Використання компонента Memo

03.07.2017МасивиКоментарі: 0

У деяких випадках для введення масиву можна використовувати компонент **Memo.** Компонент **Memo** дозволяє вводити текст, що складається з досить великої кількості рядків, тому його зручно використовувати для введення символьного масиву. Компонент **Memo** додається в форму звичайним чином. Значок компонента знаходиться на вкладці **Standard** (рис. 5.4).

Рис. 5.4. Компонент Memo

У табл. 5.3 перераховані деякі властивості компоненту **Memo**.

Таблиця 5.3. Властивості компонента Memo

Властивість	**Визначає**
Name	Ім’я компонента. використовується в програмі для доступу до властивостей компонента
Text	Текст, що знаходиться в полі Memo. Розглядається як єдине ціле
Lines	Текст, що знаходиться в полі Memo. Розглядається як сукупність рядків. Доступ до рядка здійснюється за номером
Lines .Count	Кількість рядків тексту в поле Memo
Left	Відстань від лівої межі поля до лівої межі форми
Top	Відстань від верхньої межі поля до верхньої межі форми
Height	Висоту поля
Width	Ширину поля
Font	Шрифт, використовуваний для відображення тексту, що вводиться
ParentFont	Ознака успадкування властивостей шрифту батьківської форми

ри використанні компонента **Memo** для введення масиву значення кожного елемента масиву слід вводити в окремій рядку і після введення кожного елемента масиву натискати клавішу **“Enter “** .

Отримати доступ до об'єктів в поле **Memo** рядку тексту можна за допомогою властивості **Lines**, вказавши в квадратних дужках номер потрібного рядка (рядка нумеруються з нуля).
Наступна програма, текст якої наведений у лістингу 5.5, демонструє використання компонента **Memo** для введення символьного масиву.
Основний цикл процедури введення символьного масиву з компоненту **Memo** може виглядати так:

```
for i: = l to SIZE do

a [I]: = Memol.Lines [i];
```

де:

- SIZE – іменована константа, визначає розмір масиву;
- a – масив;
- Memol – ім'я Memo-компонента;
- Lines – властивість компонента Memo, що представляє собою масив, кожен елемент якого містить один рядок що знаходиться в полі Memo тексту.

Форма програми приведена на рис. 5.5. Крім поля **Memo** вона містить командну кнопку (Buttonl), при натисканні на якій виконується введення значень елементів масиву з поля **Memo.**

