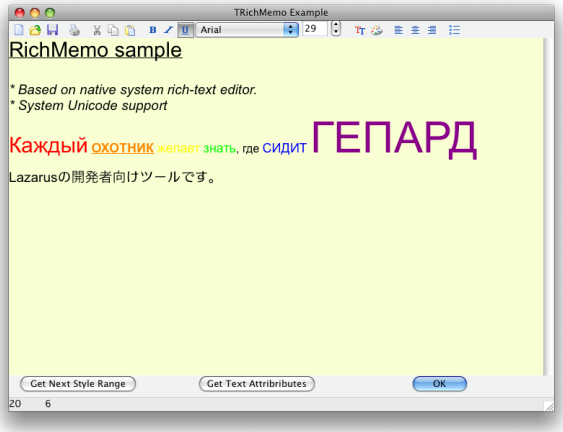
RichMemo/ru

| English (en) | polski (pl) | русский (ru) |

RichMemo - это пакет, который включает компонент для замены компонента Delphi TRichEdit. Он разработан кросс-платформенным способом, поэтому его реализация возможна для следующих платформ: Win32, MacOSX и Linux. Поскольку кроссплатформенность является основной целью, нативный API RichMemo может быть расширен, чтобы сделать его совместимым с Delphi RichEdit.



RichMemo macosx screenshot. Thanks to Dominique Louis

Его основными характеристиками являются:

- Подсветка текста
- Системная поддержка редактирования Юникода

Планируется: (исправления приветствуются)

- Добавление изображений в текст
- Встраивание элементов управления LCL?

Загружаемый файл содержит компонент, установочный пакет и демонстрационное приложение, которое иллюстрирует возможности компонента, а также некоторые инструменты для оценки диаграммы в данной системе.

Пожалуйста, добавляйте ваши багрепорты/функции на Github .

Contents [hide]

Лицензия

Автор: Дмитрий 'скалогрыз' Бояринцев

modified LGPL (так же, как FPC RTL и Lazarus LCL). Вы можете связаться с автором, если измененный LGPL не работает с вашим проектом лицензирования.

Загрузка

Последняя версия доступна здесь:

```
https://github.com/skalogryz/richmemo
```

Прим.перев.: Существует также форк данного компонента на гитхабе от Andreas Toth aka mrandreastoth.

Change Log

- Version 1.0.0 22th Jun 2009

- Version 0.8.0 Jun 2009

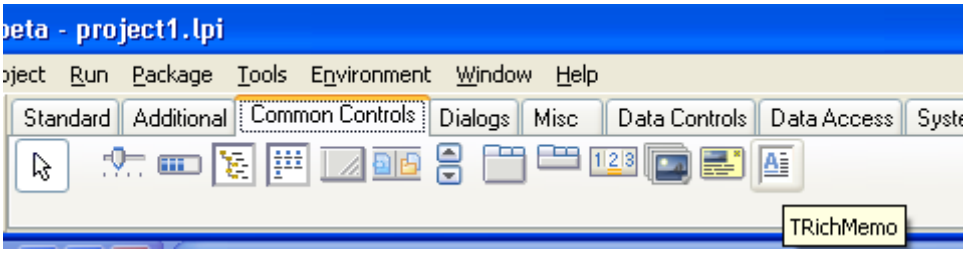
Зависимости / Системные требования

- Lazarus 1.0.0

Статус: выпущен.

Установка

- Скачайте исходники пакета
- Установите пакет в Lazarus и пересоберите его
- TRichMemo добавится в закладку 'Common Controls'.

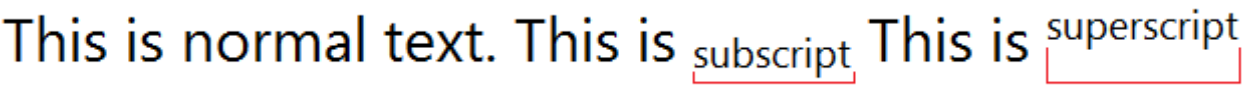


TRichMemo

Параметры шрифта

Параметры шрифта обычно представлены записью `TFontParams`. Тип используется для описания атрибутов шрифта **rich**. Некоторые атрибуты выходят за рамки класса `TFont`, поэтому необходим дополнительный тип. Однако большинство типов данных, связанных с `TFont`, используются повторно (т.е. `TFontStyles`).

- `Name` - название шрифта (семейство).
- `Size` - размер шрифта в пунктах (передача отрицательного значения может привести к неожиданным результатам).
- `Color` - цвет шрифта.
- `Style` - стили шрифта, включая полужирный, курсив, зачеркнутый, подчеркивание.
- `HasBkClr` - логический флаг, если текст для должен иметь цвет фона (если true, поле BkColor используется, в противном случае BkColor игнорируется). Прозрачность не поддерживается.
- `BkColor` - цвет фона (или подсветка) для текста.
- `VScriptPos` - вертикальная настройка текста .



  - `vpNormal` - нет особой позиции.
  - `vpSubscript` - текст - нижний индекс.
  - `vpSuperscript` - текст - верхний индекс.

  Фактическое вертикальное смещение индексов верх/низ зависит от реализации ОС и в настоящее время не может контролироваться.

Вы можете присвоить `FontParams` из структуры `TFont`. Для этого вам нужно использовать функцию `GetFontParams(afont: TFont)`. Обратите внимание, что для большинства элементов управления LCL для `TFont` установлены значения по умолчанию. Это не фактические значения, а скорее указание на то, что следует использовать шрифт по умолчанию для элемента управления (аналогично `crDefault` для `TColor`).

