КУРСОВАЯ РАБОТА

по дисциплине: «Высокоуровневые методы информатики и программирования»

на тему: Текстовый редактор

ПЛАН

1. Введение
2. Работа с текстовыми документами
3. Класс CEdit
4. Описание работы программы
5. Заключение
6. Список использованных источников
7. Приложение «Текстовый редактор»

1.ВВЕДЕНИЕ

В начале семидесятых консорциум очень умных людей разрабатывал компьютерную систему Muitix. Ее предназначением было обеспечение недорогого всеобщего доступа к графическим, текстовым и другим файлам. Конечно, это была совершенно глупая идея, и в целом проект провалился. Небольшая группа инженеров, работающих в лабораториях Белла, решила использовать фрагменты Muitix в небольшой операционной системе, которую окрестили Unix.

Эти инженеры не имели одной большой мощной машины, а лишь несколько маломощных машин разных производителей. Поскольку все они были разные, каждую программу требовалось перерабатывать под каждую машину. Чтобы избежать этих мучений, был разработан небольшой, но мощный язык, который назвали С.

Язык С оказался действительно мощным и очень скоро завоевал передовые позиции среди средств разработки программного обеспечения. Однако со временем в программировании появились новые технологии (например, достойное самого большого внимания объектно-ориентированное программирование), которые постепенно вытесняли язык С. Не желая остаться за бортом, инженерное сообщество усовершенствовало С, дополнив его новыми возможностями и получив в результате новый язык программирования C++. Язык C++ включает:

* словарь понятных для людей команд, которые конвертируются в машинный язык;
* структуру языка (или грамматику), которая позволяет пользователям составлять из команд работающие программы.

В курсовой работе по предмету ВУМИП мне было необходимо сделать текстовый редактор на языке Visual C++.

Разрабатываемый мною текстовый редактор является созданным мастером MFC Application Wizard приложением, которое поддерживает архитектуру документ-представление. Программа должна обеспечивать ввод и редактирование текста в окне представления и отображать полосы прокрутки. Меню программы должно содержать команды печати, отмены последних действий, относящихся к редактированию, команды вырезания, копирования и вставки текста, а также поиска. Для реализации операций ввода и редактирования текста внутри окна представления необходимо класс представления сделать производным от MFC-класса CEditView. При этом меню приложения будет содержать набор команд редактирования, и программисту не потребуется писать код для данных функций.

2. РАБОТА С ТЕКСТОВЫМИ ДОКУМЕНТАМИ

Специалисты Microsoft считают текстовые документы единственным достойным объектом своей работы. Вся операционная система разрабатывалась именно для работы с данным типом документов. Поэтому, рассматривая Visual C++, нельзя обойти молчанием эту область его применения.

Однако вследствие своей любви к написанию редакторов создатели Visual C++ самостоятельно разработали два типа редакторов, обеспечив их всеми необходимыми с их точки зрения функциями и, посчитав свои творения совершенством, практически исключили всякое вмешательство в их работу со стороны пользователя. Эти редакторы реализованы в объектах классов CEdit и CRichEditview. Первый из них представляет собой простейший редактор, используемый в текстовых полях диалоговых окон, и не предусматривает форматирования текста и использование различных шрифтов. Все эти возможности имеются в текстовом редакторе, реализованном в классе CRichEditview. Возможности, предоставляемые этим редактором, наглядно продемонстрированы в текстовом редакторе WordPad, являющемся примером использования данного класса, текст которого можно найти в библиотеке MSDN.

3. КЛАСС CEDIT

Объект класса CEdit обеспечивает функционирование элемента управления текстового поля Windows. Текстовое поле Windows представляет собой прямоугольное дочернее окно, в которое пользователь может вводить текст. Данный элемент управления может создаваться в шаблоне диалога или непосредственно в программе пользователя. В обоих случаях сначала вызывается конструктор класса CEdit, создающий объект данного класса, а затем вызывается функция Create, создающая текстовое поле Windows и связывающая его с объектом класса CEdit. Объект класса, производного от класса CEdit может быть создан за один шаг, если в его конструкторе будет вызвана функция Create. Объект класса CEdit наследует многие возможности объекта класса CWnd. Чтобы записать текст в объект класса CEdit, используется функция CWnd::SetWindowText, а чтобы считать его оттуда используется функция CWnd::GetWindowText. Эти функции позволяют записать или считать весь текст, содержащийся в данном элементе управления даже в том случае, когда он является многострочным. В том случае, если данный элемент управления является многострочным, в нем могут использоваться следующие функции для работы с частью содержащегося в нем текста: CEdit::GetLine, CEdit::SetSel,

