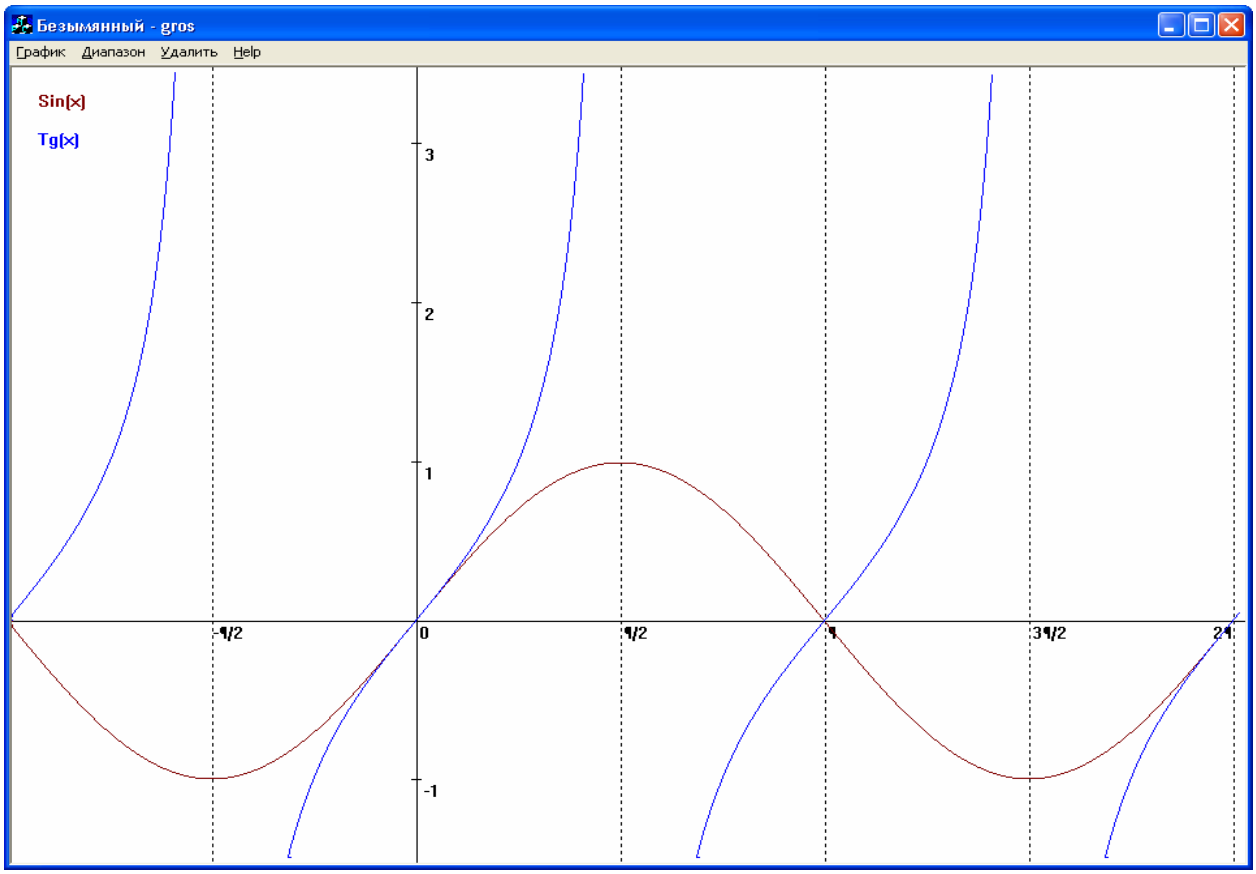


Рис. 2.23. Пример главного окна приложения, выводящего графики основных тригонометрически функций в заданном диапазоне.

2.1. Прокрутка и преобразование координат

Рассмотрим способ поддержки прокрутки в классе CScrollView. Когда документ открывается впервые, левый верхний угол рисунка (текста) отображается в верхнем левом углу окна. Однако при прокрутке документа с помощью полосы прокрутки MFC корректирует значение атрибута называемого *началом области просмотра*.

Размер области просмотра задается при обращении к функции SetScrollSizes, которая вызывается виртуальной функцией OnInitialUpdate() класса CScrollView, и изначально выглядит следующим образом:

```
void CName View::OnInitialUpdate()
{
        CScrollView::OnInitialUpdate();

        CSize sizeTotal;
        // TODO: calculate the total size of this view
        sizeTotal.cx = sizeTotal.cy = 100;
        SetScrollSizes(MM_TEXT, sizeTotal);
}
```

Здесь *Name* – имя реализуемого проекта. Первый параметр функции SetScrollSizes задает режим отображения, который определяет систему координат и единицы измерения, используемые для вывода текста и графики. Второй параметр - объект класса CSize, определяющий горизонтальный и вертикальный размеры области просмотра. При необходимости эти размеры в процессе выполнения программы с помощью функции SetScrollSizes могут быть изменены. Например, если программа выводит в окно графический объект, то эти параметры должны определять ширину и высоту прямоугольника, содержащего этот объект.

Начало области просмотра определяет положение текста или графики относительно окна. Если прокрутить документ, то MFC скорректирует начало области просмотра так, что точки рисунка появятся на новых позициях в окне представления рис.2.24.

После того как MFC скорректирует начало области просмотра, функция OnDraw перерисует линии в окне представления, задавая те же координаты для каждой линии. Из-за изменения начала области просмотра линии будут выведены с учетом прокрутки рис.2.25. Привлекательность такого подхода состоит в том, что не нужно модифицировать функцию OnDraw для работы с полосами прокрутки, операции прокрутки реализуются автоматически.

Координаты, заданные при рисовании объекта, называются *логическими координатами*. Физические координаты объекта внутри окна называют *координатами устройства*. На рис. 2.26 показано их различие.

Следует иметь в виду то, что передаваемые обработчику сообщений мыши координаты указателя мыши имеют логические значения.

При создании приложения с окном класса CEditView, позволяющего открывать, просматривать и корректировать текстовые файлы, прокрутка окна реализуется автоматически.

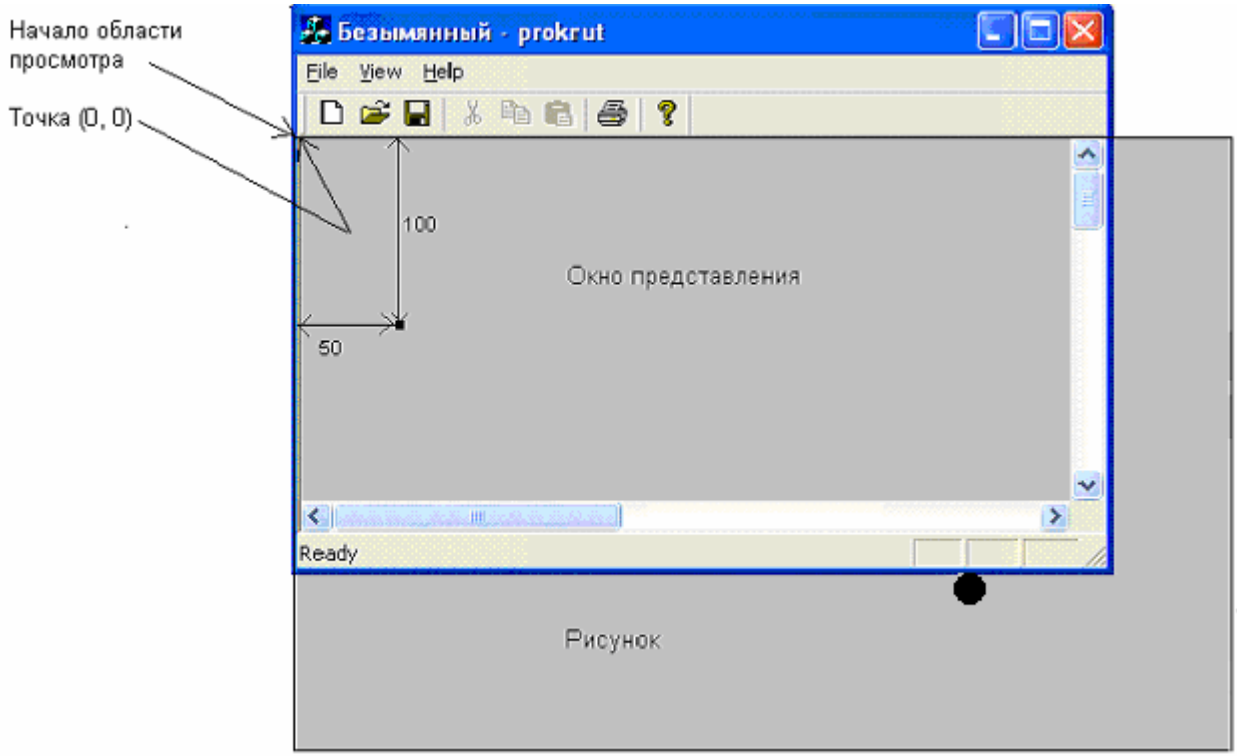


Рис. 2.24. Положение окна представления по отношению к области просмотра.

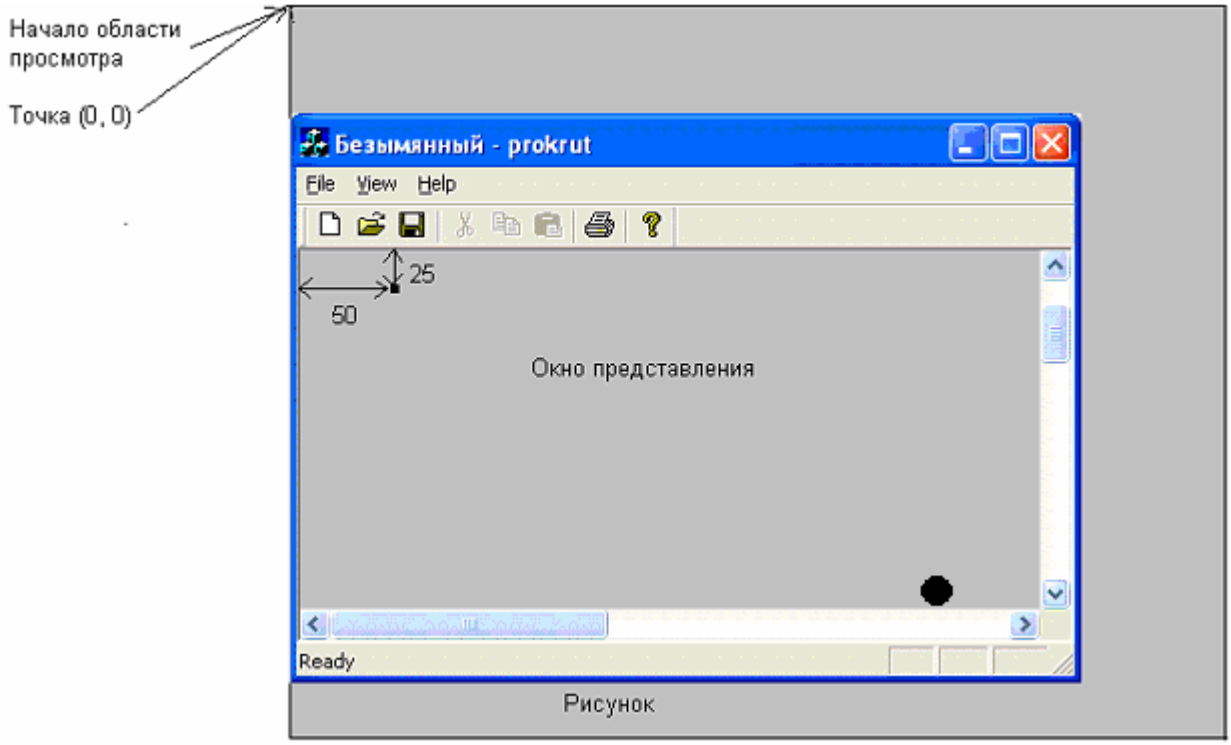


Рис. 2.25. Окно представления после прокрутки области просмотра вниз на 75 пикселей.

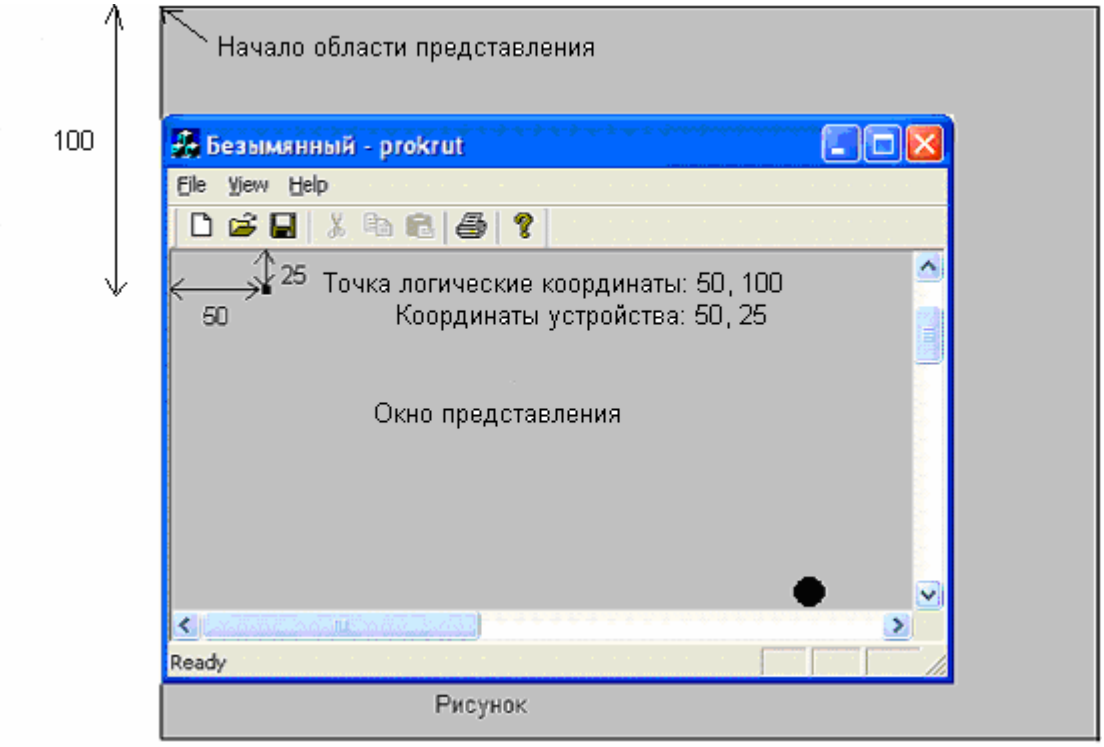


Рис. 2.26. Логические координаты в сравнении с координатами устройства (документ прокручен вниз на 75 пикселей)

2.2. Печать и предварительный просмотр

При генерации новой программы с использованием мастера AppWizard, в нее можно включить основные средства поддержки печати и предварительного просмотра, установив опцию Printing And Print Preview в диалоговом окне (Step 4) AppWizard (рис. 1.4), в следствие чего команды Print…, Print Preview и Print Setup… добавляются в меню программы File. При реализации мастером AppWizard команда Print… печатает ту часть документа, которая помещается на одной странице. Оставшаяся часть документа игнорируется. Подобным образом команда Print Preview отображает распечатку, появляющуюся на одной странице документа. Команды Print и Print Preview вызывают функцию OnDraw, чтобы сгенерировать реально выводимую текстовую или графическую информацию. Команда Print Setup… отображает обычное диалоговое окно Print Setup, позволяющее выбрать тип принтера и задать его установки (например, книжная или альбомная ориентация печати).