Функция `GetFontParams` разрешает имя шрифта по умолчанию и возвращает фактическое имя семейства шрифтов.

Методы

SetTextAttributes

```
procedure SetTextAttributes(TextStart, TextLen: Integer; AFont: TFont);
```

- `TextStart : Integer` - первый символ, который будет изменен
- `TextLen : Integer` - количество символов для изменения
- `AFont : TFont` - шрифт, который должен быть применен к части текста

```
procedure SetTextAttributes(TextStart, TextLen: Integer; const TextParams: TFontParams);
```

- `TextStart : Integer` - первый символ, который будет изменен
- `TextLen : Integer` - количество символов для изменения
- `TextParams : TFontParams` - параметры шрифта, которые будут установлены

Методы `SetTextureAttributes` изменяют указанный стиль текстового диапазона. Параметры шрифта передаются в обоих методах, параметром `AFont` (объект LCL `TFont`) или `TFontParams` (объявлено в RichMemo)

Установка атрибутов текста не меняет текущий выбор. Если необходимо изменить стиль выделенного в данный момент текста, вам следует выбрать `SelStart` и `SelLength` в качестве значений текстового диапазона:

```
RichMemo1.SetTextAttributes(RichMemo1.SelStart, RichMemo1.SelLength, FontDialog1.Font);
```

GetTextAttributes

```
function GetTextAttributes(TextStart: Integer; var TextParams: TFontParams): Boolean;
virtual;
```

- `TextStart : Integer` - позиция символа для запроса параметров шрифта
- `var TextParams : TFontParams` - выходное значение, заполненное атрибутами символа шрифта. Если метод завершается неудачно и возвращает `false`, значения поля записи будут неопределенными.

Заполняет параметры шрифта символа в позиции `TextStart`. Метод возвращает `True`, если `TextStart` является действительной позицией символа, и `False` в противном случае.

GetStyleRange

```
function GetStyleRange(CharPos: Integer; var RangeStart, RangeLen: Integer): Boolean;
virtual;
```

Возвращает диапазон символов с одинаковыми параметрами шрифта, т.е. все символы в диапазоне имеют одинаковые имя, размер, цвет и стили шрифта.

- `CharPos : Integer` - символ, который принадлежит диапазону стилей. Позиция не обязательно быть в начале диапазона стилей. Она может быть в середине конца диапазона стилей. Первая позиция символа возвращается параметром `RangeStart`.
- `var RangeStart : Integer` - первый символ в диапазоне
- `var RangeLen : Integer` - количество символов в диапазоне

Метод возвращает `true` в случае успеха. Метод возвращает `false`, если `CharPos` имеет неправильное значение - больше доступных символов или какая-либо другая ошибка.

CharAtPos

```
function CharAtPos(x,y: Integer): Integer; virtual;
```

Возвращает смещение символа от нулевой позиции (не позиция UTF8). Возвращает -1 в случае неудачи.

- `x, y` - точка в клиентской области элемента управления RichMemo.

Метод знает о состоянии прокрутки элемента управления. Таким образом, два вызова `CharAtPos(0,0)` могут возвращать разные значения, если между вызовами изменяется позиция прокрутки. Смотрите "examples/hittest" для [получения] примера использования.

Если указанные `x,y` будут вне содержимого RichMemo, возвращаемое значение [будет] неопределенным. [Поведение] остается [на усмотрение] набора виджетов, который либо возвращает -1, либо закрывает доступный символ.

SetRangeColor

```
procedure SetRangeColor(TextStart, TextLength: Integer; FontColor: TColor);
```

Метод устанавливает цвет символов в указанном диапазоне для FontColor. Другие параметры шрифта (имя, размер, стили) остаются без изменений.

- `TextStart: Integer` - первый символ в диапазоне
- `TextLength: Integer` - количество символов в диапазоне
- `FontColor: TColor` - цвет, который должен быть установлен

SetRangeParams

```
    procedure SetRangeParams(TextStart, TextLength: Integer; ModifyMask:
TTextModifyMask;
      const fnt: TFontParams; AddFontStyle, RemoveFontStyle: TFontStyles); overload;
```

Метод изменяет параметры шрифта в указанном диапазоне.

- `TextStart` - первый символ в диапазоне
- `TextLength` - количество символов в диапазоне
- `ModifyMask` - показывает, какие именно атрибуты шрифта должны быть обновлены. Маска принимает любую комбинацию из следующих значений:
  - `tmm_Color` - Цвет шрифта будет изменен (поле `fnt.Color` должно быть заполнено)
  - `tmm_Name` - имя шрифта будет изменено (поле `fnt.Name` должно быть заполнено)
  - `tmm_Size` - размер шрифта будет изменен (поле `fnt.Size` должно быть заполнено)
  - `tmm_Styles` - стили шрифта будут изменены в соответствии с параметрами `AddFontStyle, RemoveFontStyle`
  - `tmm_BackColor` - Цвет подсветки текста (фона) будет изменен. (поля `fnt.HasBkClr` и `fnt.BkColor` должны быть заполнены)
  - Отправка пустой маски приведет к возврату метода без внесения каких-либо изменений.
- `fnt` - Параметры шрифта, которые будут использоваться, в зависимости от значений `ModifyMask`.
- `AddFontStyle` - набор стилей, которые должны применяться к диапазону (используется только если `tmm_Styles` в `ModifyMask`, в противном случае игнорируется)
- `RemoveFontStyle` - набор стилей шрифта должен быть удален из диапазона (используется только если `tmm_Styles` в `ModifyMask`, в противном случае игнорируется)

```
procedure SetRangeParams(TextStart, TextLength: Integer;
    ModifyMask: TTextModifyMask; const FontName: String;
    FontSize: Integer; FontColor: TColor;
    AddFontStyle, RemoveFontStyle: TFontStyles);
```