CEdit::GetSel и CEdit::ReplaceSel. Чтобы обработать сообщение, посылаемое текстовым полем своему родительскому окну (обычно это объект класса, производного от CDialog), добавьте соответствующий макрос в карту сообщений и создайте функцию для обработки данного сообщения. Макрос карты сообщений для данного типа сообщений имеет следующий формат:

ON\_Notification(id, memberFxn)

где id - идентификатор дочернего окна элемента управления, посылающего сообщение, а memberFxn - имя функции обработки данного сообщения в классе родительского окна. Прототип функции обработки сообщения имеет следующий формат:

afx\_msg void memberFxn();

Ниже приведен список макросов карты сообщений, которые могут использоваться для обработки сообщений, посылаемых данным элементом управления:

ON\_EN\_CHANGE - пользователь произвел действие, которое может привести к изменению текста, содержащегося в текстовом поле. В отличие от сообщения EN\_UPDATE данное сообщение посылается после того, как Windows произведет обновление экрана;



ON\_EN\_ERRSPACE - для данного текстового поля не может быть выделен необходимый для него объем памяти;



ON\_EN\_HSCROLL - пользователь воспользовался горизонтальной полосой прокрутки данного элемента управления. Сообщение посылается родительскому окну до обновления экрана;



ON\_EN\_KILLFOCUS - данное текстовое поле потеряло фокус ввода;



ON\_EN\_MAXTEXT - текущая вставка привела к превышению определенного в данном объекте максимального числа символов, что привело к ее урезанию. Данное сообщение посылается также в том случае, если текстовое поле не имеет стиля ES\_AUTOHSCROLL, а количество символов в текущей строке превышает ширину текстового поля. Другим случаем, когда посылается данное сообщение, является случай, когда текстовое поле не имеет стиля ES\_AUTOVSCROLL, а количество строк в нем превышает высоту текстового поля, или же текстовое поле не имеет стиля ES\_AUTOHSCROLL, а количество символов в текущей строке превышает ширину текстового поля;



ON\_EN\_SETFOCUS - данное текстовое поле получило фокус ввода;



ON\_EN\_UPDATE - в текстовом поле будет выводиться измененный текст. Посылается после того, как элемент управления отформатирует текст, но до того, как этот текст будет выведен в текстовое поле, что позволяет изменить размеры текстового поля в случае необходимости;



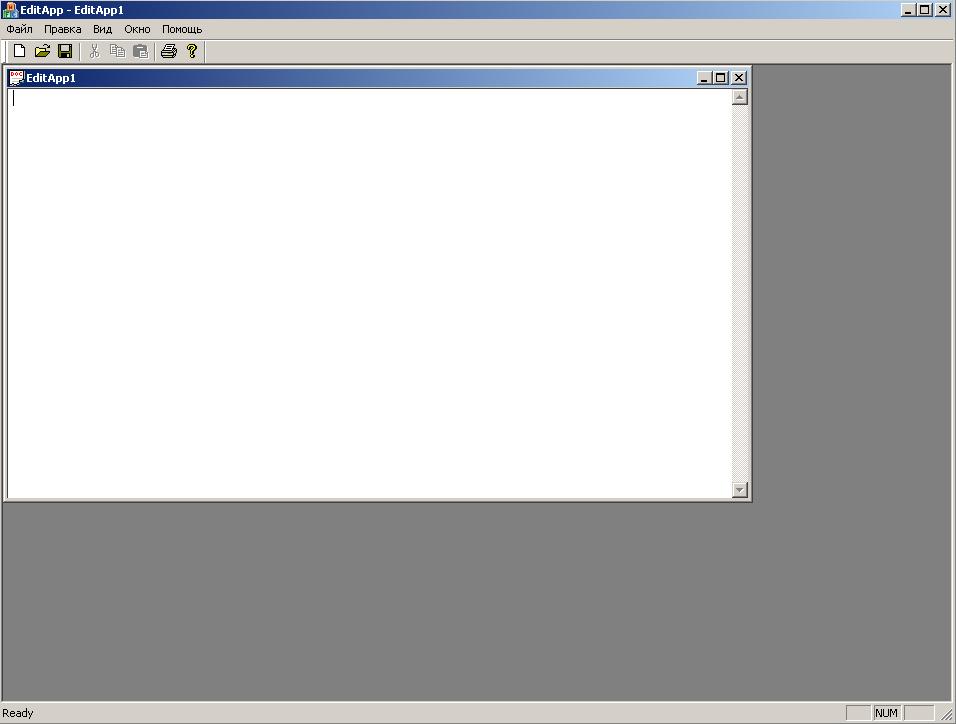
ON\_EN\_VSCROLL - пользователь воспользовался вертикальной полосой прокрутки данного элемента.