Если размер рисунка превышает размеры листа принтера и необходимо предварительно посмотреть разбивку этого рисунка по листам печати, а также вывести этот рисунок с разбивкой его по листам печати на принтер, необходимо усовершенствовать организацию функций класса CView. Если размеры рисунков, с которыми будет иметь дело приложение, достаточно большие, то естественно необходимо при организации приложения пользоваться классом с прокруткой окна представления CScrollView. Кроме того, необходимо переопределить две виртуальные функции печати класса CScrollView: OnBeginPrinting

и OnPrepareDC следующим образом. Жирным текстом выделена добавленная информация. Если функция OnPrepareDC отсутствует среди функций класса CScrollView, то ее следует определить вручную с помощью мастера MFC ClassWizard. Она необходима для переопределения текущей просматриваемой/печатаемой страницы. Таким образом , строка за строкой, продолжается просмотр и печать всего документа, как показано на рис. 2.27.

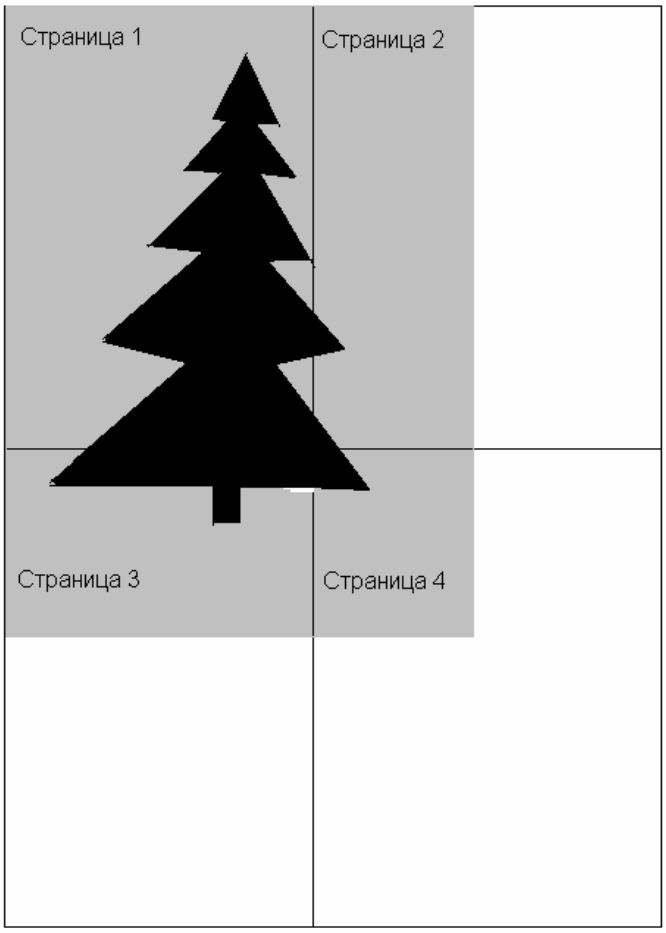


Рис. 2.27. Разбивка рисунка на страницы при его просмотре и печати, если рисунок превышает размер печатной страницы.

```
int m_PageHeight, m_PageWidth, m_NumRows, m_NumCols;
void COnkrscrView::OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo)
{
        // TODO: add extra initialization before printing
   extern CSize sizeTotal;  // Размеры рисунка
   m_PageHeight=pDC->GetDeviceCaps(VERTRES);
   m_PageWidth=pDC->GetDeviceCaps(HORZRES);
   m_NumRows=sizeTotal.cy/m_PageHeight+(sizeTotal.cy%m_PageHeight>0);
   m_NumCols=sizeTotal.cx/m_PageWidth+(sizeTotal.cx%m_PageWidth>0);
   pInfo->SetMinPage(1);
   pInfo->SetMaxPage(m_NumRows*m_NumCols);
   CScrollView::OnBeginPrinting(pDC, pInfo);
}
```

```
void COnkrscrView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
        // TODO: Add your specialized code here and/or call the base class
    CScrollView::OnPrepareDC(pDC, pInfo);
     if(pInfo==NULL) return;
     int CurRow, CurCol;
      CurRow=pInfo->m_nCurPage / m_NumCols +
                  (pInfo->m_nCurPage % m_NumCols > 0);
     CurCol=(pInfo->m_nCurPage-1)%m_NumCols+1;
      pDC->SetViewportOrg( -m_PageWidth * (CurCol - 1),
                                    -m_PageHeight * (CurRow - 1));
}
```

Пример программы, выводящей в окно любые BMP и JPG файлы с возможностью при необходимости их прокрутки и печати с предварительным просмотром вида печати

Приведены только функции, которые подверглись корректировке. Жирным текстом выделены добавления и изменения. Вывод на печать может производиться с увеличением/уменьшением рисунка до размера листа, либо по реальным размерам с возможностью при необходимости разбивки по страницам.

```
HINSTANCE hInst;
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        if( !CFrameWnd::PreCreateWindow(cs) )
                return FALSE;
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
    hInst=cs.hInstance;
        return TRUE;
}

int po=0,rpx,rpy;
CSize sizeTotal;
RECT rc;
TEXTMETRIC tm;
HDC hMemDC;
HBITMAP hBmp;
BITMAP bm;
void COnkrscrView::OnDraw(CDC* pDC)
{
        COnkrscrDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
    if(!po) { GetClientRect(&rc); pDC->GetTextMetrics(&tm); po=1;
```

```
                    sizeTotal.cx = rc.right; sizeTotal.cy = rc.bottom;
                     SetScrollSizes(MM_TEXT, sizeTotal);  return; }
          extern HINSTANCE hInst;
          extern char szFile[120];
          extern int rb;  //  Вид вывода реальный либо с масштабированием по
листу
          hBmp=(HBITMAP)LoadImage(hInst,szFile,IMAGE_BITMAP,0,0,
                                                LR_LOADFROMFILE);
          GetObject(hBmp,sizeof(BITMAP),&bm);
          hMemDC=CreateCompatibleDC(NULL);
          SelectObject(hMemDC,hBmp);
          sizeTotal.cx=bm.bmWidth;
          sizeTotal.cy=bm.bmHeight;
          SetScrollSizes(MM_TEXT, sizeTotal);
          if(rb)
          { if(rb==IDC_RADIO2)
           { rpx=pDC->GetDeviceCaps(HORZRES);
             rpy=pDC->GetDeviceCaps(VERTRES);
             StretchBlt(pDC->m_hDC,0,0,rpx,rpy,hMemDC,0,0,bm.bmWidth,
                        bm.bmHeight,SRCCOPY);
          }
          else BitBlt(pDC->m_hDC,0,0,bm.bmWidth,bm.bmHeight,hMemDC,0,0,
                        SRCCOPY);
        }
        if(!rb) BitBlt(pDC->m_hDC,0,0,bm.bmWidth,bm.bmHeight,hMemDC,0,0,
                        SRCCOPY);
        DeleteObject(hBmp);
        DeleteDC(hMemDC);
      }

int pin=0;
int m_PageHeight, m_PageWidth, m_NumRows, m_NumCols;
void COnkrscrView::OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo)
{
        // TODO: add extra initialization before printing
   extern CSize sizeTotal;
   CCPri dp;
  dp.DoModal();
   m_PageHeight=pDC->GetDeviceCaps(VERTRES);
   m_PageWidth=pDC->GetDeviceCaps(HORZRES);
   m_NumRows=sizeTotal.cy/m_PageHeight+(sizeTotal.cy%m_PageHeight>0);
   m_NumCols=sizeTotal.cx/m_PageWidth+(sizeTotal.cx%m_PageWidth>0);
   pInfo->SetMinPage(1);
   pInfo->SetMaxPage(m_NumRows*m_NumCols);
   if(pInfo->m_bPreview && !pin) pin=1;
   else { pin=0; goto xx; }
```

```
  xx:   CScrollView::OnBeginPrinting(pDC, pInfo);
}

void COnkrscrView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add cleanup after printing
        extern int rb;
        rb=0;
}

void COnkrscrView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
        // TODO: Add your specialized code here and/or call the base class
    CScrollView::OnPrepareDC(pDC, pInfo);
     if(pInfo==NULL) return;
     int CurRow, CurCol;
     CurRow=pInfo->m_nCurPage/m_NumCols+(pInfo-
>m_nCurPage%m_NumCols>0);
     CurCol=(pInfo->m_nCurPage-1)%m_NumCols+1;
     pDC->SetViewportOrg(-m_PageWidth*(CurCol-1),-m_PageHeight*(CurRow-
1));
}

#include <string.h>
#include <direct.h>
#include "Djpgd.h"
char szDirName[80],szFile[80],rab[80];
int pj=0,pd=0;
void COnkrscrView::OnFileOpen()
{
        // TODO: Add your command handler code here
  OPENFILENAME ofn;
  char szFileTitle[80],*u;
  char  szFilter[256]="BMP Files(*.BMP)|*.BMP|JPEG Files(*.JPG)|*.JPG|All
iles(*.*)|*.*|";
  if(pj) { pj=0; DeleteFile(rab); }
  CWnd *cw=GetWindow(GW_HWNDFIRST);
  SetScrollPos(SB_HORZ,0,TRUE);
  SetScrollPos(SB_VERT,0,TRUE);
  szFile[0] = '\0';
  if(!pd) { pd=1; _getcwd(szDirName,70); }
  for(int i = 0;szFilter[i]!='\0';i++)
  { if(szFilter[i] == '|') szFilter[i]='\0'; }
  memset(&ofn, 0, sizeof(OPENFILENAME));
  ofn.lStructSize = sizeof(OPENFILENAME);
  ofn.hwndOwner = cw->m_hWnd;
```

```
 ofn.lpstrFilter = szFilter;
 ofn.nFilterIndex = 1;
 ofn.lpstrFile= szFile;
 ofn.nMaxFile = sizeof(szFile);
 ofn.lpstrFileTitle = szFileTitle;
 ofn.nMaxFileTitle = sizeof(szFileTitle);
 ofn.lpstrInitialDir = szDirName;
 ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
 if(GetOpenFileName(&ofn))
 { lstrcpy(szDirName,szFile);
  u=strrchr(szDirName,'\\'); *u=0;
   if(!(strstr((const char *)szFile, (const char *)".JPG") || strstr((const char *)szFile,
        (const char *)".jpg") || strstr((const char *)szFile,(const char *)".BMP") ||
        strstr((const char *)szFile,(const char *)".bmp")))
   { MessageBox("Тип файла должен быть BMP или JPG.","Внимание!"); return;
}
   InvalidateRect(NULL);
   if(strstr((const char *)szFile,(const char *)".JPG") || strstr((const char *)szFile,
      (const char *)".jpg"))
   { CDjpgd dd;
     dd.DoModal();
     system(szFile);
     pj=1;
     char *u;
     u=strrchr(szFile,'.');
     lstrcpy(u+1,"bmp");
     lstrcat(rab,szFile);
   }
  CDC *dd=GetDC();
  OnDraw(dd);
 }
}
```

2.3. Добавление средств разделения окна

Рассмотрим, как к окну программы добавить *вешку разбиения*, позволяющую разделить окно представления на два отдельных окна, называемых также панелями. Границу между двумя панелями называют *линией разбивки*. Два окна представления отображают содержимое одного окна, а прокручиваются независимо друг от друга, отображая различные его части. Чтобы разделить окно, необходимо выполнить двойной щелчок на вешке разбиения (чтобы разделить окно на две равные части) или перетащить линию разбивки в требуемую позицию. Когда окно разделено на два, вертикальная полоса прокрутки является для них общей, однако каждая панель имеет собственную горизонтальную полосу прокрутки. В горизонтальном направлении панели прокручиваются независимо друг от друга. При создании программы с использованием мастера AppWizard,

если используется средство разделения окна, можно создать две вешки разбиения: горизонтальную и вертикальную. Таким образом, изначальное окно можно разделить на четыре части

Для добавления средства разделения окна в новую программу, используя мастер AppWizard, необходимо в диалоговом окне мастера (Step 4) выполнить следующие действия.

1. Щелкните на кнопке Advanced…, чтобы открыть диалоговое окно Advanced Options.
2. В диалоговом окне Advanced Options откройте вкладку WindowStyles для установки дополнительных опций главного окна приложения.
3. Установите опцию Use Split Window.

В новом приложении в файле MainFrm.h в классе CMainFrame появится объект – это экземпляр MFC-класса CSplitterWnd, порожденного от класса CWnd. Он служит для создания и управления разделенным окном. Кроме того, появится виртуальная функция OnCreateClient как фунукция-член класса CMainFrame. По умолчанию она выглядит следующим образом.

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
CCreateContext* pContext)
{   return m_wndSplitter.Create( this,              // родительское окно разделенного окна
                                 1, 2,              // максимальное число строк и столбцов
                                 CSize(10, 10), // минимальный размер окна представления
                                 pContext);     // информация о контексте устройства
}
```

Второй параметр (значение 1) показывает, что пользователь не может делить окно горизонтальной разделительной линией (следовательно, вешка горизонтальной разбивки в окне не появится). Значение 2 (третий параметр) показывает, что можно делить окно на левую и правую панель. Второй, третий и четвертый параметры можно корректировать вручную для изменения максимального количества окон представления и минимального размера такого окна.

Каждая из панелей управляется отдельным объектом представления, т.е. отдельным экземпляром класса CrazdelView (предполагается, что будущее приложение называется razdel). MFC автоматически вызывает соответствующую функцию OnDraw для перерисовки панели при каждом изменении данных в окне (например, пользователь расширил окно или удалил перекрывающее окно).

Покажем вначале пример программы, которая по данным читаемым из файла, разделяет окно на четыре части и распределяет прочитанную информацию по этим четырем панелям представления. Объем информации в каждой из панелей не большой и поэтому прокрутка панелей не используется.

Файл, представляющий собой поквартальный план за год и фактическое выполнение его по отраслям промышленности, информация из которого будет использоваться, имеет следующий формат.