Перегруженная версия метода. Это просто оболочка вокруг параметра, использующая структуру `TFontParams`. Метод не поддерживает изменение цвета фона, вы должны использовать версию `TFontParams`

- `FontName` - имя шрифта, которое должно быть установлено (используется только если `tmm_Name` в `ModifyMask`, в противном случае игнорируется)
- `FontColor` - Цвет шрифта, который должен быть установлен (используется только если `tmm_Color` в `ModifyMask`, в противном случае игнорируется)

Например, если используется следующий код

```
  RichMemo1.SetRangeParams (
     RichMemo1.SelStart, RichMemo1.SelLength,
     [tmm_Styles, tmm_Color], // только изменение цвета и стиля
     '',  // имя шрифта - оно не используется, поэтому мы можем оставить его пустым
     0,  // размер шрифта - это размер шрифта, мы можем оставить его пустым
     clGreen, // сделать весь текст в выбранной области зеленым цветом
     [fsBold, fsItalic],  // добавление жирного и курсивного стилей
     []
  );
```

GetParaAlignment

Получает выравнивание абзаца

```
function GetParaAlignment(TextStart: Integer; var AAlign: TParaAlignment): Boolean;
```

- `TextStart` - позиция символа, который принадлежит абзацу
- `AAlign` - получение выравнивания абзаца.

Примечание:

- Win32 - выравнивание по ширине не работает в Windows XP и более ранних версиях.
- OSX - из-за ограничений Carbon, это не работает в Carbon, но работает для виджетов Cocoa.

SetParaAlignment

Устанавливает выравнивание абзаца

```
procedure SetParaAlignment(TextStart, TextLen: Integer; AAlign: TParaAlignment);
```

Примечание:

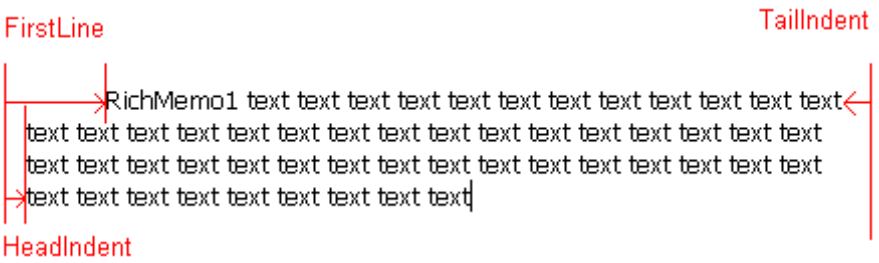
- Win32 - выравнивание по ширине не работает в Windows XP и более ранних версиях.
- OSX - из-за ограничений Carbon, это не работает в Carbon, но работает для виджетов Cocoa. Также необходимо, чтобы правильное выравнивание было загружено из файла RTF.

GetParaMetric

Возвращает отступ метрики абзаца для данного абзаца

- `FirstLine` - смещение для первой строки (в пунктах) абзаца от начала элемента управления [(т.н. "красная строка")]
- `TailIndent` - смещение для каждой строки (в пунктах), за исключением первой строки, абзаца от конца элемента управления

- `HeadIndent` - смещение для каждой строки (в пунктах) абзаца от начала элемента управления
- `SpaceBefore` - дополнительное место перед абзацем (в пунктах)
- `SpaceAfter` - дополнительное место после абзаца (в пунктах)



- `LineSpacing` - коэффициент, используемый для расчета межстрочного интервала между строками в абзаце. Коэффициент применяется к размеру шрифта (самый высокий в строке), а не к высоте строки. Таким образом, это соответствует свойству CSS line-height. Обратите внимание, что нормальный межстрочный интервал равен 1,2. То есть шрифт размером 12 пт, фактический межстрочный интервал (равный 1,2) приведет к высоте строки 14 пт.
  - не поддерживается в WinXP или более ранних версиях
  - в большинстве систем не будет разрешено, если для межстрочного интервала установлено значение меньше 1,2. Значение будет проигнорировано, и оно по умолчанию будет равно 1,2

Примечание

- Параметры абзаца RichEdit указываются в пикселях, а не в точках, поэтому вам нужно либо преобразовать эти размеры самостоятельно, либо использовать метод RichMemoHelper.
- обозначение смещений «влево»/«вправо» исключено, чтобы избежать путаницы для RTL.

```
function GetParaMetric(TextStart: Integer; var AMetric: TParaMetric): Boolean;
```

SetParaMetric

Устанавливает метрики абзаца

```
procedure SetParaMetric(TextStart, TextLen: Integer; const AMetric: TParaMetric);
```

Рекомендуется использовать функцию `InitParaMetric` для инициализации структуры `TParaMetric`. В противном случае обнуление структуры равно ее размеру (т.е. достаточно `FillChar(m, sizeof(TParaMetric), 0)`. Если значения `TParaMetric` получены из вызова `GetParaMetric`, то структура считается правильно инициализированной).