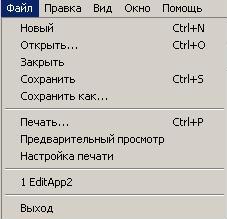
При создании объекта класса CEdit в диалоговом окне этот объект автоматически уничтожается при закрытии диалогового окна. То же самое происходит и в том случае, когда объект класса CEdit создается в шаблоне диалога. Если объекта класса CEdit создается в окне, то от пользователя может потребоваться его уничтожить. Если же объект класса CEdit создается в стеке, то он уничтожается автоматически. При создании объекта класса CEdit в куче с использованием оператора new его необходимо уничтожить после завершения работы пользователя с элементом управления Windows с использованием оператора delete. Если в объект класса, производного от CEdit, была распределена какая-либо память, то необходимо перегрузить деструктор данного класса таким образом, чтобы он освобождал эту память. Описание данного класса содержится в файле заголовка afxwin.h.

4. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

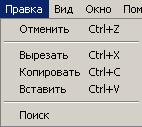
После запуска программы появляется окно «EditApp». Так выглядит внешний вид программы.



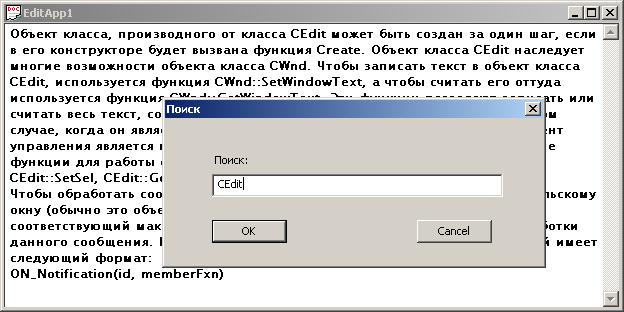
Меню Файл имеет следующие пункты: Новый, Открыть, Закрыть, Сохранить, Сохранить как, Печать, Предварительный просмотр, Настройка печати, Выход.



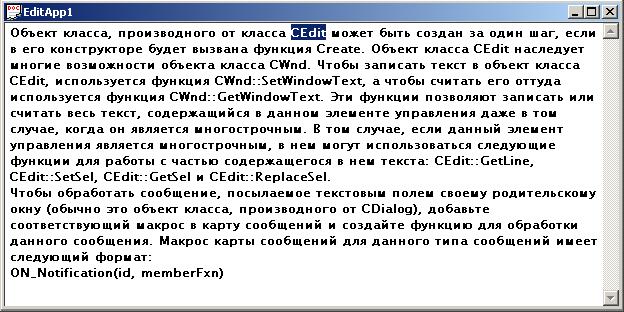
Меню Правка имеет следующие пункты: Отменить, Вырезать, Копировать, Вставить, Поиск.



Поиск: Выбрав это меню выскочить окно поиска, куда нужно ввести искомое слово и нажать ОК.