```
Вид расхода          План Факт
```

```
Тяжелая промышленность 50 48 50 51 50 51 50 52
Легкая промышленность  40 40 38 36 40 42 38 38
Сфера обслуживания     30 32 30 30 30 34 30 32
Образование            10  8 10  9 10 11 10  9
Наука                  20 18 20 20 20 21 20 19
Оборона                30 28 30 31 30 31 30 30
```

После прочтения этого файла и разделения окна на четыре равные панели окно программы должно иметь вид, представленный на рис 2.28.

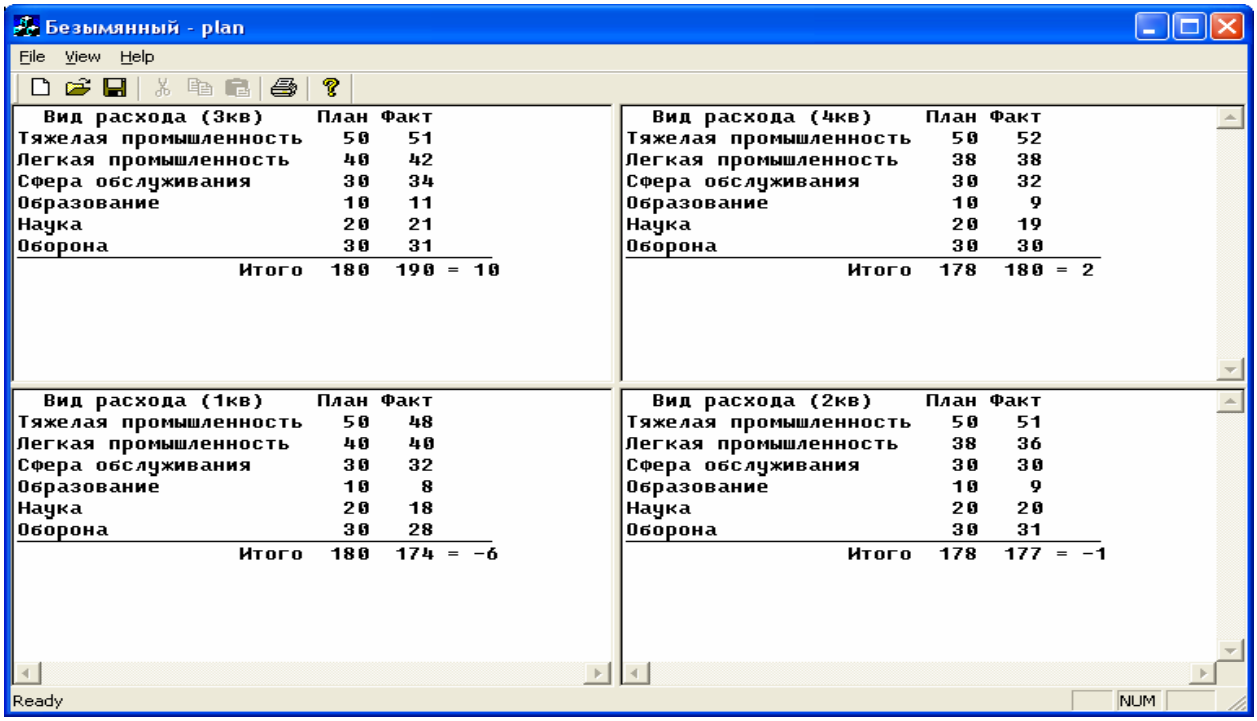


Рис. 2.28. Вид окна приложения при разбивке его на четыре равные панели.

Далее приводится текст только тех файлов это этой программы, в котором жирным текстом выделены добавления и изменения, внесенные в заготовку, созданную MFC AppWizard.

```
// MainFrm.cpp : implementation of the CMainFrame class
#include "stdafx.h"
#include "plan.h"
#include "MainFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////
///////// CMainFrame
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
```

```
    ON_WM_CREATE()
END_MESSAGE_MAP()
static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
//////////////////////////////////////////////////////
///////// CMainFrame construction/destruction
CMainFrame::CMainFrame()
{
    // TODO: add member initialization code here
}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
  if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD
|
      WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER |
CBRS_TOOLTIPS |
      CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
     !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
  { TRACE0("Failed to create toolbar\n");
    return -1;      // fail to create
  }
  if (!m_wndStatusBar.Create(this) ||
     !m_wndStatusBar.SetIndicators(indicators,
     sizeof(indicators)/sizeof(UINT)))
  { TRACE0("Failed to create status bar\n");
    return -1;      // fail to create
  }
// TODO: Delete these three lines if you don't want the
toolbar
// to be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);
    return 0;
}
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs,
    CCreateContext* pContext)
```

```
{   CDC *cdc;
    cdc=GetDC();
    cdc->SelectStockObject(SYSTEM_FIXED_FONT);
    extern TEXTMETRIC tm;
    cdc->GetTextMetrics(&tm);
    extern int ks;
    extern char naz[4][8][40];
    lpcs->cx=lstrlen(naz[0][0])*tm.tmAveCharWidth;
    lpcs->cy=ks*tm.tmHeight;
    return m_wndSplitter.Create(this,
        2, 2,       // TODO: adjust the number of rows,
columns
        CSize(10, 10),   // TODO: adjust the minimum
pane size
        pContext);
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: Modify the Window class or styles here by
modifying
    //  the CREATESTRUCT cs
    return TRUE;
}
/////////////////////////////////////////////////////////
/////// // CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{ CFrameWnd::AssertValid(); }
void CMainFrame::Dump(CDumpContext& dc) const
{ CFrameWnd::Dump(dc); }
#endif //_DEBUG
/////////////////////////////////////////////////////////
///////// CMainFrame message handlers


// planView.cpp : implementation of the CPlanView class
#include "stdafx.h"
#include "plan.h"
#include "planDoc.h"
#include "planView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////
///////// CPlanView
IMPLEMENT_DYNCREATE(CPlanView, CScrollView)
BEGIN_MESSAGE_MAP(CPlanView, CScrollView)
    //{{AFX_MSG_MAP(CPlanView)
    ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT,
CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW,
CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////
///////// CPlanView construction/destruction
CPlanView::CPlanView()
{
    // TODO: add construction code here
}
CPlanView::~CPlanView()
{
}
BOOL CPlanView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by
modifying
    //  the CREATESTRUCT cs
    return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////
///////// CPlanView drawing
char naz[4][8][40];
int pl[7][4], ft[7][4],ks,op=0,rr=0,po=0,nr=0;
TEXTMETRIC tm;
char rabs[40]="                  Итого=";
void CPlanView::OnDraw(CDC* pDC)
{
  CPlanDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  // TODO: add draw code for native data here
  CClientDC ClientDC(this);
  if(!nr) { nr=1; rr=1; return; }
  ClientDC.SelectStockObject(SYSTEM_FIXED_FONT);
  pDC->GetTextMetrics(&tm);
  if(rr==1)
  { for(int i=0;i<ks;i++)
```

```
    ClientDC.TextOut(2,tm.tmHeight*i,naz[0][i],
                     lstrlen(naz[0][i]));
    rr=2;
    ClientDC.MoveTo(2,tm.tmHeight*ks);
    ClientDC.LineTo(42*tm.tmAveCharWidth,tm.tmHeight*ks);
    ClientDC.TextOut(2,tm.tmHeight*ks+2,naz[0][ks],
                     lstrlen(naz[0][ks]));
  }
  else if(rr==2)
  { for(int i=0;i<ks;i++)
    ClientDC.TextOut(2,tm.tmHeight*i,naz[1][i],
                     lstrlen(naz[1][i]));
    rr=3;
    ClientDC.MoveTo(2,tm.tmHeight*ks);
    ClientDC.LineTo(42*tm.tmAveCharWidth,tm.tmHeight*ks);
    ClientDC.TextOut(2,tm.tmHeight*ks+2,naz[1][ks],
                     lstrlen(naz[1][ks]));
  }
    else if(rr==3)
    { for(int i=0;i<ks;i++)
      ClientDC.TextOut(2,tm.tmHeight*i,naz[2][i],
                       lstrlen(naz[2][i]));
      rr=4;
      ClientDC.MoveTo(2,tm.tmHeight*ks);

ClientDC.LineTo(42*tm.tmAveCharWidth,tm.tmHeight*ks);
      ClientDC.TextOut(2,tm.tmHeight*ks+2,naz[2][ks],
                       lstrlen(naz[2][ks]));
    }
    else if(rr==4)
    { for(int i=0;i<ks;i++)
      ClientDC.TextOut(2,tm.tmHeight*i,naz[3][i],
                       lstrlen(naz[3][i]));
      rr=1;
      ClientDC.MoveTo(2,tm.tmHeight*ks);

ClientDC.LineTo(42*tm.tmAveCharWidth,tm.tmHeight*ks);
      ClientDC.TextOut(2,tm.tmHeight*ks+2,naz[3][ks],
                       lstrlen(naz[3][ks]));
    }
}
void CPlanView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
    // TODO: calculate the total size of this view
}
```

```
/////////////////////////////////////////////////////////////
///////// CPlanView printing
BOOL CPlanView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}
void CPlanView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
    // TODO: add extra initialization before printing
}
void CPlanView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo*
/*pInfo*/)
{
    // TODO: add cleanup after printing
}
/////////////////////////////////////////////////////////////
///////// CPlanView diagnostics
#ifdef _DEBUG
void CPlanView::AssertValid() const
{
    CScrollView::AssertValid();
}
void CPlanView::Dump(CDumpContext& dc) const
{ CScrollView::Dump(dc);
}

CPlanDoc* CPlanView::GetDocument() // non-debug version
is
                                   // inline
{
    ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CPlanDoc)));
    return (CPlanDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////
///////// CPlanView message handlers
char szDirName[80]="c:\\",
     szFile[80]="D:\\Bvn\\CPP\\Release\\graf.txt";
void CPlanView::OnFileOpen()
{
    // TODO: Add your command handler code here
  GetCurrentDirectory(78,szDirName);
  OPENFILENAME ofn;
  char szFileTitle[80];
```

```
  char  szFilter[256]="TXT Files(*.TXT)|*.TXT|All
Files(*.*) |*.*|";
  char rab[80];
  if(op)
  { MessageBox("Файл уже прочитан!",szFile);
    return;
  }
  CDC *cdc=GetDC();
  CWnd *cw=cdc->GetWindow();
  szFile[0] = '\0';
  for(int i = 0;szFilter[i]!='\0';i++)
  { if(szFilter[i] == '|') szFilter[i]='\0'; }
  memset(&ofn, 0, sizeof(OPENFILENAME));
  ofn.lStructSize = sizeof(OPENFILENAME);
  ofn.hwndOwner = cw->m_hWnd;
  ofn.lpstrFilter = szFilter;
  ofn.nFilterIndex = 1;
  ofn.lpstrFile= szFile;
  ofn.nMaxFile = sizeof(szFile);
  ofn.lpstrFileTitle = szFileTitle;
  ofn.nMaxFileTitle = sizeof(szFileTitle);
  ofn.lpstrInitialDir = szDirName;
  ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
  if(!GetOpenFileName(&ofn)) return;
  FILE *inFile;
  if((inFile=fopen(szFile,"r"))==NULL)
  { MessageBox("Ошибка открытия файла %s!",szFile);
    return;
  }
  cdc->GetTextMetrics(&tm);
  ks=1;
  fgets(naz[0][0],80,inFile);
  *(naz[0][0]+lstrlen(naz[0][0])-1)='\0';
  lstrcpy(naz[0][0]+14,"(1кв)");
  *(naz[0][0]+19)=' ';
  char c;
  for(i=1,c='2';i<4;i++,c++)
  { lstrcpy(naz[i][0],naz[0][0]); *(naz[i][0]+15)=c; }
  int pls[4]={0,0,0,0}, fts[4]={0,0,0,0};
  while(!feof(inFile))
  { if(fgets(rab,80,inFile)==NULL) break;
    rab[lstrlen(rab)-1]='\0';
    cdc->TextOut(0,ks*tm.tmHeight,rab,lstrlen(rab));
    strncpy(naz[0][ks],rab,22); *(naz[0][ks]+22)='\0';
    for(i=1;i<4;i++) lstrcpy(naz[i][ks],naz[0][ks]);
```

```
    sscanf(rab+23,"%d%d%d%d%d%d%d%d",&pl[0][0],&ft[0][0],
           &pl[0][1],&ft[0][1],&pl[0][2],&ft[0][2],
           &pl[0][3],&ft[0][3]);
    for(i=0;i<4;i++) { pls[i]+=pl[0][i];
fts[i]+=ft[0][i]; }
    for(int k=0;k<4;k++) wsprintf(naz[k][ks]+22,
                                  "%5d
%4d",pl[0][k],ft[0][k]);
    ks++;
  }
  for(i=0;i<4;i++)
  { wsprintf(rabs+22,"%5d %4d =%d",pls[i],fts[i],fts[i]-
pls[i]);
    lstrcpy(naz[i][ks],rabs);
  }
  fclose(inFile);
  op=1; rr=1; nr=0;
  MessageBox("Файл прочитан.\nДля поквартального пред-
ставления\n разделите окно на четыре панели.",szFile);
}
```

При разделении окна, следует особо обратить внимание на изменение содержимого хотя бы одной из панелей представления. Если одна из панелей была изменена (было что-то добавлено или изменился ее размер), то другая панель должна быть полностью перерисована, включая даже те части, которые в данный момент не видны.

Покажем на примере рисования линий, как осуществляется эффективная перерисовка.

Определим класс Cline в файле razdelDoc.h:

```
Class Cline : public CObject
{
  protected:
    int m_X1,m_Y1,m_X2,m_Y2,m_L,m_XK,m_GR;
  public:
    Cline(int X1, int Y1, int X2, int Y2)
    {  m_X1=X1; m_Y1=Y1; m_X2=X2; m_Y2=Y2; }
    void Draw(*PDC);
    CRect GetDimRect();
}
```

Теперь в том же файле в класс документа введем определение функции AddLine и GetLine, GetNumLines :

```
CLine *AddLine(int X1,int Y1,int X2,int Y2);
CLine *GetLine(int Index);
```

Теперь в файле razdelDoc.cpp определите эти функции:

```
CLine *CrazdelDoc::AddLine(int X1,int Y1,int X2,int Y2)
{ CLine *PLine=new CLine(X1,Y1,X2,Y2,S,L,K,G);
```

```
  m_LineArray.Add(PLine);
  SetModifiedFlag();
  return PLine;
}

CLine *CrazdelDoc::GetLine(int Index)
{ if(Index<0 || Index>m_LineArray.GetUpperBound())
   return 0;
 return m_LineArray.GetAt(Index);
}

int CrazdelDoc::GetNumLines()
{ return m_LineArray.GetSize(); }
```

Кроме того в файле razdelDoc.cpp определяются еще две функции.

```
CRect CLine::GetDimRect()
{ return CRect( min(m_X1,m_X2),min(m_Y1,m_Y2),
                max(m_X1,m_X2)+1,max(m_Y1,m_Y2)+1); }
```

Функция GetDimRect возвращает объект класса CRect с размерами ограничевающего линию прямоугольника. Макрокоманды min и max (определенные в Windows.h) используются для проверки его сторон (координата левой стороны по горизонтали должна быть меньше правой, координата нижней стороны по вертикали – меньше верхней, в противном случае прямоугольник интерпритируется как пустой).

```
void CLine::Draw(CDC *PDC)
{ PDC->MoveTo(m_X1, m_Y1);
  PDC->LineTo(m_X2, mY2);
```

Теперь для отрисовки линии в функции OnDraw класса CRazdelView достаточно выполнить следующую последовательность команд.

```
CClientDC ClientDC(this);  // создается объект контекста устройства окна представления
CLine *PCLine;
ClientDC.MoveTo(x1, y1);
ClientDC.LineTo(x2, y2);
PCLine=pDoc->AddLine(x1, y1, x2, y2);
pDoc->UpdateAllViews(this,0,PCLine);
```

Кроме того, функция OnDraw обновляет весь рисунок, даже если только часть его попадает в недействительную область окна представления. Чтобы функция OnDraw перерисовывала только область ставшую недействительной она должна содержать следующий код.