GetParaRange

Возвращает диапазон символов для абзаца. Абзац идентифицируется символом.

```
  function GetParaRange(CharOfs: Integer; var ParaRange: TParaRange): Boolean;
  function GetParaRange(CharOfs: Integer; var TextStart, TextLength: Integer): Boolean;
```

`CharOfs` - символ, который принадлежит абзацу, диапазон которого возвращается. `TParaRange` - это структура, которая возвращает 3 поля

- `start` - первый символ в абзаце
- `lengthNoBr` - длина абзаца, за исключением символа разрыва строки
- `length` - длина абзаца, включая разрыв строки, если присутствующая последняя строка в элементе управления не содержит символа перевода строки, следовательно, `длина = lengthNoBr`.

SetParaTabs

Устанавливает набор табуляций для абзацев указанной длины.

```
procedure SetParaTabs(TextStart, TextLen: Integer; const AStopList: TTabStopList);
```

- TextStart
- TextLen
- AStopList - структура, которая содержит массив табуляций.
  - Count - указывает количество инициализированных элементов.
  - Tabs - массив содержит описание каждой позиции табуляции, состоящее из Offset и Alignment.
    - Offset - смещение точки табуляции в пунктах (с левой стороны элемента управления)
    - Align - выравнивание позиции табуляции (tabLeft - по умолчанию, tabCenter, tabRight, tabDecimal).

Примечание

Renamefest

до r4140 перечисляемыми значениями TTabAlignment были taLeft, taCenter, taRight, taDecimal, но они вступали в противоречие с объявлениями Classes.TAlignment. Таким образом суффикс был изменен.

win32

позволяет не более 32 табуляций

gtk2

поддерживается только выравнивание tabLeft

carbon

не реализовано

GetParaTabs

Получает массив табуляции. Если конкретные вкладки не указаны (по умолчанию виджет табуляции), счетчик AStopList будет установлен в 0.

```
function GetParaTabs(CharOfs: Integer; var AStopList: TTabStopList): Boolean;
```

SetRangeParaParams

Выборочно устанавливает метрики абзаца

```
procedure SetRangeParaParams(TextStart, TextLength: Integer; ModifyMask:
TParaModifyMask; const ParaMetric: TParaMetric);
```

- TextStart - символ первого абзаца, чтобы тоже применить стиль
- TextLength - количество символов, которые должны изменить свойства абзаца
- ModifyMask - Маска определяет, какой показатель должен быть установлен для каждого абзаца в блоке TextStart..TextLength.Другие символы останутся без изменений. Являются членами набора
  - pmm_FirstLine - изменяет отступ первой строки. Поле FirstLine константы ParaMetric должно быть инициализировано
  - pmm_HeadIndent - изменяет отступ абзаца с головы. Поле HeadIndent константы ParaMetric должно быть инициализировано
  - pmm_TailIndent - изменяет отступ абзаца с хвоста. Поле TailIndent константы ParaMetric должно быть инициализировано
  - pmm_SpaceBefore - меняет пробел до абзаца. Поле SpaceBefore константы ParaMetric должно быть инициализировано
  - pmm_SpaceAfter - меняет пробел после (ниже) абзаца. Поле SpaceAfter константы ParaMetric должно быть инициализировано

    - `pmm_LineSpacing` - межстрочный интервал внутри абзаца. Поле `LineSpace` константы `ParaMetric` должно быть инициализировано
- `ParaMetric` - структура, содержащая значения, которые будут установлены. Только поля, указанные [структурой] `ModifyMask` должны быть инициализированы

LoadRichText

```
function LoadRichText(Source: TStream): Boolean; virtual;
```

- `Source: TStream` - поток для чтения данных из форматированного текста

Метод загружает данные в формате RTF из указанного потока. Возвращает true в случае успеха и false в противном случае. Если source равно nil, метод возвращает false.

Содержимое TRichMemo полностью заменяется содержимым в исходном потоке. Текущий выбор текста сбрасывается.

SaveRichText

```
function SaveRichText(Dest: TStream): Boolean; virtual;
```

- `Source: TStream` - поток для записи данных форматированного текста

Метод сохраняет RTF-кодированные данные в указанном потоке. Возвращает true в случае успеха и false в противном случае. Если source равно nil, метод возвращает false.

Текущее состояние TRichMemo не меняется после возврата метода.

```
function Search(const ANiddle: string; Start, Len: Integer; const SearchOpt:
TSearchOptions): Integer;
```

- `ANiddle: string` - текст для поиска.
- `Start: Integer` - позиция символа, с которой начинается поиск, если -1 или 0 - поиск сначала.
- `Len: Integer` - длина (в символах, а не в байтах) текста для поиска посередине.
- `SearchOpt: TSearchOptions`
    - `soMatchCase` - поиск с учетом регистра.
    - `soBackward` - поиск от конца документа до символа, обозначенного параметром "Start".
    - `soWholeWord` - поиск всего слова.

Метод возвращает положение символа найденной подстроки. Позиция подходит для использования в свойстве `SelStart`. Тем не менее, может быть невозможно использовать для операции прямого копирования, если в коде присутствуют символы Unicode. Вместо этого следует использовать UTF8Copy.

Если `ANiddle` не был найден в тексте, возвращается -1

```
function Search(const ANiddle: string; Start, Len: Integer; const SearchOpt:
TSearchOptions;
  var TextStart, TextLength: Integer): Boolean;
```

(Метод введен с r5115)

- `ANiddle: string` - текст для поиска.
- `Start: Integer` - позиция символа, с которой начинается поиск, если -1 или 0 - поиск сначала.
- `Len: Integer` - длина (в символах, а не в байтах) текста для поиска посередине.
- `SearchOpt: TSearchOptions`

- soMatchCase - поиск с учетом регистра
- soBackward - поиск от конца документа до символа, обозначенного параметром "Start".
- soWholeWord - поиск всего слова.