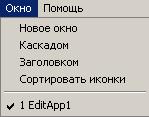
Результат поиска



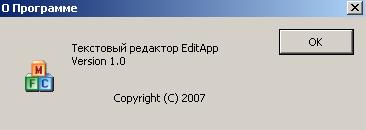
Меню Вид имеет следующие пункты: Панель инструментов, Строка состояния, т.е по желанию их можно скрыть.



Меню Окно имеет следующие пункты: Новое окно, Каскадом, Заголовками, Сортировать иконки, за разделителем располагаются имена открытых файлов, а галочкой помечается тот файл, с которым работают. При создание нового документа окно открывается прямо в программе, а не в новом файле. По желанию пользователь может расположить их каскадом или заголовками для удобства работы.



Меню Помощь содержит информацию о программе.



5. ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы предоставленная программа была выполнена и работала. Конечно это простой текстовый редактор и у него мало функций по сравнению с MS WORD. Он чем-то похож на простой блокнот.

Единственное его отличие что он был написан на языке С++. Язык С++ работал без наладок, был прост в обращении, возникающие ошибке при программировании легко исправлялись. С++ зарекомендовал себя с лучшей стороны, и я бы порекомендовал его как для начинающих программистов так и для профессионалов.

6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эффективная работа: Visual C++. Net. – СПб.: Питер, 2002. – 816 с.: ил.
2. C++, Объективно – ориентированное программирование: Практикум. – СПб.: Питер, 2004. – 265 с.: ил.
3. C/C++. Программирование на языке высокого уровня – СПб.: Питер, 2005. – 461 с.: ил.
4. Круглински Д.Д., Уингоу С., Шеферд Д. Программирование на Microsoft Visual C++ 6.0 : Пер. с англ..-СПб.- М. Харьков - Минск: Питер; Русская редакция, 2000.-821 с.: ил. + CD-ROM.

7. ПРИЛОЖЕНИЕ

Файл EDitApp.cpp. Этот файл является основным файлом приложения.Он включает в себя определение класса CEditApp.

#include "stdafx.h"

#include "EditApp.h"

#include "MainFrm.h"

#include "ChildFrm.h"

#include "EditAppDoc.h"

#include "EditAppView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CEditAppApp

BEGIN\_MESSAGE\_MAP(CEditAppApp, CWinApp)

ON\_COMMAND(ID\_APP\_ABOUT, &CEditAppApp::OnAppAbout)

// Standard file based document commands

ON\_COMMAND(ID\_FILE\_NEW, &CWinApp::OnFileNew)

ON\_COMMAND(ID\_FILE\_OPEN, &CWinApp::OnFileOpen)

// Standard print setup command

ON\_COMMAND(ID\_FILE\_PRINT\_SETUP, &CWinApp::OnFilePrintSetup)

END\_MESSAGE\_MAP()

// CEditAppApp construction

CEditAppApp::CEditAppApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

// The one and only CEditAppApp object

CEditAppApp theApp;

// CEditAppApp initialization

BOOL CEditAppApp::InitInstance()

{

// InitCommonControlsEx() is required on Windows XP if an application

// manifest specifies use of ComCtl32.dll version 6 or later to enable

// visual styles. Otherwise, any window creation will fail.

INITCOMMONCONTROLSEX InitCtrls;

InitCtrls.dwSize = sizeof(InitCtrls);

// Set this to include all the common control classes you want to use

// in your application.

InitCtrls.dwICC = ICC\_WIN95\_CLASSES;

InitCommonControlsEx(&InitCtrls);

CWinApp::InitInstance();

// Initialize OLE libraries

if (!AfxOleInit())

{

AfxMessageBox(IDP\_OLE\_INIT\_FAILED);

return FALSE;

}

AfxEnableControlContainer();

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need

// Change the registry key under which our settings are stored

// TODO: You should modify this string to be something appropriate

// such as the name of your company or organization

SetRegistryKey(\_T("Local AppWizard-Generated Applications"));

LoadStdProfileSettings(4); // Load standard INI file options (including MRU)