```
void  Crazdel View::OnDraw(CDC* pDC)
{  CrazdelDoc* pDoc=GetDocument();
```

```
    ASSERT_VALID(pDoc);
   . . .
   CRect ClipRect, CRect DimRect, CLine *PLine, CRect IntRect;
   pDC->GetClipBox(&ClipRect);
   int Index=pDoc->GetNumLines();
   while(Index--)
   {  PLine=pDoc->GetLine(Index);
      DimRect=PLine->GetDimRect();
      if(IntRect.IntersectRect(DimRect, ClipRect))
         PLine->Draw(pDC);
   }
}
```

Функция CDC::GetClipBox используется для получения размеров недействительной области, а функция GetDimRect для получения ограничевающего прямоугольника. Функция IntersectRect определяет попадает ли ограничивающий прямоугольник в область, признанную недействительной. Последняя функция возвращает значение TRUE, если прямоугольники имеют не пустое пересечение. В этом случае функция Draw рисует линию.

В качестве примера приводится текст программы, выводящий линейный график по координатам, которые содержатся в текстовом файле. Вид окна соответствующего приложения может иметь вид, представленный на рис. 2.30.

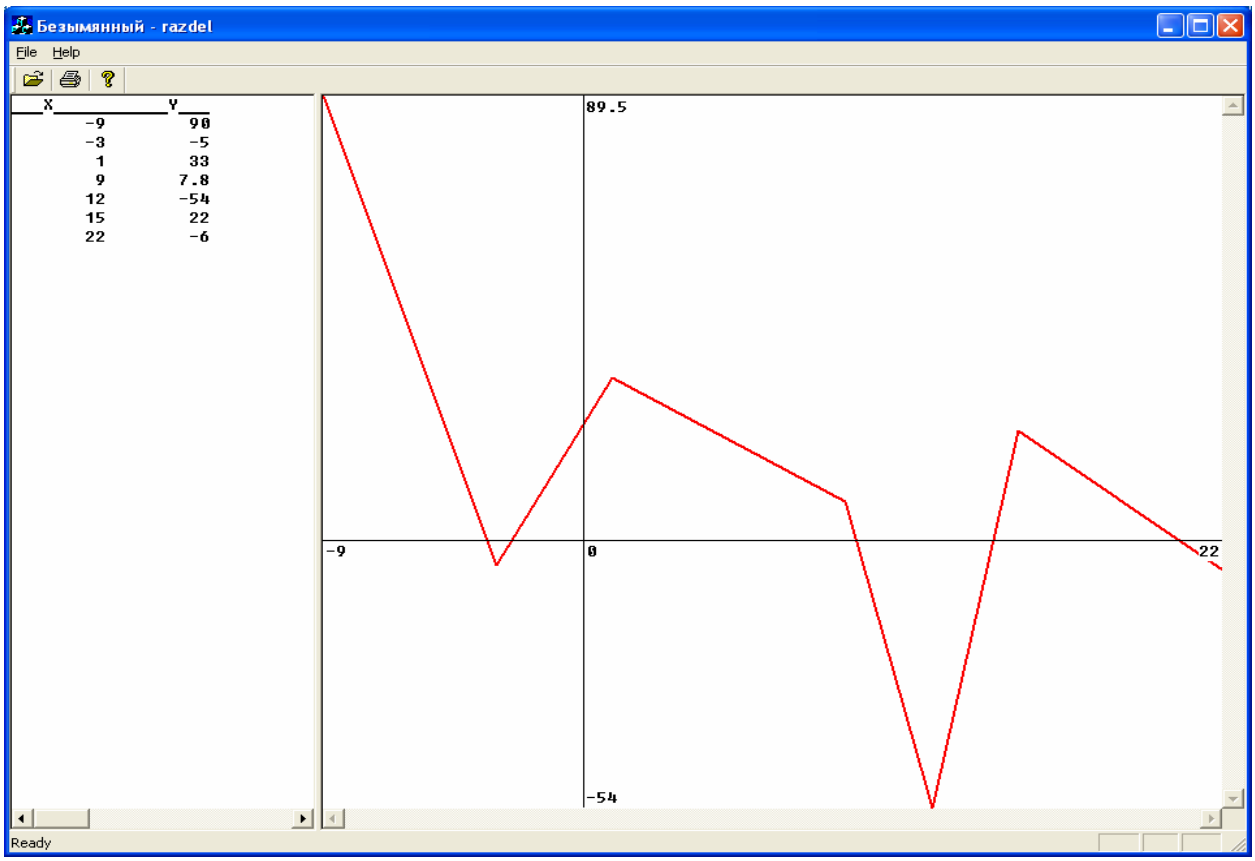


Рис. 2.30. Вид окна разбитого на два окна представления, одно из которых содержит текст, а другое соответствующий график.

Следует обратить внимание, что в качестве добавляемых объектов в окна представления используются не только линии, но и текст, который трактуется как линия соответствующего размера. Кроме того, при изменении размеров панели представления графика, происходит соответствующее масштабирование выводимого представления.

При предварительном просмотре вида печати и выводе на печать (выводится только график) устанавливается отступ в 15 миллиметров от краев листа.

В следующем далее примере, выведены только файлы, подвергшиеся корректировке. Жирным шрифтом выделены добавления и исправления в стандартном построении MFC AppWizard.

```
// razdelDoc.h : interface of the CRazdelDoc class
#if
!defined(AFX_RAZDELDOC_H__136DBFA2_BC52_4C49_B2A0_7A2B859D74F
D__INCLUDED_)
#define
AFX_RAZDELDOC_H__136DBFA2_BC52_4C49_B2A0_7A2B859D74FD__INC
LUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CLine : public CObject
{
   protected:
   int m_X1,m_Y1,m_X2,m_Y2,m_L,m_XK,m_GR;
        char* m_S;
   public:
        CLine(int X1,int Y1,int X2,int Y2,char *S,int L,int XK,int GR)
        { m_X1=X1; m_Y1=Y1; m_XK=XK; m_GR=GR;
          if(!L) { m_X2=X2; m_Y2=Y2; m_L=0; }
          else if(L)
          { m_S=new char[L+1]; m_L=L; lstrcpy(m_S,S);
            m_X2=X2; m_Y2=Y2;
          }
        }
        ~CLine() { if(m_L) delete []m_S; }
        void Draw(CDC *PDC);
        CRect GetDimRect();
        int getL() { return m_L; }
};
class CRazdelDoc : public CDocument
{
  protected:
        CTypedPtrArray<CObArray, CLine*> m_LineArray;
```

```
  public:
        CLine *AddLine(int X1,int Y1,int X2,int Y2,char *S,int L,int K,int G);
        CLine *GetLine(int Index);
        int GetNumLines();
 protected: // create from serialization only
        CRazdelDoc();
        DECLARE_DYNCREATE(CRazdelDoc)
// Attributes
public:
// Operations
public:
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CRazdelDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL
// Implementation
public:
        virtual ~CRazdelDoc();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
        //{{AFX_MSG(CRazdelDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.
#endif //
!defined(AFX_RAZDELDOC_H__136DBFA2_BC52_4C49_B2A0_7A2B859D74F
D__INCLUDED_)

// razdelDoc.cpp : implementation of the CRazdelDoc class
//
#include "stdafx.h"
#include "razdel.h"
#include "razdelDoc.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CRazdelDoc
IMPLEMENT_DYNCREATE(CRazdelDoc, CDocument)

BEGIN_MESSAGE_MAP(CRazdelDoc, CDocument)
	//{{AFX_MSG_MAP(CRazdelDoc)
	// NOTE - the ClassWizard will add and remove mapping macros here.
	//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CRazdelDoc construction/destruction
CRazdelDoc::CRazdelDoc()
{
	// TODO: add one-time construction code here
}
CRazdelDoc::~CRazdelDoc()
{
}
BOOL CRazdelDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;
	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)
	return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CRazdelDoc serialization
void CRazdelDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: add storing code here
	}
	else
	{
		// TODO: add loading code here
	}
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CRazdelDoc diagnostics
#ifdef _DEBUG
void CRazdelDoc::AssertValid() const
{
        CDocument::AssertValid();
}
void CRazdelDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CRazdelDoc commands
#include <math.h>
HPEN hp,hps;
void CLine::Draw(CDC *PDC)
{ extern RECT rco,rct;
  extern double xh,yh;
  extern double xm[10],ym[10],ymi,yma,xmi;
  extern int ks;
  extern TEXTMETRIC tm;
  if(!getL())
  { if(m_XK)
    { if(rct.right>=rco.right)
      { if(!m_GR)
          { PDC->MoveTo((int)(fabs(xmi)*xh),m_Y1);
            PDC->LineTo((int)(fabs(xmi)*xh),(int)(fabs(yma-ymi)*yh));
          }
        else
        { hp=CreatePen(PS_SOLID,2,RGB(250,0,0));
          hps=(HPEN)PDC->SelectObject(hp);
          PDC->MoveTo(0,(int)((yma-ym[1])*yh));
          for(int i=2;i<ks;i++) PDC->LineTo((int)((fabs(xm[1])+xm[i])*xh),
                                                    (int)(fabs(yma-ym[i])*yh));
          PDC->SelectObject(hps);
          DeleteObject(hp);
        }
      }
      else
      { if(m_GR)
        { hp=CreatePen(PS_SOLID,2,RGB(250,0,0));
          hps=(HPEN)PDC->SelectObject(hp);
        }
        PDC->MoveTo(m_X1,m_Y1);
        PDC->LineTo(m_X2,m_Y2);
```

```
    if(m_GR) { PDC->SelectObject(hps);  DeleteObject(hp); }
   }
  }
  else { if(rct.right>=rco.right)
         { PDC->MoveTo(m_X1,(int)(yma*yh));
           PDC->LineTo(m_X2+rct.right-rco.right,(int)(fabs(yma)*yh));
          }
         else
         { PDC->MoveTo(m_X1,m_Y1);
           PDC->LineTo(m_X2,m_Y2);
          }
        }
}
else
{ PDC->SelectStockObject(SYSTEM_FIXED_FONT);
  extern TEXTMETRIC tm;
  PDC->GetTextMetrics(&tm);
  extern char vvod[10][40];
  if(m_X1==2) PDC->TextOut(m_X1,(int)(yma*yh+2),m_S,m_L);
  else if(!m_GR)
          for(int i=0;i<ks;i++)
           PDC->TextOut(m_X1,m_Y1+tm.tmHeight*i,vvod[i],m_L);
  else if(m_GR==1)
       { if(m_Y1<5) PDC->TextOut((int)(fabs(xmi)*xh+2),m_Y1,m_S,m_L);
         else if(m_S[0]=='0') PDC->TextOut((int)(fabs(xmi)*xh+2),
                                           (int)(yma*yh+2),m_S,m_L);
              else
              { char rab[10];
               SIZE szt;
               sprintf(rab,"%lg",xmi);
                GetTextExtentPoint(PDC->m_hDC,rab,lstrlen(rab),&szt);
               PDC->TextOut((int)(fabs(xmi)*xh+2),(int)(fabs(yma-ymi)*yh)-
                  szt.cy,m_S,m_L);
              }
          }
         else
        { char rab[10];
          extern double xma;
          sprintf(rab,"%lg",xma);
          SIZE szt;
          extern int xmap;
          int sr;
          if(xmap>rct.right) sr=xmap; else sr=rct.right;
          GetTextExtentPoint(PDC->m_hDC,rab,lstrlen(rab),&szt);
          PDC->TextOut(sr-2-szt.cx,(int)(yma*yh+2),m_S,m_L);
        }
```

```
   }
}

CLine *CRazdelDoc::AddLine(int X1,int Y1,int X2,int Y2,char *S,int L,
                                        int K,int G)
{ CLine *PLine=new CLine(X1,Y1,X2,Y2,S,L,K,G);
  m_LineArray.Add(PLine);
  SetModifiedFlag();
  return PLine;
}

CLine *CRazdelDoc::GetLine(int Index)
{ if(Index<0 || Index>m_LineArray.GetUpperBound())
     return 0;
  return m_LineArray.GetAt(Index);
}

int CRazdelDoc::GetNumLines()
{ return m_LineArray.GetSize(); }

CRect CLine::GetDimRect()
{ return CRect( min(m_X1,m_X2),min(m_Y1,m_Y2),max(m_X1,m_X2)+1,
              max(m_Y1,m_Y2)+1);
}

// razdelView.cpp : implementation of the CRazdelView class
//
#include "stdafx.h"
#include "razdel.h"
#include "razdelDoc.h"
#include "razdelView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CRazdelView
IMPLEMENT_DYNCREATE(CRazdelView, CScrollView)
BEGIN_MESSAGE_MAP(CRazdelView, CScrollView)
        //{{AFX_MSG_MAP(CRazdelView)
        ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
```

```
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW,
CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CRazdelView construction/destruction
CRazdelView::CRazdelView()
{
        // TODO: add construction code here
}
CRazdelView::~CRazdelView()
{
}
BOOL CRazdelView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
        return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CRazdelView drawing
#include <math.h>
int rr=0,ksd=0,kdst;
HWND hw1=0,hw2=0;
RECT rco,rct;
double xh;
double yh;
double yma,ymi,xma,xmi;
int xmip,xmap;
void CRazdelView::OnDraw(CDC* pDC)
{
        CRazdelDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
         extern int ks,op,print;
         extern char vvod[10][40];
        extern double xm[10],ym[10];
        if(!rr) { rr=1; return; }
        CClientDC ClientDC(this);
        CLine *PCLine;
        CPoint cp;
        SIZE sz;
        CWnd *cw;
        if(rr==1 && !print)
        {
          cw=ClientDC.GetWindow();
```

```
  hw1=cw->m_hWnd;
  extern TEXTMETRIC tm;
  ClientDC.SelectStockObject(SYSTEM_FIXED_FONT);
  pDC->GetTextMetrics(&tm);
  for(int i=0;i<ks;i++)
  { ClientDC.TextOut(0,tm.tmHeight*i,vvod[i],lstrlen(vvod[i]));
     if(i==0)
     { GetTextExtentPoint(pDC->m_hDC,vvod[i],lstrlen(vvod[i]),&sz);
        PCLine=pDoc->AddLine(0,tm.tmHeight*i,sz.cx,sz.cy+tm.tmHeight*i,
                                          vvod[i],lstrlen(vvod[i]),0,0);
       pDoc->UpdateAllViews(this,0,PCLine);
       ksd++; kdst=1;
      }
  }
   if(op==1) op=2; else rr=2;
}
else if(rr==2)
{ if(op!=2) return;
  cw=ClientDC.GetWindow();
  cw->GetClientRect(&rco);
  hw2=cw->m_hWnd;
  cw->GetScrollRange(SB_HORZ,&xmip,&xmap);
  yma=ymi=ym[1];
  xma=xmi=xm[1];
  for(int i=2;i<ks;i++)
  { if(yma<ym[i]) yma=ym[i];
    if(ymi>ym[i]) ymi=ym[i];
    if(xma<xm[i]) xma=xm[i];
    if(xmi>xm[i]) xmi=xm[i];
  }
  yh=rco.bottom/fabs(yma-ymi);
  xh=xmap/fabs(xma-xmi);
  ClientDC.MoveTo(0,(int)(yma*yh));
  ClientDC.LineTo(xmap,(int)(yma*yh));
  PCLine=pDoc->AddLine(0,(int)(yma*yh),xmap,(int)(yma*yh),0,0,0,0);
  pDoc->UpdateAllViews(this,0,PCLine);
  ksd++; kdst++;
  ClientDC.MoveTo(int(fabs(xmi)*xh),0);
  ClientDC.LineTo(int(fabs(xmi)*xh),rco.bottom);
  PCLine=pDoc->AddLine((int)(fabs(xmi)*xh),0,(int)(fabs(xmi)*xh),
                                      rco.bottom,0,0,1,0);
  pDoc->UpdateAllViews(this,0,PCLine);
  ksd++; kdst++;
  ClientDC.TextOut((int)(fabs(xm[1]*xh))+2,(int)(yma*yh)+2,"0",1);
  GetTextExtentPoint(pDC->m_hDC,"0",1,&sz);
```