- (output) ATextStart - позиция найденного текста (в символах)
- (output) ATextLength - длина найденного текста (в символах (позиции курсора!))

Метод возвращает значение true, если искомая подстрока найдена, в противном случае - значение false.

Возвращенные значения ATextStart и ATextLength подходят для использования в свойствах SelStart и SelLength.

Примечание: для сложных сценариев найденный текст может совпадать с искомой подстрокой. Поведение поиска (в сложных сценариях) зависит от набора виджетов. Таким образом, поиск одной и той же строки в разных наборах виджетов (например, win32 или gtk2) может привести к разным результатам. Если вам нужны одинаковые результаты и вы их не получили, создайте отчет об ошибке.

Если вам нужно извлечь найденный текст, вы должны использовать метод GetText() (передавая значения ATextStart и ATextLength).

GetText, GetUText

```
function GetText(TextStart, TextLength: Integer): String;
function GetUText(TextStart, TextLength: Integer): UnicodeString;
```

- TextStart: Integer -позиция символа для начала извлечения (0 - это первый символ в тексте).
- TextLength: Integer - длина (в символах, а не в байтах) текста для извлечения

GetText() возвращает подстроку UTF8

GetUText() возвращает подстроку UTF16

Текущее выделение не будет зависеть от операции. (Если вы видите, что выделение пострадало, пожалуйста, сообщите о проблеме).

Вы не должны учитывать [абстрактную] эффективность любого метода. Например, WinAPI внутренне работает с символами UTF16, поэтому GetUText() **может** быть более эффективным для него. В то время как Gtk2 работает с UTF8 и вызов GetText() **может** быть более эффективным для него. Вместо того, чтобы думать о базовой системе, вы должны учитывать потребности вашей задачи.

Redo

```
procedure Redo;
```

Метод восстанавливает последнее предыдущее изменение undone .

Вы всегда можете вызвать для проверки CanRedo, если есть какие-либо действия[, которые] могут быть переделаны.

Свойства

ZoomFactor

```
property ZoomFactor: double
```

Read/Write свойство. Управляет масштабированием содержимого RichMemo. 1.0 - без увеличения. Менее < 1,0 - уменьшить масштаб, более 1,0 - увеличить масштаб. Если установлено 0 - [приводится к] значению по умолчанию равным 1,0 коэффициенту масштабирования (без увеличения).

HideSelection

```
property HideSelection: Boolean default false
```

Read/Write свойство. Если True, то выделение RichMemo скрыто, пока элемент управления не в фокусе. Если False, выделение отображается постоянно.

CanRedo

```
property CanRedo: Boolean
```

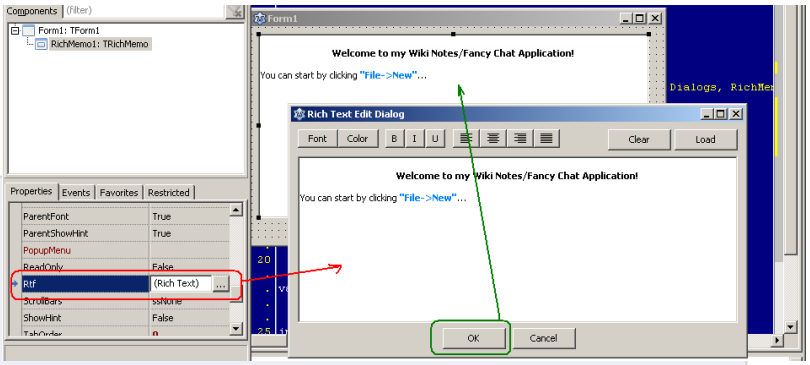
Если [возвращаемое значение] True, в очереди отмены есть действия, которые можно повторить. Если [возвращаемое] значение False, в очереди отмены нет действий, которые можно повторить. Вызов Redo не будет иметь никакого эффекта.

RTF

```
property Rtf: String
```

Read/Write свойство, которое позволяет читать или записывать текст в формате RTF для всего элемента управления. Назначение свойства - иметь возможность устанавливать Rich-Text во время разработки, однако свойство также будет работать во время выполнения.

Если значение свойства является пустой строкой, то свойство Lines используется как простой текст для инициализации значения элемента управления во время загрузки.



Примечание: Свойство доступно только в классе TRichMemo, оно недоступно (или вообще не реализовано) в классе TCustomRichMemo. Рекомендуется использовать LoadRichText/SaveRichText для изменения содержимого страницы только потому, что они могут занимать меньше памяти, чем копирование всего расширенного текста в память.

Примечание: для кроссплатформенной и прямой совместимости - в настоящее время свойство реализовано на основе набора виджетов, обеспечивающего чтение/запись в формате Rich-text. Весьма вероятно, что в ближайшем будущем оно будет изменено на использование кода чтения/записи RichMemo только RTF-stream. Проблема в том, что старые системы могут не поддерживать последнюю версию RTF.