// Register the application's document templates. Document templates

// serve as the connection between documents, frame windows and views

CMultiDocTemplate\* pDocTemplate;

pDocTemplate = new CMultiDocTemplate(IDR\_EditAppTYPE,

RUNTIME\_CLASS(CEditAppDoc),

RUNTIME\_CLASS(CChildFrame), // custom MDI child frame

RUNTIME\_CLASS(CEditAppView));

if (!pDocTemplate)

return FALSE;

AddDocTemplate(pDocTemplate);

// create main MDI Frame window

CMainFrame\* pMainFrame = new CMainFrame;

if (!pMainFrame || !pMainFrame->LoadFrame(IDR\_MAINFRAME))

{

delete pMainFrame;

return FALSE;

}

m\_pMainWnd = pMainFrame;

// call DragAcceptFiles only if there's a suffix

// In an MDI app, this should occur immediately after setting m\_pMainWnd

// Parse command line for standard shell commands, DDE, file open

CCommandLineInfo cmdInfo;

ParseCommandLine(cmdInfo);

// Dispatch commands specified on the command line. Will return FALSE if

// app was launched with /RegServer, /Register, /Unregserver or /Unregister.

if (!ProcessShellCommand(cmdInfo))

return FALSE;

// The main window has been initialized, so show and update it

pMainFrame->ShowWindow(m\_nCmdShow);

pMainFrame->UpdateWindow();

return TRUE;

}

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog

{

public:

CAboutDlg();

// Dialog Data

enum { IDD = IDD\_ABOUTBOX };

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

// Implementation

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialog)

END\_MESSAGE\_MAP()

// App command to run the dialog

void CEditAppApp::OnAppAbout()

{

CAboutDlg aboutDlg;

aboutDlg.DoModal();

}

// CEditAppApp message handlers

Первая карта сообщений (конструкция BEGIN\_MESSAGE\_MAP

END\_MESSAGE\_MAP) принадлежит классу СEditApp. Вней сообщения с индетификатором ID\_APP\_ABOUT, ID\_FILE\_NEW, ID\_FILE\_OPEN,

ID\_FILE\_PRINT\_SETUP связываются соответственно с обработчиками OnAppAbout(), CWinApp::OnFileNew(), CWinApp::OnFileOpen(), CWinApp::OnFilePrintSetup(). В этом файле реализуется конструктор класса СEditApp, а также его методы OnAppAbout() и InitInstance().

Файл MainFrm.cpp. Этот файл содержит реализацию класса CMainFrame, которой порождается от класса CFrameWnd и управляет всеми дочерними MDI-окнами.

#include "stdafx.h"

#include "EditApp.h"

#include "MainFrm.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CMainFrame

IMPLEMENT\_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN\_MESSAGE\_MAP(CMainFrame, CMDIFrameWnd)

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 (CMDIFrameWnd::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::PreCreateWindow(CREATESTRUCT& cs)

{

if( !CMDIFrameWnd::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

{

CMDIFrameWnd::AssertValid();

}

void CMainFrame::Dump(CDumpContext& dc) const

{

CMDIFrameWnd::Dump(dc);

}

#endif //\_DEBUG

// CMainFrame message handlers

В массиве indicators перечислены идентификаторы полей строки состояния, которые служат индикаторами нажатия некоторых клавиш. Добавление в окно приложения панели инструментов и строки состояния производится выделенным полужирным шрифтом. Функции-члены AssertValid() и

Dump() используют объявления содержащиеся родительском классе. Класс

CMainFrame изначально не имеет обработчиков сообщений.

Файл EditAppDoc.cpp. Этот файл содержит реализацию класса CEditAppDoc, Который управляет работой с конкретными документами, а также обеспечивает загрузку и сохранение файлов.