```
        PCLine=pDoc->AddLine((int)(fabs(xm[1]*xh))+2,(int)(yma*yh)+2,

sz.cx+(int)(fabs(xm[1]*xh))+2,sz.cy+(int)(yma*yh)+2,"0",1,0,1);
        pDoc->UpdateAllViews(this,0,PCLine);
        ksd++; kdst++;
        char rab[10];
        sprintf(rab,"%lg",xm[1]);
        ClientDC.TextOut(2,(int)(yma*yh)+2,rab,lstrlen(rab));
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        PCLine=pDoc->AddLine(2,(int)(yma*yh)+2,sz.cx+2,
                                    sz.cy+(int)(yma*yh)+2,rab,lstrlen(rab),0,1);
        pDoc->UpdateAllViews(this,0,PCLine);
        ksd++; kdst++;
        sprintf(rab,"%lg",xm[ks-1]);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        ClientDC.TextOut((int)(fabs(xm[ks-1]-xm[1])*xh)-2-sz.cx,(int)(yma*yh)+2,
                            rab,lstrlen(rab));
        PCLine=pDoc->AddLine((int)(fabs(xm[ks-1]-xm[1])*xh)-2-sz.cx,
                                    (int)(yma*yh)+2,(int)(fabs(xm[ks-1]-xm[1])*xh)-
2,
                                    sz.cy+(int)(yma*yh)+2,rab,lstrlen(rab),0,2);
        pDoc->UpdateAllViews(this,0,PCLine);
        ksd++; kdst++;
        sprintf(rab,"%lg",yma);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        ClientDC.TextOut((int)(fabs(xm[1]*xh))+2,2,rab,lstrlen(rab));
        PCLine=pDoc-
>AddLine((int)(fabs(xm[1]*xh))+2,2,sz.cx+(int)(fabs(xm[1]*xh))+2,
                                    sz.cy+2,rab,lstrlen(rab),0,1);
        pDoc->UpdateAllViews(this,0,PCLine);
        ksd++; kdst++;
        sprintf(rab,"%lg",ymi);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        ClientDC.TextOut((int)(fabs(xm[1]*xh))+2,rco.bottom-sz.cy,rab,lstrlen(rab));
        PCLine=pDoc->AddLine((int)(fabs(xm[1]*xh))+2,rco.bottom-sz.cy,

sz.cx+(int)(fabs(xm[1]*xh))+2,rco.bottom,rab,lstrlen(rab),0,1);
        pDoc->UpdateAllViews(this,0,PCLine);
        ksd++; kdst++;
        extern HPEN hp,hps;
        hp=CreatePen(PS_SOLID,2,RGB(250,0,0));
        hps=(HPEN)ClientDC.SelectObject(hp);
        ClientDC.MoveTo(0,(int)((yma-ym[1])*yh));
        for(i=2;i<ks;i++)
        { ClientDC.LineTo((int)((fabs(xm[1])+xm[i])*xh),(int)((yma-ym[i])*yh));
```

```
          PCLine=pDoc->AddLine((int)((fabs(xm[1])+xm[i-1])*xh),(int)((yma-ym[i-
1])*yh),
                                        (int)((fabs(xm[1])+xm[i])*xh),(int)((yma-
ym[i])*yh),0,0,1,i);
           ksd++; kdst++;
         }
          ClientDC.SelectObject(hps);
          DeleteObject(hp);
        rr=3;
       }
       else if(!print)
        { CRect ClipRect;
         CRect DimRect;
         CRect IntRect;
         CLine *PLine;
         pDC->GetClipBox(&ClipRect);
         CWnd *cw;
         cw=pDC->GetWindow();
         if(cw->m_hWnd==hw1) rr=4;
         else if(cw->m_hWnd==hw2) rr=5;
         int Index=pDoc->GetNumLines();
         while(Index--)
         { PLine=pDoc->GetLine(Index);
          DimRect=PLine->GetDimRect();
          if(IntRect.IntersectRect(DimRect, ClipRect))
          { if(Index==ksd-kdst && rr==4) PLine->Draw(pDC);
            else if(Index>ksd-kdst && rr==5)
                { cw->GetClientRect(&rct);
                  cw->GetScrollRange(SB_HORZ,&xmip,&xmap);
                 if(xmap>rct.right) xh=xmap/fabs(xma-xmi);
                    else xh=rct.right/fabs(xma-xmi);
                  if(rco.bottom<=rct.bottom) yh=rct.bottom/fabs(yma-ymi);
                  else yh=rco.bottom/fabs(yma-ymi);
                  PLine->Draw(pDC);
                }
                if(Index<ksd-kdst) break;
           }
          }
       }
       if(print)
       { int rx,ry,rxm,rym;
         rx=pDC->GetDeviceCaps(PHYSICALWIDTH);
         ry=pDC->GetDeviceCaps(PHYSICALHEIGHT);
          rxm=pDC->GetDeviceCaps(HORZSIZE);
         rym=pDC->GetDeviceCaps(VERTSIZE);
         double xhm,yhm;
```

```
        xhm=rx/rxm;
        yhm=ry/rym;
        yh=(ry-30*yhm)/fabs(yma-ymi);
        xh=(rx-30*xhm)/fabs(xma-xmi);
        pDC->MoveTo(int(15*xhm),(int)(yma*yh+15*yhm));
        pDC->LineTo((int)(rx-1-15*xhm),(int)(yma*yh+15*yhm));
        pDC->MoveTo(int(fabs(xmi)*xh+15*xhm),(int)(15*yhm));
        pDC->LineTo(int(fabs(xmi)*xh+15*xhm),(int)(ry-1-15*yhm));
        pDC->TextOut((int)(fabs(xm[1]*xh)+15*xhm+2*xhm),
                        (int)(fabs(yma*yh)+yhm+15*yhm),"0",1);
        char rab[10];
        sprintf(rab,"%lg",xm[1]);
        pDC->TextOut((int)(xhm+15*xhm),(int)(fabs(yma*yh)+yhm+15*yhm),
                        rab,lstrlen(rab));
        sprintf(rab,"%lg",xm[ks-1]);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        pDC->TextOut((int)(fabs(xm[ks-1]-xm[1])*xh-2-sz.cx+15*xhm),
                        (int)(fabs(yma*yh)+yhm+15*yhm),rab,lstrlen(rab));
        sprintf(rab,"%lg",yma);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        pDC->TextOut((int)(fabs(xm[1]*xh)+2*xhm+15*xhm),(int)(yhm+15*yhm),
                        rab,lstrlen(rab));
        sprintf(rab,"%lg",ymi);
        GetTextExtentPoint(pDC->m_hDC,rab,lstrlen(rab),&sz);
        pDC->TextOut((int)(fabs(xm[1]*xh)+2*xhm+15*xhm),(int)(ry-sz.cy-
15*yhm),
                        rab,lstrlen(rab));
        extern HPEN hp,hps;
        hp=CreatePen(PS_SOLID,2,RGB(250,0,0));
        hps=(HPEN)pDC->SelectObject(hp);
        pDC->MoveTo((int)(15*xhm),(int)((yma-ym[1])*yh+15*yhm));
        for(int i=2;i<ks;i++)
           pDC->LineTo((int)((fabs(xm[1])+xm[i])*xh+15*xhm),
                          (int)((yma-ym[i])*yh+15*yhm));
         pDC->SelectObject(hps);
         DeleteObject(hp);
        print=0; return;
       }
}
TEXTMETRIC tm;
int razx;
void CRazdelView::OnInitialUpdate()
{ CScrollView::OnInitialUpdate();

       CSize sizeTotal;
       // TODO: calculate the total size of this view
```

```
	RECT rc;
	CDC *cdc=GetDC();
	cdc->GetTextMetrics(&tm);
	GetClientRect(&rc);
	razx=sizeTotal.cx = rc.right+19*tm.tmAveCharWidth;
	  sizeTotal.cy = rc.bottom;
	SetScrollSizes(MM_TEXT, sizeTotal);
}

/////////////////////////////////////////////////////////////////////////////
// CRazdelView printing
BOOL CRazdelView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}
void CRazdelView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add extra initialization before printing
}
void CRazdelView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add cleanup after printing
}
/////////////////////////////////////////////////////////////////////////////
// CRazdelView diagnostics
#ifdef _DEBUG
void CRazdelView::AssertValid() const
{
	CScrollView::AssertValid();
}
void CRazdelView::Dump(CDumpContext& dc) const
{
	CScrollView::Dump(dc);
}
CRazdelDoc* CRazdelView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CRazdelDoc)));
	return (CRazdelDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CRazdelView message handlers
void CRazdelView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
```

```
        // TODO: Add your specialized code here and/or call the base class
        if(pHint!=0)
        { CRect InvalidRect=((CLine *)pHint)->GetDimRect();
          CClientDC ClientDC(this);
          OnPrepareDC(&ClientDC);
          ClientDC.LPtoDP(&InvalidRect);
          InvalidateRect(&InvalidRect);
        }
        else CScrollView::OnUpdate(pSender,lHint,pHint);
}

#include "otkr.h"
char vvod[10][40];
int ks,op=0;
double xm[10],ym[10];
char szDirName[80]="c:\\",szFile[80]="D:\\Bvn\\CPP\\Release\\graf.txt";
void CRazdelView::OnFileOpen()
{
        // TODO: Add your command handler code here
  GetCurrentDirectory(78,szDirName);
  OPENFILENAME ofn;
  char szFileTitle[80];
  char  szFilter[256]="TXT Files(*.TXT)|*.TXT|All Files(*.*)|*.*|";
  Cotkr oto;
  RECT rcox;
  ::GetClientRect(hw1,&rcox);
  if(rcox.right<razx)
  { oto.DoModal(); return; }
  CDC *cdc=GetDC();
  CWnd *cw=cdc->GetWindow();
  szFile[0] = '\0';
  for(int i = 0;szFilter[i]!='\0';i++)
  { if(szFilter[i] == '|') szFilter[i]='\0'; }
  memset(&ofn, 0, sizeof(OPENFILENAME));
  ofn.lStructSize = sizeof(OPENFILENAME);
  ofn.hwndOwner = cw->m_hWnd;
  ofn.lpstrFilter = szFilter;
  ofn.nFilterIndex = 1;
  ofn.lpstrFile= szFile;
  ofn.nMaxFile = sizeof(szFile);
  ofn.lpstrFileTitle = szFileTitle;
  ofn.nMaxFileTitle = sizeof(szFileTitle);
  ofn.lpstrInitialDir = szDirName;
  ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
  if(!GetOpenFileName(&ofn)) return;
  FILE *inFile;
```

```
 if((inFile=fopen(szFile,"r"))==NULL)
 { MessageBox("Ошибка открытия файла!",szFile);
  return;
 }
 ks=1;
 lstrcpy(vvod[0],"___X_____Y___");
 while(!feof(inFile))
 { if(fscanf(inFile,"%lg%lg",&xm[ks],&ym[ks])==EOF) break;
  sprintf(vvod[ks],"%9.2lg %9.2lg",xm[ks],ym[ks]);
  ks++;
 }
 fclose(inFile);
 ::InvalidateRect(hw2,NULL,TRUE); ::InvalidateRect(hw1,NULL,TRUE);
 op=1; rr=1;
}

BOOL CRazdelView::OnScroll(UINT nScrollCode, UINT nPos, BOOL bDoScroll)
{
        // TODO: Add your specialized code here and/or call the base class
        if(hw2==0) return FALSE;
        return CScrollView::OnScroll(nScrollCode, nPos, bDoScroll);
}
```

2.4. Создание многодокументных приложений

В предыдущих параграфах мы создавали программы с *однодокументным интерфейсом*, называемые также SDI-приложениями (SDI – Single Document Interface). SDI-приложения предназначены для одновременного просмотра и возможно редактирования только одного документа. Программы с многодокументны интерфейсом, называемые MDI-приложениями (Multiple Document Interface) позволяют открывать одновременно несколько документов. Каждый из них просматривается и редактируется в отдельном дочернем окне, находящемся в рабочей области приложения внутри главного окна программы. Более того, при необходимости через общую память содержимое или часть одного документа может быть скопирована в другой документ.

Для создания MDI-приложения с использованием мастера AppWizard необходимо только на первом шаге создания соответствующего проекта вместо опции Single Documents выбрать опцию Multiple Documents.

В качестве примера, рассмотрим программу с MDI-интерфейсом, позволяющую одновременно по выбору соответствующего документа моделировать работу комбинационного элемента, триггера и логической схемы. Для определенности пусть это будут логический элемент И-НЕ, R-S-T триггер и логическая схема, приведенная на рис.2.31.

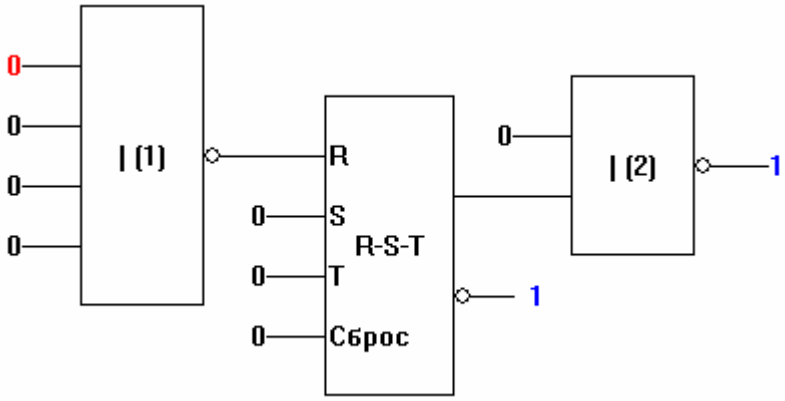


Рис. 2.31. Вид логической схемы, для моделирования ее работы.

Окно соответствующего приложения, в этом случае, может выглядеть следующим образом (рис. 2.32).

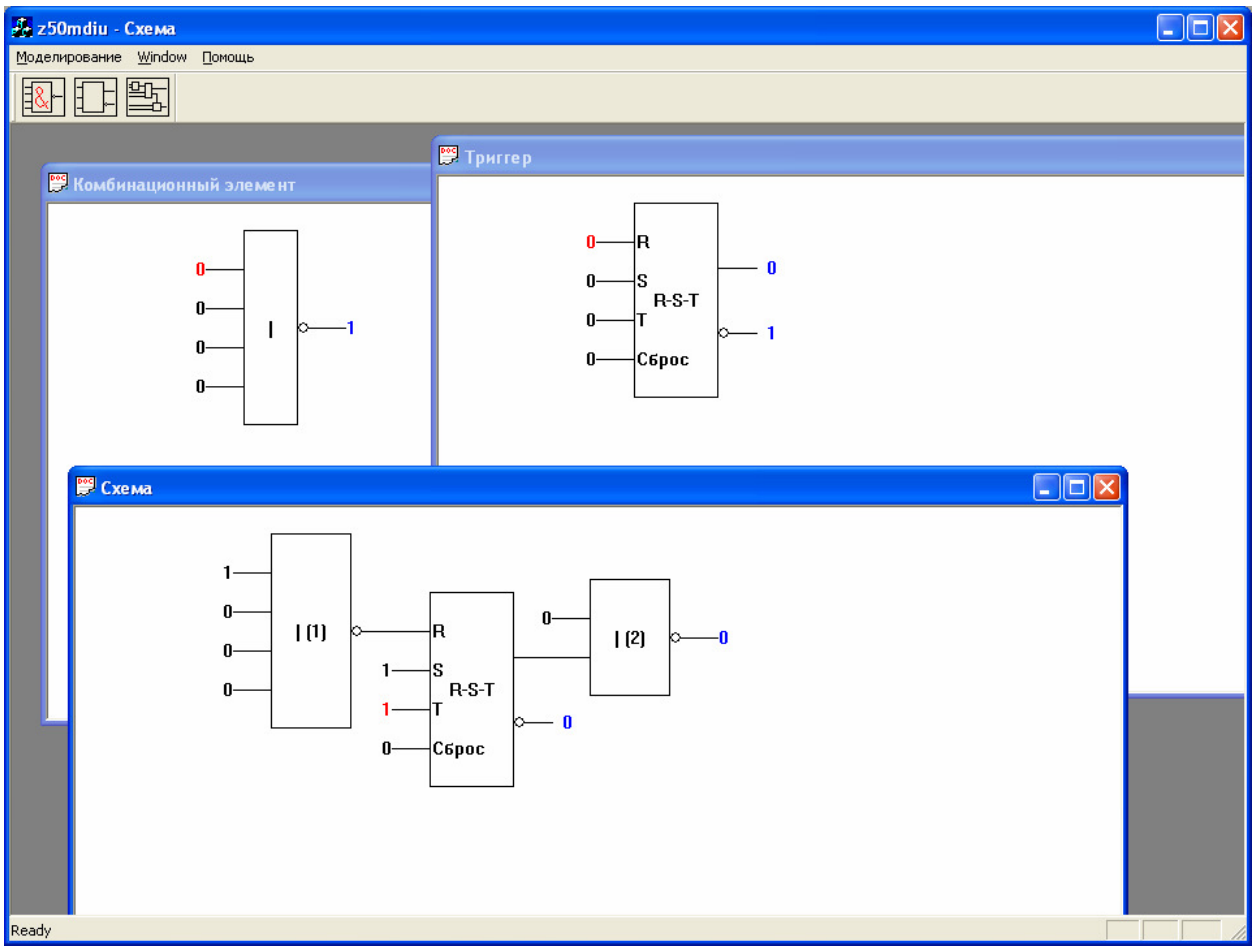


Рис. 2.32. Вид окна мультидокументного приложения.