Так что прямо сейчас вы можете столкнуться с проблемой, когда вы создадите проект в Gtk2 и сохраните richmemo время разработки, содержащее символы Unicode. Затем, если вы попытаетесь загрузить его на компьютер с XP и использовать собственный загрузчик widgetset, вы можете увидеть символы, отсутствующие или неправильно отображенные. Переход на использование [механизмов] RichMemo загрузки/сохранения RTF предотвратит эту проблему. В настоящее время вы также можете избежать этого, зарегистрировав загрузчики RichMemo (вызовите процедуры RegisterRTFLoader, RegisterRTFSaver при инициализации проекта перед загрузкой любого RichMemo)

События

OnSelectionChange

```
property OnSelectionChange: TNotifyEvent
TNotifyEvent = procedure (Sender: TObject) of object;
```

Событие запускается всякий раз, когда в RichMemo изменяется выделение: либо программно, либо из-за действий пользователя.

TRichEditForMemo

Помощник класса, который реализует программный интерфейс RichEdit. Помощник объявлен в модуле RichEditHelpers, поэтому вы должны добавить его в раздел использования. Помощники доступны в FPC 2.6.0 или старше.

Методы

FindText

Поиск заданного диапазона в тексте для целевой строки

Свойства

SelAttributes

Читает/Изменяет атрибуты символа текущего выделения

Paragraph

Читает/Изменяет атрибуты абзаца текущего выделения.

RichMemoUtils

Представлен модуль, добавляющий некоторые полезные функции, специфичные для ОС, для работы с RichMemo.

InsertImageFromFile

```
function InsertImageFromFile (const ARichMemo: TCustomRichMemo; APos: Integer;
    const FileNameUTF8: string;
    const AImgSize: TSize
): Boolean = nil;
```

Отказ от ответственности: функция будет вставлять файл изображения в RichMemo (если он реализован набором виджетов), но очень неэффективным способом. Изображение будет прочитано снова, и память будет перераспределяться для изображения каждый раз. Поэтому, пожалуйста, не используйте его для смайликов в чате. Наилучшим [выбором будет] API (с кэшированием данных). (Вот почему этот метод не является частью класса TCustomRichMemo).

- `APos` - позиция в тексте
- `AImgSize` - размер, который нужно выставлять в **ТОЧКАХ**, а не в пикселях! Если и `cx`, и `cy` равны 0, размер изображения не будет изменен вообще. Если только один [из размеров] - `cx, cy` - равен нулю, результаты [будут] неопределенными.

Есть также служебная функция **InsertImageFromFileNoResize**, которая вызывает InsertImageFromFile, передавая размер как 0.

Пример использования

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 // вызов диалога для выбора картинки для вставки
 if OpenDialog1.Execute then
   // вставка картинки в текущую позицию курсора RichMemo
   InsertImageFromFileNoResize(RichMemo1, RichMemo1.SelStart, OpenDialog1.FileName);
end;
```

InsertStyledText, InsertColorStyledText, InsertFontText

Набор функций, которые упрощают добавление и замену стилизованного текста. Указанный стиль текста применяется к вставке. Общее правило для всех функций, если `InsPos` (позиция вставки) отрицательна - текст добавляется в конец. Для вставки текста в начало теста `InsPos` должна быть установлена в 0.

Вставленный текст не обязательно должен находиться на новой строке или добавлять новую строку. Возможно, вы захотите добавить символ `LineFeed` в параметр `TextUTF8`, чтобы вставленный текст был новой строкой.

```
procedure InsertStyledText(
  const ARichMemo: TCustomRichMemo;
  const TextUTF8: String;
  AStyle: TFontStyles;
  InsPos : Integer = -1 )
```

`InsertStyledText` вставляет стилизованный текст в указанную позицию.

```
procedure InsertColorStyledText(
  const ARichMemo: TCustomRichMemo;
  const TextUTF8: String;
  AColor: TColor;
  AStyle: TFontStyles;
  InsPos : Integer = -1 )
```

`InsertColorStyledText` вставляет текст с указанным стилем и цветом в указанной позиции (по умолчанию в конец RichMemo).

```
procedure InsertFontText(
  const ARichMemo: TCustomRichMemo;
  const TextUTF8: String;
  const prms: TFontParams;
  InsPos : Integer = -1 )
```

`InsertFontText` вставляет текст и применяет к нему указанные FontParams.

Возможно, вы захотите создать помощник класса для реализации этих функций в качестве методов для RichMemo. Осторожно, если вы используете Delphi-совместимый помощник - вы можете [попасть] в конфликт.

Установка

- Получить последнюю версию из SVN
- Откройте пакет и установите его, пересоберите IDE
- TRichMemo будет добавлен на вкладку 'Common Controls'

Часто задаваемые вопросы

Использование RichMemo в общих библиотеках

Issue #17412 Если вам нужно использовать компонент в разделяемой библиотеке, вам может понадобиться добавить ключ -fPIC к опции компилятора «пакета» и «проекта».

(Delphi) RichEdit-подобный интерфейс

Issue #14632 Типичная проблема - портирование существующего кода, использующего RichEdit из Delphi. Интерфейс RichMemo не совпадает с RichEdit во многих отношениях. Но есть два способа справиться с этим:

- вы можете создать подкласс из TCustomRichMemo (или RichMemo) и реализовать методы Delphi RichEdit;
- вы можете использовать модуль RichMemoHelpers (требуется fpc 2.6.0 или более поздняя версия) и использовать методы, предоставляемые классом Helpers; в настоящее время реализованы свойства SelAttributes и Paragraph.