#include "stdafx.h"

#include "EditApp.h"

#include "EditAppDoc.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CEditAppDoc

IMPLEMENT\_DYNCREATE(CEditAppDoc, CDocument)

BEGIN\_MESSAGE\_MAP(CEditAppDoc, CDocument)

END\_MESSAGE\_MAP()

// CEditAppDoc construction/destruction

CEditAppDoc::CEditAppDoc()

{

// TODO: add one-time construction code here

}

CEditAppDoc::~CEditAppDoc()

{

}

BOOL CEditAppDoc::OnNewDocument()

{

if (!CDocument::OnNewDocument())

return FALSE;

// TODO: add reinitialization code here

// (SDI documents will reuse this document)

return TRUE;

}

// CEditAppDoc serialization

void CEditAppDoc::Serialize(CArchive& ar)

{

// CEditView contains an edit control which handles all serialization

reinterpret\_cast<CEditView\*>(m\_viewList.GetHead())->SerializeRaw(ar);

}

// CEditAppDoc diagnostics

#ifdef \_DEBUG

void CEditAppDoc::AssertValid() const

{

CDocument::AssertValid();

}

void CEditAppDoc::Dump(CDumpContext& dc) const

{

CDocument::Dump(dc);

}

#endif //\_DEBUG

// CEditAppDoc commands

Некоторые методы класса CEditAppDoc могут применяться для поддержки самых необходимых операций работы с документами. В функции OnNewDocument() используются эта же функция из родительского класса

<CEditView\*>(m\_viewList.GetHead())->SerializeRaw(ar);

Эта строка кода поддерживает работу команд меню File, обеспечивающих создание, открытие и сохранение файлов. Функции-члены AssertValid() и

Dump() используют описание, предлагаемое родительским классом.

Файл CEditAppView.cpp. Этот файл содержит реализацию класса CEditAppView, который порождается от класса CEditView и управляет отображением документа.

#include "stdafx.h"

#include "EditApp.h"

#include "SeekDialog.h"

#include "EditAppDoc.h"

#include "EditAppView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CEditAppView

IMPLEMENT\_DYNCREATE(CEditAppView, CEditView)

BEGIN\_MESSAGE\_MAP(CEditAppView, CEditView)

// Standard printing commands

ON\_COMMAND(ID\_FILE\_PRINT, &CEditView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_DIRECT, &CEditView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_PREVIEW, &CEditView::OnFilePrintPreview)

ON\_COMMAND(ID\_SEEK, &CEditAppView::OnSeek)

END\_MESSAGE\_MAP()

// CEditAppView construction/destruction

CEditAppView::CEditAppView()

{

// TODO: add construction code here

}

CEditAppView::~CEditAppView()

{

}

BOOL CEditAppView::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: Modify the Window class or styles here by modifying

// the CREATESTRUCT cs

BOOL bPreCreated = CEditView::PreCreateWindow(cs);

cs.style &= ~(ES\_AUTOHSCROLL|WS\_HSCROLL); // Enable word-wrapping

return bPreCreated;

}

// CEditAppView printing

BOOL CEditAppView::OnPreparePrinting(CPrintInfo\* pInfo)

{

// default CEditView preparation

return CEditView::OnPreparePrinting(pInfo);

}

void CEditAppView::OnBeginPrinting(CDC\* pDC, CPrintInfo\* pInfo)

{

// Default CEditView begin printing

CEditView::OnBeginPrinting(pDC, pInfo);

}

void CEditAppView::OnEndPrinting(CDC\* pDC, CPrintInfo\* pInfo)

{

// Default CEditView end printing

CEditView::OnEndPrinting(pDC, pInfo);

}

// CEditAppView diagnostics

#ifdef \_DEBUG

void CEditAppView::AssertValid() const

{

CEditView::AssertValid();

}

void CEditAppView::Dump(CDumpContext& dc) const

{

CEditView::Dump(dc);

}

CEditAppDoc\* CEditAppView::GetDocument() const // non-debug version is inline

{

ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CEditAppDoc)));

return (CEditAppDoc\*)m\_pDocument;

}

#endif //\_DEBUG

// CEditAppView message handlers

void CEditAppView::OnSeek()

{

// TODO: Add your command handler code here

CSeekDialog dlg(this);

if( dlg.DoModal() == IDOK){

FindText( dlg.m\_Sample);

}

}

Класс CEditAppView управляет печатью документов с помощью функций OnPreparePrinting(), OnBeginPrinting(),OnEndPrinting(). Выделенный полужирным шрифтом тектс отвечает за поиск слов в тексте.