```
// z50mdiuDoc.h : interface of the CZ50mdiuDoc class
/////////////////////////////////////////////////////////////
///////#if
!defined(AFX_Z50MDIUDOC_H__99E5ECCD_669D_11D7_8A00_82D38D
AAD863__INCLUDED_)
#define
AFX_Z50MDIUDOC_H__99E5ECCD_669D_11D7_8A00_82D38DAAD863__I
NCLUDED_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CZ50mdiuDoc : public CDocument
{
protected: // create from serialization only
    CZ50mdiuDoc();
    DECLARE_DYNCREATE(CZ50mdiuDoc)
// Attributes
public:
   int pe,pozt;
   void *ue;
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CZ50mdiuDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CZ50mdiuDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:
// Generated message map functions
protected:
    //{{AFX_MSG(CZ50mdiuDoc)
    // NOTE - the ClassWizard will add and remove member
    // functions here.
    //   DO NOT EDIT what you see in these blocks of
    //generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////
/////////{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional
declarations immediately before the previous line.
#endif //
!defined(AFX_Z50MDIUDOC_H__99E5ECCD_669D_11D7_8A00_82D38D
AAD863__INCLUDED_)
#include "z50mdiuDoc.h"
```

```
int poztt=0;
class Element { protected:
                int kvx,kvix,poz;
                int x,y;
                int *masz;
              char name[10];
                public:
                Element(int nv,int xn =150,int yn =20,
                        int nvi =1)
                { kvx=nv; kvix=nvi; x=xn; y=yn; poz=1;
                  masz=new int[kvx];
                  for(int i=0;i<kvx;i++) masz[i]=0; }
                virtual ~Element(void)
                { if(masz!=NULL) delete []masz; }
                int getkvx() { return kvx; }
               void settp(int n) { poz=n; }
                void ust(int v,int zn=0,int pm=0)
                { extern int poztt;
                   if(!pm)
                   { if(zn>1 && masz[v-1]!=zn-48)
                     { if(masz[v-1]) masz[v-1]=0;
                       else masz[v-1]=1;
                      }
                    }
                   else
                   { for(int i=1;i<=kvx;i++)
                       if(v>=x-35 && v<=x-30 &&
                          zn>=y+i*30-3 && zn<=y*i*30+3)
                          poz=i;
                       if(poz<=kvx)
                       { if(masz[poz-1]) masz[poz-1]=0;
                         else masz[poz-1]=1;
                         poztt=poz;
                       }
                   }
                   return;
                }
               };
class Memory:public Element
             { friend class Sxema;
               protected:
               int vx,nvx;
               public:
               Memory(int n,char *nm):Element(n,150,20,2)
               { lstrcpy(name,nm); }
                 virtual ~Memory(){;}
                 void showm(CDC *hDC,int pr =0);
```

```
                 void reak()
                 { if(masz[1]==0 && masz[0]==1 &&
                      masz[2]==1)vx=0;
                   else if(masz[1]==1 && masz[0]==0 &&
                           masz[2]==1 && masz[3]==0)
                           vx=1;
                        else if(masz[3]==1) vx=0;
                    nvx=!vx;
                  }
                };
class Combi:public Element
            { friend class Sxema;
              protected:
              int pi;
              public:
              Combi(int n,char *nm):Element(n)
             { pi=1; lstrcpy(name,nm); }
              virtual ~Combi(){;}
              int reak()
              { int z=0,i;
                for(i=0;i<kvx;i++)
                   if(masz[i]) { z=1; break; }
                if(z) z=0; else z=1;
                return(z);
              }
              void showc(CDC *hDC,int pr =0);
             };
class Sxema{ protected:
             Combi *cs1,*cs2;
              Memory *ts;
              int maszs[10],pozs;
             public:
            Sxema()
            { cs1=new Combi(4," | (1) ");
              cs2=new Combi(2," | (2) ");
              ts=new Memory(4,"R-S-T");
              for(int i=0;i<10;i++) maszs[i]=0;
              pozs=1;
            }
            void settp(int n) { pozs=n; }
             void ust(int v,int zn=0,int pm=0)
             { extern int poztt;
               int xn,yn,nn,j;
               if(!pm)
               { if(zn>1 && maszs[v-1]!=zn-48)
                  if(maszs[v-1]) maszs[v-1]=0;
                  else maszs[v-1]=1;
```

```
                }
                else
                  { for(int i=1;i<=3;i++)
                     { if(i==1) { xn=cs1->x-30; yn=cs1->y;
                                  nn=cs1->kvx; }
                       else if(i==2) { xn=ts->x-30;
                                       yn=ts->y+30;
                                       nn=ts->kvx-1; }
                       else { xn=cs2->x-30; yn=cs2->y;
                              nn=cs2->kvx-1; }
                       for(j=1;j<=nn;j++)
                         if(v>=xn-5 && v<=xn &&
                            zn>=yn+j*30-3 && zn<=yn+j*30+3)
                         { if(i==1) pozs=j;
                           else if(i==2) pozs=j+cs1->kvx+1;
                           else pozs=cs1->kvx+ts->kvx+1;
                           if(maszs[pozs-1])
                              maszs[pozs-1]=0;
                           else maszs[pozs-1]=1;
                           poztt=pozs;
                           return;
                         }
                     }
                  }
                }
                void show(CDC *hDC);
            };
void Memory::showm(CDC *hDC,int pr)
{ char *mv[4]={"R","S","T","Ñáðîñ"};
  char buf[14];
  COLORREF cv;
  TEXTMETRIC tm;
  int prh=0,pper=0;
  CSize ll=hDC->GetTextExtent("R-S-T",5);
  int l=ll.cx;
  hDC->GetTextMetrics(&tm);
  int yc=tm.tmHeight;
  int xc=tm.tmAveCharWidth;
  hDC->Rectangle(x,y,x+l+30,y+(kvx+1)*30);
  for(int i=1;i<=kvx;i++)
  { hDC->MoveTo(x,y+i*30);
    hDC->LineTo(x-30,y+i*30);
    hDC->TextOut(x+2,y+i*30-yc/2,mv[i-1],
                 lstrlen(mv[i-1]));
  }
  hDC->MoveTo(x+l+30,y+(kvx+1)*10);
  hDC->LineTo(x+l+60,y+(kvx+1)*10);
```

```
  hDC->Ellipse(x+l+30,y+(kvx+1)*20-4,
               x+l+38,y+(kvx+1)*20+4);
  hDC->MoveTo(x+l+38,y+(kvx+1)*20);
  hDC->LineTo(x+l+60,y+(kvx+1)*20);
  hDC->TextOut(x+15,(y+y+(kvx+1)*30-yc)/2,"R-S-T",5);
  cv=hDC->SetTextColor(RGB(255,0,0));
  if(poz>kvx && !pr) poz=1;
  if(!poz && !pr) poz=kvx;
  if((poz!=1 && pr) || !pr)
  { if(masz[poz-1]==0) hDC->TextOut(x-30-xc,y+poz*30-
                                    yc/2,"0",1);
    else hDC->TextOut(x-30-xc,y+poz*30-yc/2,"1",1);
  }
  hDC->SetTextColor(cv);
  for(int k=1;k<=kvx;k++)
  { if((k==poz && !pr)||(k==poz && pr==1)||(k==1 && pr))
       continue;
    if(masz[k-1]==0) hDC->TextOut(x-30-xc,y+k*30-
                                  yc/2,"0",1);
    else hDC->TextOut(x-30-xc,y+k*30-yc/2,"1",1);
  }
  reak();
  if(!pr)
  { wsprintf(buf,"%d",vx);
    hDC->SetTextColor(RGB(0,0,255));
    hDC->TextOut(x+l+60+xc,y+(kvx+1)*10-yc/2,buf,1);
  }
  wsprintf(buf,"%d",nvx);
  if(pr) hDC->SetTextColor(RGB(0,0,255));
  hDC->TextOut(x+l+60+xc,y+(kvx+1)*20-yc/2,buf,1);
  hDC->SetTextColor(cv);
  return;
}
void Combi::showc(CDC *hDC,int pr)
{ char buf[6];
  COLORREF cv;
  TEXTMETRIC tm;
  int prh=0;
  hDC->GetTextMetrics(&tm);
  int yc=tm.tmHeight;
  int xc=tm.tmAveCharWidth;
  CSize ll=hDC->GetTextExtent(name,lstrlen(name));
  int l=ll.cx;
  hDC->Rectangle(x,y,x+l+30,y+(kvx+1)*30);
  for(int i=1;i<=kvx;i++)
  { hDC->MoveTo(x,y+i*30);
    hDC->LineTo(x-30,y+i*30);
```

```
  }
  cv=hDC->GetTextColor();
  hDC->Ellipse(x+l+30,y+(kvx+1)*15-4,x+l+38,
               y+(kvx+1)*15+4);
  hDC->MoveTo(x+l+38,y+(kvx+1)*15);
  hDC->LineTo(x+l+68,y+(kvx+1)*15);
  hDC->TextOut(x+15,(y+y+(kvx+1)*30-yc)/2,name,
                    lstrlen(name));
  if(poz>kvx&&!pr) poz=1;
  if(!poz&&!pr) poz=kvx;
  if((strstr(name,"(1)")&&pr==2)||(strstr(name,"(2)")&&
      poz<kvx&&pr==2)||!pr)
  { hDC->SetTextColor(RGB(255,0,0));
    if(masz[poz-1]==0) hDC->TextOut(x-30-xc,y+poz*30-
                                    yc/2,"0",1);
    else hDC->TextOut(x-30-xc,y+poz*30-yc/2,"1",1);
    hDC->SetTextColor(RGB(0,0,0));
  }
  for(int k=1;k<=kvx;k++)
  { if((k==poz&&!pr)||(strstr(name,"(1)") &&
        pr==2&&poz==k)||(strstr(name,"(2)")&&
        ((k==2&&pr==1)||pr==2))) continue;
    if(masz[k-1]==0) hDC->TextOut(x-30-xc,y+k*30-
                                  yc/2,"0",1);
    else hDC->TextOut(x-30-xc,y+k*30-yc/2,"1",1);
  }
  i=reak();
  if(!pr||(pr&&strstr(name,"(2)")))
  { wsprintf(buf,"%d",i);
    hDC->SetTextColor(RGB(0,0,255));
    hDC->TextOut(x+l+60+xc,y+(kvx+1)*15-yc/2,buf,1);
    hDC->SetTextColor(cv);
  }
  return;
}

void Sxema::show(CDC *hDC)
{ int i;
  CSize ll=hDC->GetTextExtent(cs1->name,
                              lstrlen(cs1->name));
  int l=ll.cx;
  ts->x=cs1->x+l+90;
  ts->y=cs1->y+(cs1->kvx+1)*15-30;
  ll=hDC->GetTextExtent(ts->name,lstrlen(ts->name));
  l=ll.cx;
  cs2->x=ts->x+l+88;
  cs2->y=ts->y+(ts->kvx+1)*10-60;
```

```
  for(i=0;i<cs1->kvx;i++) cs1->masz[i]=maszs[i];
  for(i=0;i<cs2->kvx;i++) cs2->masz[i]=maszs[i+cs1->kvx+
                                          ts->kvx];
  for(i=0;i<ts->kvx;i++) ts->masz[i]=maszs[i+cs1->kvx];
  if(pozs>9) pozs=1;
  if(!pozs) pozs=9;
  if(pozs<5) cs1->settp(pozs);
  else if(pozs<9) ts->settp(pozs-4);
  else cs2->settp(1);
  ts->masz[0]=cs1->reak();
  ts->reak();
  cs2->masz[1]=ts->vx;
  if(pozs<5) cs1->showc(hDC,2);
  else cs1->showc(hDC,1);
  if(pozs>=5 && pozs<9) ts->showm(hDC,1);
  else ts->showm(hDC,2);
  if(pozs==9) cs2->showc(hDC,2);
  else cs2->showc(hDC,1);
  return;
}

// z50mdiuView.cpp : implementation of the CZ50mdiuView
class
//

#include "stdafx.h"
#include "z50mdiu.h"

#include "z50mdiuDoc.h"
#include "z50mdiuView.h"
#include "ZAd50c.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////
///////// CZ50mdiuView
IMPLEMENT_DYNCREATE(CZ50mdiuView, CView)
BEGIN_MESSAGE_MAP(CZ50mdiuView, CView)
    //{{AFX_MSG_MAP(CZ50mdiuView)
    ON_WM_KEYDOWN()
    ON_WM_LBUTTONDOWN()
    ON_COMMAND(IDM_APP_ABOUT, OnAppAbout)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////
///////// CZ50mdiuView construction/destruction

CZ50mdiuView::CZ50mdiuView()
{
    // TODO: add construction code here
}
CZ50mdiuView::~CZ50mdiuView()
{
}
BOOL CZ50mdiuView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by
modifying
    //  the CREATESTRUCT cs
    return CView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////
///////// CZ50mdiuView drawing
void CZ50mdiuView::OnDraw(CDC* pDC)
{
    CZ50mdiuDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    extern int pre;
    if(pre==1)
    { pDoc->SetTitle("Комбинационный элемент");
      pDoc->pe=1;
      pDoc->pozt=1;
      pDoc->ue=new Combi(4," | ");
    }
    else if(pre==2)
    { pDoc->SetTitle("Триггер");
      pDoc->pe=2;
      pDoc->pozt=1;
      pDoc->ue=new Memory(4,"R-S-T-");
    }
    else if(pre==3)
    { pDoc->SetTitle("Схема");
      pDoc->pe=3;
      pDoc->pozt=1;
      pDoc->ue=new Sxema();
    }
    pre=0;
    if(pDoc->pe==1) ((Combi *)(pDoc->ue))->showc(pDC);
    else if(pDoc->pe==2)
           ((Memory *)(pDoc->ue))->showm(pDC);
```