```
uses ... RichMemo, RichMemoHelpers;

TForm = class
  RichMemo1 : TRichMemo;

  // Свойство SelAttributes недоступно в базовом классе,
  // но добавлено помощником, определенным в модуле RichMemoHelpers
RichMemo1.SelAttributes.Name := 'Courier New';
```

Добавление смешанного цветного текста в конец RichMemo

Если вам просто нужна простая раскраска, то вот пример, который будет каждый раз добавлять новую строку со случайным цветом (протестировано в Windows):

```
  procedure TForm1.Button1Click(Sender: TObject);
  var
    i, j: integer;
    str: string;
  begin
    with Richmemo1 do
    begin
      str := 'Newline text';
      i := Length(Lines.Text) - Lines.Count; // CR(перевод каретки) как #10#13 считается
только один раз, так что вычтите его один раз
      j := Length(str) + 1;                  // +1 сделать CR в том же формате
      Lines.Add(str);
      SetRangeColor(i, j, Round(random * $FFFFFF));
    end;
  end;
```

```
Newline text
NewLine text
NewLine text
```

Альтернативный пример простой раскраски (протестировано на Windows):

```
  procedure TForm1.Button2Click(Sender: TObject);
  begin
    with RichMemo1 do
    begin
      Lines.Add('Line in red');
      SetRangeColor(Length(Lines.Text) - Length(Lines[Lines.Count - 1]) - Lines.Count -
1, Length(Lines[Lines.Count - 1]), clRed);

      Lines.Add('Line in blue');
      SetRangeColor(Length(Lines.Text) - Length(Lines[Lines.Count - 1]) - Lines.Count -
1, Length(Lines[Lines.Count - 1]), clBlue);

      Lines.Add('Normal line.');
      Lines.Add('Normal line.');

      Lines.Add('Line in green ');
      SetRangeColor(Length(Lines.Text) - Length(Lines[Lines.Count - 1]) - Lines.Count -
1, Length(Lines[Lines.Count - 1]), clGreen);
```

```
    end;
  end;
```

```
Line in red
Line in blue
Normal line.
Normal line.
Line in green
```

Если вам нужна смешанная раскраска, то это пример, который добавит новую строку с несколькими разноцветными словами (протестировано в Windows):

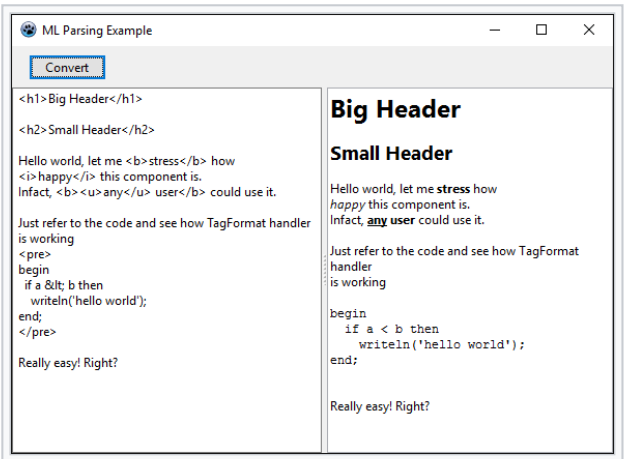
```
procedure TForm1.Button3Click(Sender: TObject);
    procedure AddColorStr(s: string; const col: TColor = clBlack; const NewLine: boolean
= true);
    begin
      with RichMemo1 do
      begin
        if NewLine then
        begin
          Lines.Add('');
          Lines.Delete(Lines.Count - 1); // избегаем двойного межстрочного интервала
        end;

        SelStart  := Length(Text);
        SelText   := s;
        SelLength := Length(s);
        SetRangeColor(SelStart, SelLength, col);

        // отменяем выделение вставленной строки и помещаем курсор в конец текста
        SelStart  := Length(Text);
        SelText   := '';
      end;
    end;
  begin
    AddColorStr('Black, ');
    AddColorStr('Green, ', clGReen, false);
    AddColorStr('Blue, ', clBlue, false);
    AddColorStr('Red', clRed, false);
  end;
```