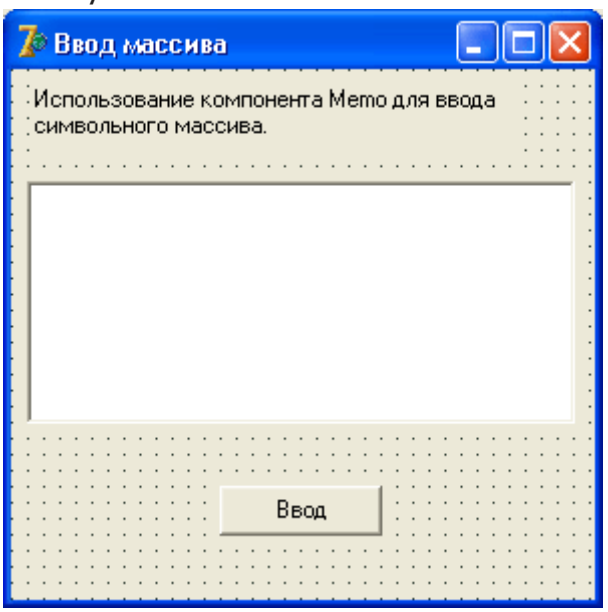

Рис. 5.5. Діалогове вікно програми Введення масиву

Лістинг 5.5. введення масиву рядків з компонента Memo

```
unit fr_memo_; interface

uses

Windows, Messages, SysUtils, Classes,



Graphics, Controls, Forms, Dialogs, Menus, StdCtrls;

type

TForm1 = Class (TForm)
```

```
Memo1: TMemo;

Button1: TButton;

Label1: TLabel;

procedure ButtonlClick (Sender: TObject);


private

{ Private declarations}


public

{Public declarations}


end;

var

Forml: TForm1;

implementation

($ R * .DFM}

procedure TForml .ButtonlClick (Sender: TObject);


const

SIZE = 5; // розмір масиву var

a: array [l..SIZE] of string [30]; // масив


n: integer; // кількість рядків, введених в поле Memo


i: integer; // індекс елемента масиву


st : string;


begin

n: = Memo1.Lines.Count;
```

```
if n = 0 then begin

ShowMessage ( 'Вихідні дані не введені! ');


Exit; // вихід з процедури обробки події


end;

// в поле Memo є текст


if n > SIZE then begin

ShowMessage ( 'Кількість рядків перевищує розмір масиву. ');


n: = SIZE; // будемо вводити тільки перші SIZE рядків


end;


for i: = 1 to n do

a [i]: = Form1.Memo1.Lines [i-1]; // рядка Memo пронумеровані з нуля

// вивід масиву в вікно повідомлення


if n > 0 then begin

st: = 'Введений масив: '+ # 13;


for i: = 1 to n do

st: = st + IntToStr (i) + ' '+ A [i] + f13; ShowMessage (st);


end;

end;


end.
```

Основну роботу виконує процедура **TForm1.Button1Click,** яка спочатку перевіряє, чи є в полі **Memo1** текст. Якщо текст є (в цьому випадку значення

властивості **Lines.Count** більше нуля), то процедура порівнює кількість введених рядків і розмір масиву. Якщо це кількість перевищує розмір масиву, то програма змінює значення n, тим самим готує введення тільки перших **SIZE** рядків.
На рис. 5.6 наведено вид діалогового вікна програми **Введення масиву**. Після клацання на командній кнопці Введення з'являється вікно (рис. 5.7), яке містить значення елементів масиву, отримані з **Memo-поля.**