```
    else ((Sxema *)(pDoc->ue))->show(pDC);
}
/////////////////////////////////////////////////////////////
///////// CZ50mdiuView diagnostics
#ifdef _DEBUG
void CZ50mdiuView::AssertValid() const
{
    CView::AssertValid();
}
void CZ50mdiuView::Dump(CDumpContext& dc) const
{ CView::Dump(dc); }

CZ50mdiuDoc* CZ50mdiuView::GetDocument() // non-debug
version is
                                          // inline
{
    ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CZ50mdiuDoc)));
    return (CZ50mdiuDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////
///////// CZ50mdiuView message handlers
void CZ50mdiuView::OnKeyDown(UINT nChar,UINT nRepCnt,
                             UINT nFlags)
{
// TODO: Add your message handler code here and/or call
// default
    CZ50mdiuDoc* pDoc = GetDocument();
    CDC *pDC=GetDC();
    if(nChar==40 || nChar==38)
    { if(nChar==40) pDoc->pozt++;
      else if(nChar==38) pDoc->pozt--;
      if(pDoc->pe==1)
      { if(pDoc->pozt>((Combi *)(pDoc->ue))->getkvx())
        { pDoc->pozt=1;
          ((Combi *)(pDoc->ue))->settp(1);
        }
        else if(pDoc->pozt==0)
        { pDoc->pozt=((Combi *)(pDoc->ue))->getkvx();
          ((Combi *)(pDoc->ue))->settp(((Combi *)
                                (pDoc->ue))->getkvx());
        }
        else ((Combi *)(pDoc->ue))->settp(pDoc->pozt);
        ((Combi *)(pDoc->ue))->showc(pDC);
      }
      else if(pDoc->pe==2)
```

```
      { if(pDoc->pozt>((Memory *)(pDoc->ue))->getkvx())
        { pDoc->pozt=1; ((Memory *)(pDoc->ue))->settp(1);
        }
        else if(pDoc->pozt==0)
        { pDoc->pozt=((Memory *)(pDoc->ue))->getkvx();
          ((Combi *)(pDoc->ue))->settp(((Memory *)
                                (pDoc->ue))->getkvx());
        }
        else ((Memory *)(pDoc->ue))->settp(pDoc->pozt);
        ((Memory *)(pDoc->ue))->showm(pDC);
      }
      else if(pDoc->pe==3)
      { if(pDoc->pozt==5)
          if(nChar==40) pDoc->pozt++;
          else if(nChar==38) pDoc->pozt--;
        if(pDoc->pozt>9)
        { pDoc->pozt=1;
          ((Sxema *)(pDoc->ue))->settp(1);
        }
        else if(pDoc->pozt==0)
        { pDoc->pozt=9;
          ((Sxema *)(pDoc->ue))->settp(9);
        }
        else ((Sxema *)(pDoc->ue))->settp(pDoc->pozt);
        ((Sxema *)(pDoc->ue))->show(pDC);
      }
    }
    else if(nChar==49 || nChar==48)
    { if(pDoc->pe==1)
      { ((Combi *)(pDoc->ue))->ust(pDoc->pozt,nChar);
        ((Combi *)(pDoc->ue))->showc(pDC);
      }
      if(pDoc->pe==2)
      { ((Memory *)(pDoc->ue))->ust(pDoc->pozt,nChar);
        ((Memory *)(pDoc->ue))->showm(pDC);
      }
      if(pDoc->pe==3)
      { ((Sxema *)(pDoc->ue))->ust(pDoc->pozt,nChar);
        ((Sxema *)(pDoc->ue))->show(pDC);
      }
    }
    CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
void CZ50mdiuView::OnLButtonDown(UINT nFlags,
                                 CPoint point)
{
```

```
// TODO: Add your message handler code here and/or call
// default
    CZ50mdiuDoc* pDoc = GetDocument();
    CDC *pDC=GetDC();
    extern int poztt;
    if(pDoc->pe==1)
    { ((Combi *)(pDoc->ue))->ust(point.x,point.y,1);
      pDoc->pozt=poztt;
      ((Combi *)(pDoc->ue))->showc(pDC);
    }
    else if(pDoc->pe==2)
    { ((Memory *)(pDoc->ue))->ust(point.x,point.y,1);
      pDoc->pozt=poztt;
      ((Memory *)(pDoc->ue))->showm(pDC);
    }
    else if(pDoc->pe==3)
    { ((Sxema *)(pDoc->ue))->ust(point.x,point.y,1);
      pDoc->pozt=poztt;
      ((Sxema *)(pDoc->ue))->show(pDC);
    }
    CView::OnLButtonDown(nFlags, point);
}
void CZ50mdiuView::OnAppAbout()
{
    // TODO: Add your command handler code here
   MessageBox("Установка значений – клавишами 0,1 и
стрелками вверх/вниз или левой клавишей мыши.","Помощь");
}
```

2.5. Использование множественных потоков

Потоком (thread) называется выполнение последовательности команд программы. Все программы, рассмотренные ранее, выполняются в виде одного потока, называемого *первичным*. Однако в программе, написанной для Windows, можно запускать один или несколько *вторичных потоков* (или подзадач). Каждый поток независимо выполняет свою последовательность команд, соответствующих тексту программы. С точки зрения пользователя или программиста, создающего приложение, потоки в программе выполняются одновременно. Операционная система достигает этой кажущейся одновременности за счет быстрого переключения управления с одного потока на другой. В многопроцессорной системе потоки могут выполняться одновременно в буквальном смысле.

Многопотоковая обработка особенно полезна в программах с графическим интерфейсом, когда первичный поток выделяется для обработки сообщений, что позволяет программе быстро реагировать на поступающие команды и другие события. Вторичный поток (или потоки) обычно используется для вы-

полнения любой длительной задачи, которая блокировала бы обработку сообщений при ее выполнении первичным потоком. Кроме того, все потоки программы выполняются в одной области памяти и используют общий набор ресурсов Windows, следовательно, могут совместно использовать переменные и объекты Windows (окна, перья, распределения памяти, контексты устройств и т.д.).

Новый проект имеет стандартные установки, но если их необходимо изменить на многопотоковое приложение, переустановите их следующим образом: выберите команду Settings… в меню Project и имя проекта в иерархическом списке, откройте вкладку C/C++ и выберите пункт Code Generation в списке Category. Для задания выходной конфигурации проекта в списке Use Time-Time Library выберите параметр Multitreaded или Multitreaded DLL.

После построения кода программы в файле *Имя_пректа*View.h будет автоматически создан указатель на объект: CwinThread *m_PDrawThread, который используется для хранения адреса объекта, управляющего вторичным потоком. Кроме того в файле *Имя_пректа*View.cpp дополнения в конструктор и деструктор соответствующего класса: в констукторе – m_PDrawThread=0, а в деструкторе – delete m_PdrawThread.

Для запуска нового потока используется глобальная MFC-функция, которая имеет следующие аргументы.

```
CWinThread*  AfxBeginThread(AFX_THREADPROC pfnThreadProc,
                            LPVOID pParam,  int nPiorrity=
                            THREAD_PRIORITY_NORMAL,
                            UINT nStackSize=0,
                            DWORD dwCreateFlags=0,
                            LPSECURITY_ATTRIBUTES
                            lpSecurityAttra=NULL)
```

Первый параметр pfnThreadProc задает функцию потока, и новый поток начинает ее выполнять. При выходе из нее новый поток завершается. Параметр pParam задает значение дескриптора окна, с которым будет связан новый поток. Заголовок функции потока определяется следующим образом.

UINT ThreadFunction(LPVOID pParam).

Четыре последних параметра имеют стандартные значения. Параметр nPriority задает *приоритет* нового потока, который определяет, как часто поток будет выполняться при переключении управления от одного потока к другому. Описание различных приоритетов приведено в следующих разделах справочной системы: Platform SDK, Windows Base Service, Processes and Thread).

Если параметр dwCreateFlags равен 0 (стандартное значение), то новый поток начинает выполняться немедленно. Если же присвоено значение CREATE_SUSPENDED, то он не запустится до вызова функции CwinThread::ResumeThread, описанной ниже.

Если пятый параметр равен 0, то объект потока уничтожает себя после завершения потока, а если этот параметр имеет значение CREATE_SUSPENDED, что позволяет присваивать значение FALSE переменной потока m_bAutoDelete перед его завершением.

Если в программе прсутсвует следующий код

```
CwinThread *PwinThread;
PwinThread=AfxBeginThread(TreadFunction, …),
```

То для временной простановки выполнения потока вызывается функция SuspendThread класса CwinThread.

```
PwinThread->SuspendThread();
```

Для повторного запуска потока необходимо вызвать функцию PwinThread->ResumeThread() .

Функция Win32 API::GetExitCodeThread вызывается, чтобы определить, продолжается ли выполнение потока. Если поток завершен, то она позволяет получить его код возврата. Например.

```
DWORD ExitCode;
:: GetExitCodeThread(PwinThread->m_hThread, &ExitCode);
if(ExitCode==STILL_ACTIVE)
    //  поток продолжает выполнение …
else
    //  поток завершен и переменная ExitCode содержит код возврата.
```

Наконец, следует сказать об ограничениях на использование MFC-классов.

Первое: два потока не должны одновременно получать доступ к одному объекту MFC. Например, два потока могут использовать один объект класса Cstring при условии, что они не будут одновременно вызывать функцию класса Cstring. Два потока могут одновременно получить доступ к разным объектам, принадлежащим даже одному MFC-классу, например, к двум объектам класса CString.

Второе ограничение: объект любого из указанных ниже классов или объект класса, наследуемый от них, доступен только для потока, создавшего объект. Это объекты классов: CWnd, CDC, CMenu, CgdiObject. Эти классы имеют одно общее свойство: каждый из них хранит дескриптор для некоторых подчиненных объектов.

Чтобы преодолеть эти ограничения Win32 API содержит набор *объектов синхронизации*, использующихся для синхронизации отдельных потоков. Используя эти объекты, можно организовать синхронизацию потоков: запрет одновременного доступа к ресурсам более одного потока.

Одним из наиболее простых и типичных объектов синхронизации является *мьютекс*. Этот объект используется для ограничения одновременного доступа к данному ресурсу единственным потоком. Например.

```
HANDLE  HMutex;  //объявляется глобально и все потоки имеют к нему
                 // доступ.
                 // …
void SomeFunction()
{
   //  …
   HMutex=::CreateMutex(NULL,     // присваивает стандартные атрибуты
                                  // защиты
                        FALSE,  // мьютекс изначально свободный
```

```
                                        NULL);  // имя мьютексу не присваивается
                                                //  …
}
```

Функция CreateMutex возвращает дескриптор, используемый потоками программы для ссылки на мьютекс.

Далее следует добавить вызов функции ::WaitForSingleObject Win32 API в начало каждого блока, обращающегося к защищенным ресурсам, и вызов функции Win32 API ::ReleaseMutex в конец каждого из этих блоков. Например, можно запретить одновременный доступ нескольких потоков к глобальному счетчику таким образом.

```
::WaitForSingleObject(HMutex,       // дескриптор мьютекса
                                    INFINITE);  //  ждите столько, сколько нужно
++Count;
cout<<Count<<'\n';
::ReleaseMutex(HMutex);
```

Далее приведем пример программы, которая при инициализации пункта меню «Счет», создает два потока. Один из потоков создает графическое изображение и выполняет некоторый счет (строящееся изображение отображается на экране по мере его создания), а второй поток занят только счетом и отображением на экране соответствующего значения. После завершения построения графического изображения, второй поток выводит сумму подсчитанных значений завершенного и своего потоков. Если после запуска главное окно приложения не разделено, то оба потока осуществляют вывод в это окно. В случае разделения окна после этого вывода, оба потока осуществляют вывод в одно и то же из рааделенных окон. Если разделить главное окно сразу после запуска приложения, то при выборе пункта меню «Счет», вывод потоков осуществляется в разные окна представления (рис. 2.33).

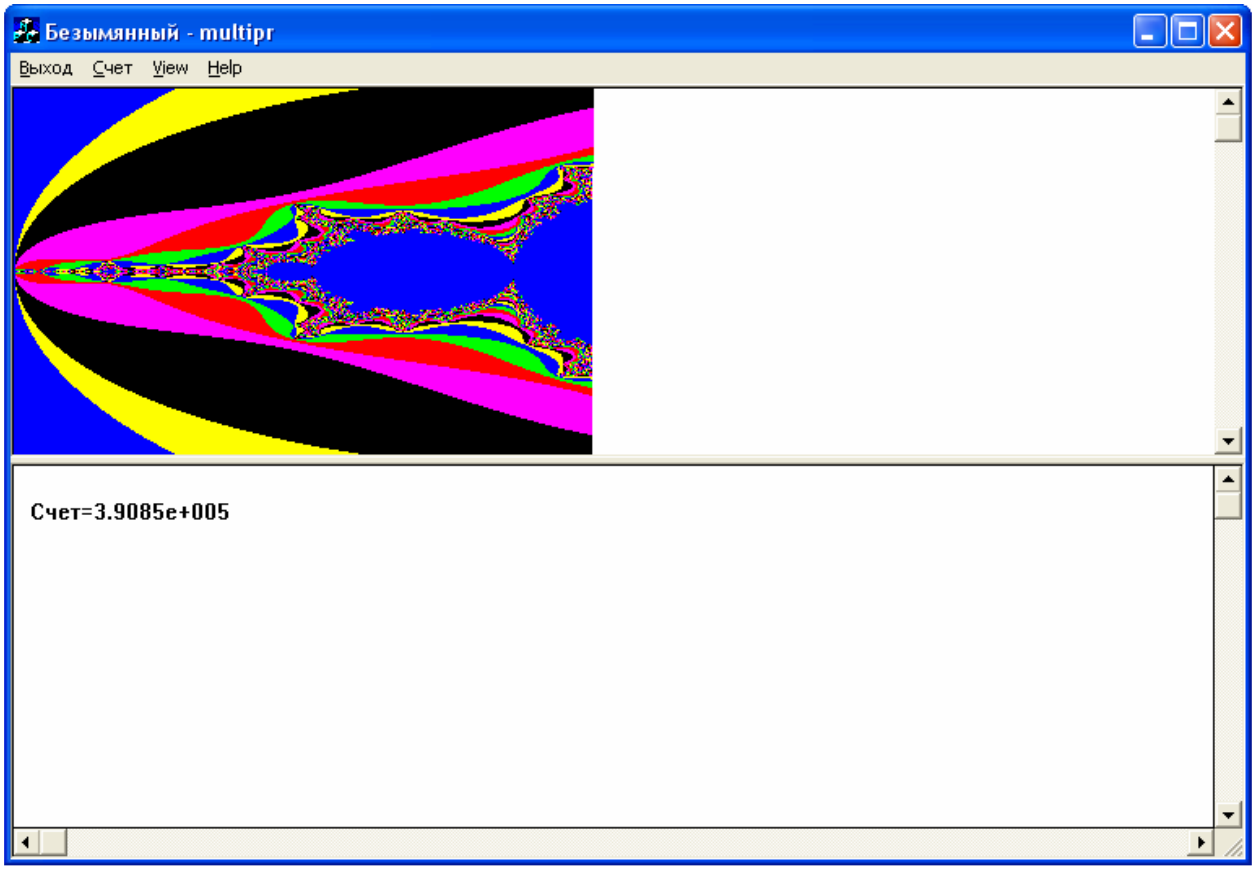


Рис. 2.33. Вид окна приложения в процессе выполнения двух потоков с выводом в разные окна представления.

В приводимом примере представлены только добавленные и измененные фрагменты программы, после того как она была изначально построена. После создания и коррекции проекта, изменялся только файл *Имя_проекта*View.cpp.