```
Black, Green, Blue, Red
Black, Green, Blue, Red
Black, Green, Blue, Red
```

Разбор языка разметки

Нативные элементы управления rich-edit не поддерживают синтаксический анализ языка разметки.

Вам нужно написать эту [функциональность] самостоятельно, как показано в примере с mlparse.



mlparse example

Что внутри

Матрица поддержки платформы

Ожидается, что значения в матрице не будут статическими (за исключением столбца "carbon"), каждая система должна поддерживать все функции, объявленные RichMemo.

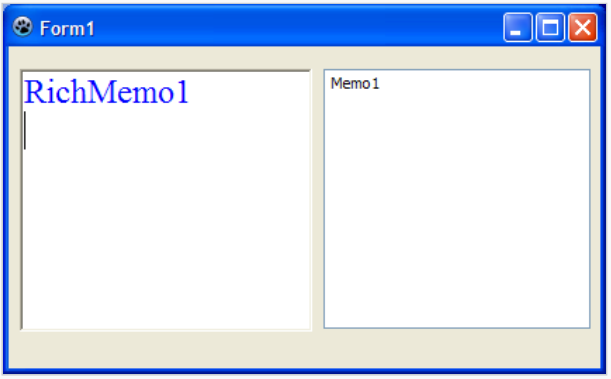
Обратите внимание, что таблица содержит больше функций, чем опубликовано выше. Это означает, что API-интерфейсы

уже доступны в RichMemo, но, поскольку они нестабильны (на некоторых платформах), они считаются «экспериментальными», а не обещанными.

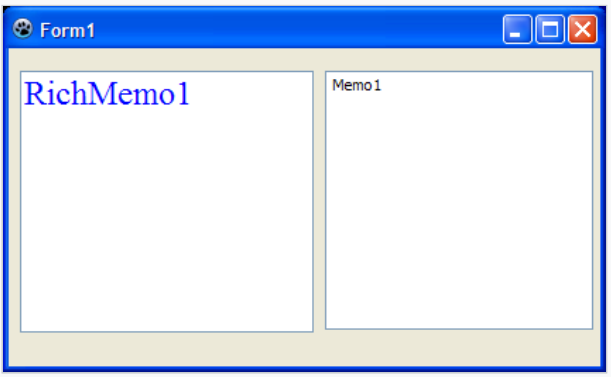
Особенность	Win32	Gtk2	Qt	Cocoa	Carbon
Выбор цвета и стиля шрифта	Да	Да	Да	Да	Да
Цвет фона шрифта	Да	Да	Да	Да	Нет
Подстрочный, надстрочный [текст]	Да	Да	Нет	Нет	Нет
[функция] GetStyleRange	Да	Да	Нет	Да	Да
Выравнивание абзаца	Да	Да	Да	Да	Почти невозможно
Метрика абзаца	Да	Да	Нет	Да	Почти невозможно
Отступы абзаца	Да	Да	Нет	Да	Нет
Масштабирование	Да	не завершено	Нет	Да	Нет
Печать	Да	Нет	Нет	Нет	Нет
Загрузка/сохранение RTF	OS	RichMemo	Нет	OS	OS
Возможность вставки	Да	Да	Нет	Нет	Нет

Win32

- RichEdit используется в качестве системного виджета. Последний известный .dll загружается и инициализируется при запуске. Обратите внимание, что RichEdit 1.0 не поддерживает большинство функций.
- Внутренняя оболочка RichEditManager предоставляется для совместимости с Win 9x. Однако это никогда не испытывалось и не проверялось. Также ожидается, что объект TOM может быть упакован как одна из реализаций RichEditManager. Однако имеющийся на месте «менеджер» может быть полностью удален.
- Согласно Интернету, элемент управления RichEdit не был обновлен Microsoft для поддержки рисования тем (начиная с XP и до Windows 8.1). Таким образом, RichEdit всегда может выглядеть как старый Win9x в 3D-элементе управления. На основе патча, предоставленного в опубликованной проблеме способ переопределить WM_NCPAINT был представлен в r4153.



RichMemo without Theme patch



RichMemo with Theme

Win32RichMemo предоставляет глобальную переменную NCPaint. Это обработчик рисования не-клиентской области. По умолчанию он пытается нарисовать тематическую границу (см. Win32RichMemo.ThemedNCPaint для реализации). Он обеспечивает хорошие результаты по темам Windows XP, но в более поздних темах (где используются анимация) результаты не так хороши и должны быть обновлены.

Чтобы система могла выполнять только NCPaint (т.е. реализация LCL вызывает проблемы или новые окна обновляли RichEdit для правильного рисования границ), вы можете изменить значение NCPaint во время выполнения, сбросив его в nil:

```
uses
  RichMemo, ..
  {$ifdef WINDOWS}
  Win32RichMemo
  {$endif}
```

```
initialization
  {$ifdef WINDOWS}
  Win32RichMemo.NCPaint:=nil;
  {$endif}
```

Вы также можете предоставить собственную реализацию NCPaint. Однако, если вы реализуете правильную анимацию рисунка темы, пожалуйста, предоставьте патч.

Поведение зависит от реализации Windows и не должно (и не будет) частью интерфейса RichMemo.

Gtk2

- В качестве системного виджета используется GtkTextView .
- Subscript и Superscript изначально не поддерживаются, в gtk2richmemo реализован дополнительный код для их реализации.
- Маркированные и нумерованные списки эмулируются.

Gtk3

API между Gtk2 и Gtk3 на самом деле не отличаются. Основное отличие в отрисовке. Для Gtk3 активно используется Cairo Canvas. Единственная область, на которую это влияет - "внутренности". Если кто-то хочет внести свой вклад в реализацию Gtk2 - пожалуйста, создайте отдельные модули gtk3richmemoXXX. Не нужно создавать {$ ifdefs} в модулях gtk2.

Cocoa

- Для Cocoa в качестве системного виджета используется NSTextView .
- Не каждое семейство шрифтов предоставляет «курсивный» шрифт. Таким образом, даже если вы передадите fsItalic как часть Style для шрифта, он может быть проигнорирован.

Qt

- Используется QTextEdit виджет. В современных C-отображениях для Qt отсутствует множество API-интерфейсов RichText, поэтому полная реализация в настоящее время невозможна.
- Следующие классы должны быть сопоставлены с C-функциями:
  - QTextBlockFormatH необходим для выравнивания абзацев (например, отступы абзаца, межстрочный интервал и табуляции). QTextEdit предоставляет только выравнивание абзаца.
  - QTextCharFormatH необходим для дополнительного форматирования символов (то есть вертикальное выравнивание, поддержка ссылок). QTextEdit предоставляет только стили шрифтов.

См. также

- RichMemo/Features - комментирует некоторые функции в процессе.
- RichMemo/WorkArounds - некоторые заметки о функциях еще не завершены.
- RichMemo/Defines - дополнительные переключатели пакетов, которые позволяют решить проблемы компиляции/времени выполнения
- RichMemo/Samples - еще примеры
- MyNotex приложение, которое использует модифицированный пакет Massimo Nardello RichMemo для Linux-Gtk2
- TMemo
- KMemo control

Categories: Pages using deprecated source tags | Russian | Lazarus-CCR/ru | Components/ru | Packages/ru | RichMemo/ru