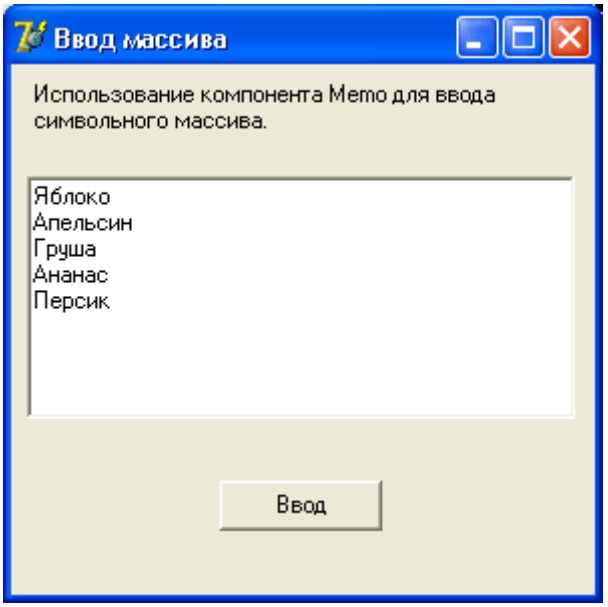

Рис. 5.6. Вікно програми Введення масиву

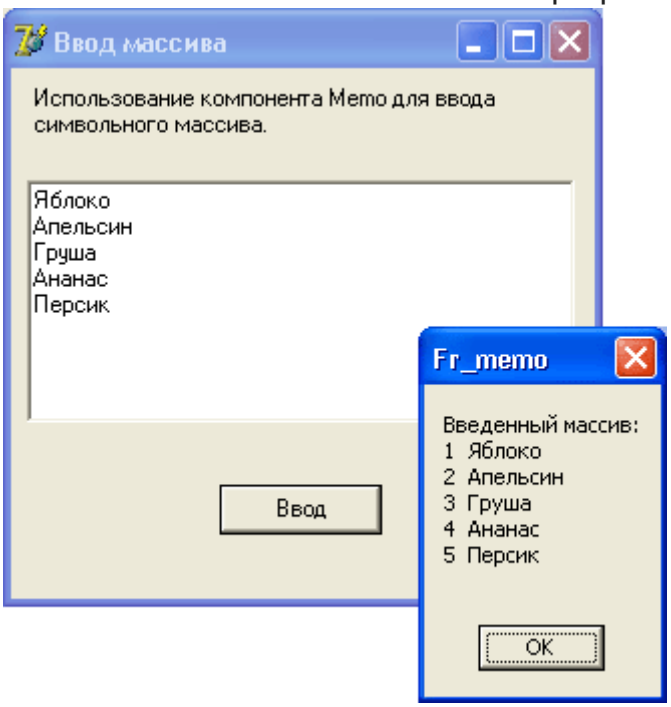

Рис. 5.7. Масив, введений з Memo-поля

Memo (панель инструментов **Standard**) – многострочный текстовый редактор в Дельфи. Позволяет редактировать текст окна (сходство с окном **Edit**). Различие между **Edit** и **Memo** это множество строк. В свойстве **Font** компонента **Memo** формат всего текста одинаков. Для различных свойств формата текста лучше использовать компонент **RichEdit**.

Если Вы решили сохранять текст, введенный в Memo, то созданный текстовый файл не будет содержать элементов форматирования текста. По сути в сохраненном файле будет лишь сам текст, а не его атрибуты (пр. шрифт, курсив и т. д.). К примеру, если Вы выгружаете из файла текст обратно в **Memo**, то придется задавать **Font** программно.

Рассмотрим следующие свойства, свойство **Lines** имеет много свойств и методов типа **Tstrings**. Обычно используются для форматирования и редактирования текста. Сам текст хранится в свойстве **Text**. Свойства **SelStart, SelLength, SelStart, Modified** – описываются в предыдущей статье (читайте «*Однострочное окно редактирования – Edit*»).
Свойство Alignment определяет выравнивание текста (влево, вправо, по центру). ***Пример: Memo1.Alignment:=taCenter;***.
WordWrap – свойство переноса длинных строк. Значения **true** или **false**. Полосы прокрутки задаются свойством **ScrollBars**. Адаптацию размера окна **Memo** к размеру формы или приложения задаются свойствами **Align** и **Anchors**.

К основным свойствам (**Properties**) относятся:

свойство	описание
Align	Способ выравнивания в родительском окне
Alignment	Выравнивание текста (taLeftJustify, taCenter, taRightJustify)
WantReturns	**property WantReturns:Boolean;** Можно ли выставить в текст символы возврата корретки
WantTabs	**property WantTabs:Boolean;** Можно ли выставить в текст символы каретки

WantWrap	**property WantWrap:Boolean;** Указывает переносится ли текст на новую строчку
Font	**property Font: TFont;** Атрибуты шрифта
PopupMenu	**property PopupMenu: TPopupMenu;** Всплывающее меню компонента
ReadOnly	**property ReadOnly:Boolean;** Может ли пользователь изменять текст в компоненте.
SelLength	**property SelLength:Integer;** Определяет количество выделенных символов в строке.
MaxLength	**property MaxLength:Integer;** Указывает максимальное количество символов, по умолчанию 0 – не ограничено
Modified	**property Modified:Boolean;** Указывает, редактировался ли текст в компоненте.

Рассмотрим основные методы. **Clear** – удаляет текст окна. **ClearSelection** – удаляет текст, выделенный в окне. **ClearUndo** – очищаем буфер обмена от команд. **CopyToClipboard** – перенос выделенного текста в **Clipboard**. **CutToClipboard** – перенос выделенного текста в **Clipboard** и уничтожение текста в окне. **PasteToClipboard** – вставка текста в окно из буфера обмена. **Selectall** – выделяем весь текст в окне. **Undo** – отменяем все изменения, которые хранятся в буфере с момента последнего вызова **ClearUndo**. Все эти методы являются процедурами, и задавать их нужно как **Procedure <имя процедуры>;**

Основные события (**Events**) для **Memo**:

OnChange – наступает, когда текст в окне изменился, **OnKeyDown** – наступает при нажатии пользователем любой клавиши (можно распознать нажатую клавишу в обработчике). **OnKeyPress** – событие наступает при нажатии пользователем клавиши символа (можно распознать вводимый символ в обработчике, а также запретить ввод). **OnKeyUp –** наступает, когда пользователь отпустил какую-либо клавишу (также можно распознать клавишу).