```
// multiprView.cpp : implementation of the CMultiprView
// class

#include "stdafx.h"
#include "multipr.h"

#include "multiprDoc.h"
#include "multiprView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#define CIMAX 1.2
#define CIMIN -1.2
#define CRMAX 1.0
#define CRMIN -2.0
#define NMAX 128
```

```
double k1=0.,kod=0.,vv=0.,k2=0.;
/////////////////////////////////////////////////////////////
////////////////////
// CMultiprView

IMPLEMENT_DYNCREATE(CMultiprView, CView)

BEGIN_MESSAGE_MAP(CMultiprView, CView)
    //{{AFX_MSG_MAP(CMultiprView)
    ON_WM_SIZE()
    ON_COMMAND(ID_FILE_NEW, OnFileNew)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    ON_COMMAND(IDM_SH, OnSh)
    ON_COMMAND(IDM_GGG, OnGgg)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////
////////////////////
// CMultiprView construction/destruction

CMultiprView::CMultiprView()
{
    // TODO: add construction code here
   m_PDrawThread=0;
   m_PDrawThread2=0;

}

CMultiprView::~CMultiprView()
{
   delete m_PDrawThread;
   delete m_PDrawThread2;
}

BOOL CMultiprView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by
    // modifying the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

#include <math.h>
int kon1=0;
int ps=0,r=0;
UINT DrawFractal(LPVOID PHWndView)
```

```
{ CClientDC ClientDC(CWnd::FromHandle(*(HWND
*)PHWndView));
  float CI,I,ISqr,R,RSqr,x=0;
  int Row,ColorVal,Col;
  static DWORD;
  ColorTable[6]={0x0000ff,0x00ff00,0xff0000,0x00ffff,
                 0xffff00,0xff00ff};
  RECT rc;
  ClientDC.GetClipBox(&rc);
  float DCR=(float)((CRMAX-CRMIN)/(rc.right-1));
  float DCI=(float)((CIMAX-CIMIN)/(rc.bottom-1));
  float CR=(float)CRMIN;
  for(Col=0;Col<rc.right;++Col)
  { CI=(float)CIMAX;
    for(Row=0;Row<rc.bottom;++Row)
    { R=I=(float)0.0;
      RSqr=ISqr=(float)0.0;
      ColorVal=0;
       while(ColorVal<NMAX && RSqr+ISqr<4)
      { ++ColorVal;
         RSqr=R*R;
        ISqr=I*I;
        I*=R;
        I+=I+CI;
        R=RSqr-ISqr+CR;
      }
      for(int i=0;i<100;i++)
         x+=(float)(sin(x)/(tan(x+1.)+1./tan(x+1)));
      ClientDC.SetPixelV(Col,Row,ColorTable[ColorVal%6]);
      CI-=DCI;
    }
    CR+=DCR;
    k2+=100000;
  }
  CR=(float)CRMIN;
  kon1=1;
  ClientDC.Rectangle(rc.left-1,rc.top-1,rc.right,
                     rc.bottom);
  return 0;
}

char rab[60];
UINT DrawGGG(LPVOID PHWndView)
{ CClientDC ClientDC(CWnd::FromHandle(*(HWND *)
                                      PHWndView));
  RECT rc;
  ClientDC.GetClipBox(&rc);
```

```
  ClientDC.Rectangle(rc.left-1,rc.top-1,rc.right,
                     rc.bottom);
xx: kod++;
    sprintf(rab,"Счет=%.5lg",kod);
    ClientDC.TextOut(10,20,rab,strlen(rab));
    if(kon1)
    { vv=kod;
      vv+=k2;
      sprintf(rab,"Итог=%.5lg (%.5lg + %lg)",vv,kod,k2);
      ClientDC.TextOut(10,40,rab,strlen(rab));
      return 0;
    }
    goto xx;
}
////////////////////////////////////////////////////////
////////////////////
// CMultiprView drawing

int pa=0;
CWnd *cw;
HWND hw1=0,hw2=0;
void CMultiprView::OnDraw(CDC* pDC)
{
    CMultiprDoc* pDoc = (CMultiprDoc *)GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    if(!r) { r=1; return; }
    CClientDC ClientDC(this);
    if(pa) { pa=0; return; }
    if(r==1)
    { cw=ClientDC.GetWindow();
      hw1=cw->m_hWnd;
      r=2; return;
    }
    else if(r==2)
    { cw=ClientDC.GetWindow();
      hw2=cw->m_hWnd;
      r=3; return;
    }
    if(r==4 || r==5)
    { if(r==4)
      { if(!hw1)
          m_PDrawThread=AfxBeginThread(DrawFractal,
&m_hWnd,THREAD_PRIORITY_NORMAL, 0,CREATE_SUSPENDED);
        else m_PDrawThread=AfxBeginThread(DrawFractal,
&hw1,THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
         m_PDrawThread->m_bAutoDelete=FALSE;
```

```
        m_PDrawThread->ResumeThread();
       }
       else
       { if(!hw1)
m_PDrawThread2=AfxBeginThread(DrawGGG,&m_hWnd,
              THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
        else m_PDrawThread2=AfxBeginThread(DrawGGG,&hw2,
              THREAD_PRIORITY_NORMAL,0,CREATE_SUSPENDED);
         m_PDrawThread2->m_bAutoDelete=FALSE;
        m_PDrawThread2->ResumeThread();
       }
     }
}

////////////////////////////////////////////////////////////
/////////////////////
// CMultiprView diagnostics

#ifdef _DEBUG
void CMultiprView::AssertValid() const
{
    CView::AssertValid();
}

void CMultiprView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CMultiprDoc* CMultiprView::GetDocument()
 // non-debug version is inline
{
    ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CMultiprDoc)));
    return (CMultiprDoc*)m_pDocument;
}
#endif //_DEBUG

////////////////////////////////////////////////////////////
/////////////////////
// CMultiprView message handlers
void CMultiprView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);
    // TODO: Add your message handler code here
   if(cx<=1 || cy<= 1) return;
   if(m_PDrawThread)
```

```
   { ps=1;
     CDC *cdc;
     cdc=GetDC();
     OnDraw(cdc);
   }
}

void CMultiprView::OnFileNew()
{
     // TODO: Add your command handler code here
     r=4; k1=0;
     CDC *cdc=GetDC();
     OnDraw(cdc);
}

void CMultiprView::OnGgg()
{
     // TODO: Add your command handler code here
     r=5; kod=0;
     CDC *cdc=GetDC();
     OnDraw(cdc);

}

/////////////////////////////////////////////////////////////////////
////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
     CAboutDlg();

// Dialog Data
     //{{AFX_DATA(CAboutDlg)
     enum { IDD = IDD_ABOUTBOX };
     //}}AFX_DATA

     // ClassWizard generated virtual function overrides
     //{{AFX_VIRTUAL(CAboutDlg)
     protected:
     virtual void DoDataExchange(CDataExchange* pDX);
     // DDX/DDV support
     //}}AFX_VIRTUAL

// Implementation
protected:
```

```
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


// App command to run the dialog
void CMultiprView::OnAppAbout()
{
    // TODO: Add your command handler code here
    if(m_PDrawThread) m_PDrawThread->SuspendThread();
    pa=1;
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
    if(m_PDrawThread) m_PDrawThread->ResumeThread();
}
/////////////////////////////////////////////////////////////////////
////////////////////
// CMultiprApp message handlers


void CMultiprView::OnSh()
{
    // TODO: Add your command handler code here
    if(!hw1)
```

```
     if(MessageBox("Для наглядности, перед запуском
счета\n лучше разделить окна
представлений.","Внимание",MB_YESNO)==IDYES) return;
    kon1=0; ps=0;
    vv=kod=k1=k2=0;
    m_PDrawThread=0;
    m_PDrawThread2=0;
    OnFileNew();
    OnGgg();
}
```

СОДЕРЖАНИЕ

1. Основные сообщения Windows. 5
1.1 Назначение и обработка сообщений в Win32 application. 5
1.2.Основные сообщения общего назначения. 6
1.3. Специализированные сообщения. 26
1.3.1. Сообщения окна-списка (класс Listbox). 27
1.3.2. Сообщения окна-редактора (класс Edit). 32
1.3.3. Сообщения комбинированного списка (класс Combobox). 36
1.3.4. Сообщения кнопок (класс Button). 43
1.4. Пример программы ColorDialog. 46
2. Создание программ с использованием мастера генерации приложений
AppWizard и библиотеки MFC 57
2.1. Прокрутка и преобразование координат 68
2.2. Печать и предварительный просмотр 71
2.3. Добавление средств разделения окна 76
2.4. Создание многодокументных приложений 101
2.5. Использование множественных потоков 113

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

Кафедра Прикладной Математики (ПМ), в дальнейшем Информатики и Прикладной Математики (ИПМ), организована в 1975 г. путем выделения части сотрудников из коллектива кафедры вычислительной техники для обеспечения обучения основам программирования для всех специальностей ЛИТМО. Одновременно кафедра сохранила за собой ряд специальных курсов:

- Основы дискретной математики,
- Теории вероятностей,
- Теории булевых функций и арифметические и логические основы ЭВМ,
- Основы САПР и специализированные базы данных,
- Диагностика и контроль ЭВМ и ряд других.

Научное направление кафедры определилось как системы автоматизации проектирования в области контроля и диагностики логических схем ЭВМ. Для обеспечения учебного процесса и выполнения НИР в интересах министерств оборонного профиля (МОМ,МРП,МАП и ряда других) были приобретены за счет Х/Д тем следующие вычислительные машины: Минск-32,ЕС-1022,М-4030, ЕС-1033 и ЕС-1045, что позволило создать общеинститутский ВЦ, в котором проходили подготовку будущие специалисты, в дальнейшем преподаватели, в области программирования и вычислительной техники.

Для организации учебного процесса были созданы дисплейные учебные классы на базе ЕС-1033, работающие как в пакетном режиме, так и в режиме разделения времени. В это же время был создан СВЦ на машинах Мир-1 и Мир-2, замененных в дальнейшем на машины СМ-3 и СМ-4. Создание дисплейных классов позволило удовлетворить потребности учебного процесса по программированию в лабораторной базе.

Научное направление кафедры состояло в разработке регулярных методов проектирования тестов для логических схем ЭВМ на основе аппарата исчисления кубических комплексов. В результате выполнения ряда НИР были

созданы САПР тестов для плат бортовых ЦВМ, стендовое оборудование для контроля, диагностики и наладке цифровых схем в виде двухслойных и многослойных плат ЭВМ. Сотрудничество с академическими организациями и участие в ежегодных школах-семинарах по технической диагностике, проводимых под руководством члена-корреспондента АН СССР Пархоменко П.П. (ИПУ г. Москва) отраслевыми НИИ (НИЦЭВТ г. Москва), конструкторскими бюро (КБЭ г. Харьков) и промышленными предприятиями (НПО ВТ г. Минск), позволило интегрировать научные исследования с последующим внедрением результатов в производственные технологии и учебный процесс в единое целое.

В результате научных исследований были подготовлены кадры высшей квалификации: защищены 5 докторских и несколько десятков кандидатских диссертаций. После защиты сотрудники НИЧ нашего института, как правило, переходили на преподавательскую работу. В частности, по кафедре ИПМ защитили диссертации и стали преподавателями: Шипилов П.А., Голованевский Г.Л., Блохин В.Н., Усвятский А.Е., Звягин В.Ф., Голыничев В.Н., Щупак Ю.А., Кукушкин Б.А., Раков С.В., Слоев Б.А., Павловская Т.А., Денисова Э.В. и ряд других.

В области образовательного процесса кафедра сочетает два рода деятельности, обеспечивая обучение всех специальностей университета основам программирования и алгоритмических языков и подготовку специалистов как выпускающая кафедра совместно с кафедрой ВТ по направлению "Информатика и вычислительная техника" и по специализации "Технология разработки программных систем" в рамках специальности "Вычислительные комплексы, системы и сети".

Общеобразовательная нагрузка состоит из двух компонент:
- обучение основам алгоритмических языков в виде грамотного построения тех или иных грамматических конструкций в заданном языке, как то Ассемблер, C++, Паскаль и др. по заявкам выпускающих кафедр;
- основам вычислительных процессов путем алгоритмизации и программирования типовых структур, таких как:
  - линейные и интервальные формулы;
  - простые и сложные условия предикаты методом функциональной декомпозиции алгебраических (булевых) выражений логических функций;
  - циклические и ациклические параллельные структуры с учетом их вложенности.

Подготовка специалистов сочетает в себе основы знаний по вычислительным комплексам и сетям с методами и средствами программирования с углубленным изучением алгоритмических языков, что позволяет осуществлять подготовку синтетических специалистов по разработке специализированного программного обеспечения: построение имитационно-моделирующих программ, систем автоматизированного проектирования, функциональных и управляющих программ, разработку баз знаний и данных, методам верификации и тестирования программного продукта на протяжении всего его жизненного цикла, методам диагностики вирусоопасных кодов и ряду других перспективных направлений в области технологии программирования.

Кафедра занимается разработкой теоретических основ программирования в области моделирования, верификации, тестирования и диагностики вычислительных процессов программ. Вычислительные процессы, порождаемые программами, представляются и описываются графо-аналитическими моделями (ГАМ) в виде множества вершин и дуг связи между ними. ГАМ строится на основе концептуальной двухконтурной итерационно - рекурсивной модели (IRM), позволяющей описывать как ациклические, так и циклические вычислительные процессы. Математическое описание ГАМ строится в виде кубических покрытий с использованием алгебро-топологического аппарата исчисления кубических комплексов. При построении кубических покрытий вычислительный процесс декомпозируется на множество параллельных структур с любым уровнем вложенности их друг в друга. Переход от программ к вычислительным процессам позволяет решать задачи проектирования программного продукта через верификацию в общем виде, т.е. без учета конкретных особенностей языков программирования, операционных систем и процессоров, которые могут быть как реальными, так и виртуальными. Разрабатываемые методы являются детерминированными и являются составной частью любой технологии проектирования программного продукта.

На основе разрабатываемых методов на кафедре осуществляется создание учебно-исследовательской системы (УИС) в виде САПР, позволяющей унифицировать лабораторный практикум путем создания единой базы знаний и базы данных, повысить объективность и качество оценки знаний студентов, повысить производительность работы преподавателей путем переноса основной тяжести обучения на самостоятельную работу студентов, исследовать новые методы и алгоритмы верификации вычислительных процессов.

Сотрудники кафедры принимают активное участие в международных конференциях в области искусственного интеллекта, моделирования сложных систем, теории информационных и вычислительных процессов, защиты информации и верификации. Подготовка специалистов высшей квалификации осуществляется последовательно через магистратуру, институт соискателей, аспирантуру и докторантуру. За последние десять лет подготовлено два доктора наук, более двадцати кандидатов наук.

Блохин Валентин Николаевич
Зыков Анатолий Геннадьевич
Лабода Юрий Александрович

СООБЩЕНИЯ И ПРИЛОЖЕНИЯ WINDOWS
Учебное пособие

В авторской редакции
Дизайн И.И. Иванов
Верстка П.П. Петров
Редакционно-издательский отдел Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики
Зав. РИО Н.Ф. Гусарова
Лицензия ИД № 00408 от 05.11.99
Подписано к печати <дата фактического подписания>
Заказ № <получить в РИО>
Тираж <в соответствии с планом изданий>
Отпечатано на ризографе

Редакционно-издательский отдел
Санкт-Петербургского национальный
исследовательский университет
информационных технологий
механики и оптики
197101, Санкт-Петербург, кронверкский пр., 49