Компонент Memo в Delphi

https://delphi-prg.ru/komponent-memo-v-delphi

Компонент **Memo** находиться на странице "Standard" Палитры компонентов. Его основное предназначение - работа с большим количеством строк (ввод, отображение и редактирование текстового материала). Эти строки содержатся в свойстве **Lines** компонента Memo.

Как и во многих других текстовых редакторах у компонента **Memo** есть возможность использовать общепринятые горячие клавиши, такие как: Ctrl-X — выделенный текст вырезается и помещается в буфер обмена, Ctrl-C — копируем выделенный текст в буфер обмена, Ctrl-V — вставляем текстовое содержимое из буфера обмена в место нахождения курсора, для отмены последней команды используем Ctrl-Z.

Заполняя поле Memo, как и любого другого текстового редактора, необходимо переходить на новую строчку. Для того чтобы мы могли использовать клавишу ENTER свойству **WantReturns** должно быть присвоено значение TRUE, иначе на новую строку можно переходить только сочетанием клавиш CTRL + ENTER.

Записывая длинную текст, чтобы он автоматически продолжался с новой строки указываем свойству **WordWrap** значение TRUE. При включенной горизонтальной полосе прокрутке это свойство игнорируется.

Если у нас имеется большой текст и он не помещается в компоненте Memo, то для удобства просмотра устанавливаем полосу прокрутки свойством **ScrollBar**. Оно принимает следующие значения:

- ssNone - Нет полосы прокрутки;
- ssHorizontal - Установлена горизонтальная прокрутка;
- ssVertical - Установлена вертикальная прокрутка;
- ssBoth - Установлень две полосы прокрутки;

Свойство **ReadOnly** разрешает редактирование текста (програмно все равно текст можно добавлять).

Свойство **MaxLength** задает количество символов, которое можно ввести. Значение равное нулю не ограничивает ввода.

Для работы с выделенным текстом используются функции: **SelStat** - позиция первого выделенного символа, **SelLength** - число сколько выделено символов, **SelectAll** - выделение всего текста, **ClearSelection** - очистить выделенный текст. Для работы выделенного текста с буфером обмена используются методы:

- CutToClipboard - вырезать выделенный текст;
- CopyToClipboard - скопировать выделенный текст;
- PasteFromClipboard - вставить выделенный текст;

Для сохранения содержимого текстового поля Memo в файл используется функция **SaveToFile('mytetxt.txt')**, а для извлечения - **LoadFromFile('mytetxt.txt')**, где mytetxt.txt - текстовый файл расположенный в каталоге программы.

Для закрепления материала выполним небольшое практическое задание, создадим простой текстовый редактор.

Расположим на форме компонент Memo, установим у него вертикальную прокрутку. Справа будут кнопки компонент Button 8 штук, согласно рисунка сверху.
Теперь запишем обработчики событий для наших кнопок.

Кнопка "Открыть":
```
Memo1.Lines.LoadFromFile('mytetxt.txt');
```

Кнопка "Сохранить:
```
Memo1.Lines.SaveToFile('mytetxt.txt');
```

Кнопка "Копировать"
```
Memo1.CopyToClipboard;
```

Кнопка "Вырезать"
```
Memo1.CutToClipboard;
```

Кнопка "Очистить все"
```
Memo1.Clear;
```

Кнопка "Вставить"
```
Memo1.PasteFromClipboard;
```

Кнопка "Выделить все"
```
Memo1.SetFocus; // если компонент не в фокусе, то выделение  не увидим
Memo1.SelectAll;
```

Кнопка "Удалить выделенное"
```
Memo1.ClearSelection;
```