Ввод – вывод данных

Устройства вывода

К основным устройствам вывода, прежде всего, можно отнести мониторы и принтеры.

Монитор — устройство визуального отображения информации (в виде текста, таблиц, рисунков чертежей и др.). В основном используют мониторы на базе электронно-лучевых трубок и на базе жидкокристаллических матриц. Существуют опытные образцы мониторов на базе полимеров, светодиодов, органических светодиодов, флуоресцентной технологии, электрохимических красителей, голографических проекторов и прочие модели.

Принтер — печатающее устройство. Осуществляет вывод из компьютера закодированной информации в виде печатных копий текста или графики (лазерные; струйные, с термопереносом, матричные, трехмерные)

Устройства ввода

Клавиатура компьютера — устройство для ввода информации в компьютер и подачи управляющих сигналов.

Манипуляторы (мышь, джойстик и др.) — это специальные устройства, которые используются для управления курсором. К манипуляторам можно отнести мышь, джойстик, трекбол.

Джойстик — обычно это стержень-ручка, отклонение которой от вертикального положения приводит к передвижению курсора в соответствующем направлении по экрану монитора.

Трекбол — небольшая коробка с шариком, встроенным в верхнюю часть корпуса.

Мышью (Mouse) исторически называют координатно-позиционирующее устройство, преобразующее перемещение на плоскости в команды позиционирования указателя на экране монитора.

Сканер — устройство для ввода в компьютер графических изображений. Создает оцифрованное изображение документа и помещает его в память компьютера.

Компоненты вывода данных на экран

- **Label** (метка) - отображение текста, который не изменяется пользователем. Никакого оформления текста не предусмотрено, кроме цвета метки и текста. Основное свойство — **Caption**.
- **StaticText** (метка с бордюром) - подобен компоненту **Label**, но обеспечивает возможность задания стиля бордюра. Основное свойство — **Caption**.
- **Panel** (панель) - компонент является контейнером для группирования органов управления, но может использоваться и для отображения текста с возможностями объемного оформления. Основное свойство — **Caption**.
- **Edit** (окно редактирования) - отображение, ввод и редактирование однострочных текстов. Имеется возможность оформления объемного бордюра. Основное свойство — **Text**.
- **MaskEdit** (окно маскированного редактирования) - используется для форматирования данных или для ввода символов в соответствии с шаблоном. Основные свойства — **Text** и **EditMask**.
- **Memo** (многострочное окно редактирования). Отображение, ввод и редактирование многострочных текстов. Имеется возможность оформления объемного бордюра. Основное свойство — **Lines**.

Компоненты вывода данных на экран

- **RichEdit** (многострочное окно редактирования в формате RTF) - компонент представляет собой окно редактирования в стиле Windows в обогащенном формате RTF, позволяющее производить выбор атрибутов шрифта, поиск текста и многое другое. Основное свойство — **Lines**.
- **ListBox** (окно списка) - отображение стандартного окна списка Windows, позволяющего пользователю выбирать из него пункты. Основное свойство — **Items**.
- **CheckListBox** (список с индикаторами) - компонент является комбинацией свойств списка **ListBox** и индикаторов **CheckBox** в одном компоненте.
- **ComboBox** (редактируемый список) - объединяет функции **ListBox** и **Edit**. Пользователь может либо ввести текст, либо выбрать его из списка. Основное свойство — **Items**.
- **StringGrid** (таблица строк) - Отображение текстовой информации в таблице из строк и столбцов с возможностью перемещаться по строкам и столбцам и осуществлять выбор. Основное свойство — **Cells**.

Компонент Label

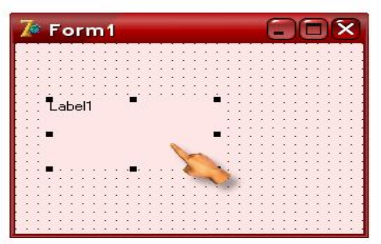
Тексты, отображаемые в компоненте, определяются значением их свойства `Caption`:

```
Label1.Caption := 'Новый текст';
```

Цвет фона определяется свойством `Color`, а цвет надписи — подсвойством `Color` свойства `Font`.

С точки зрения оформления выводимого текста `Label` дает минимальные возможности.

Компонент располагается во вкладке `Standard`.

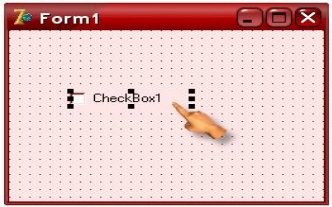


Компонент CheckBox

Индикаторы с флажком `CheckBox` используются в приложениях для того, чтобы пользователь мог включать и выключать опции, или для индикации состояния.

При каждом щелчке пользователя на индикаторе его состояние изменяется, проходя через значения: выделенное (свойство `cbChecked`) и не выделенное (свойство `cbUnchecked`).

Проверять состояние индикатора можно по значению свойства `Checked`. Если `Checked` равно `true`, то индикатор выбран, т.е.:



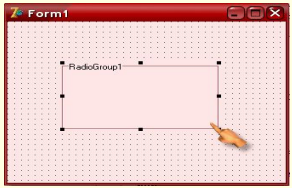
Компонент RadioGroup

`RadioGroup` — панель группы радиокнопок. Это панель, которая может содержать регулярно расположенные столбцами и строками радиокнопки. Надпись в левом верхнем углу панели определяется свойством `Caption`. А надписи кнопок и их количество определяются свойством `Items`, имеющим тип `TStrings`.

Определить, какую из кнопок выбрал пользователь, можно по свойству `ItemIndex`, которое показывает индекс выбранной кнопки. По умолчанию `ItemIndex` = -1, что означает отсутствие выбранной кнопки.

`RadioGroup` хорошо использовать, если надписи кнопок имеют примерно одинаковую длину и если число кнопок в каждом столбце одинаково.

`RadioGroup` при размещении кнопок ориентируется на надпись максимальной длины.



Компоненты Edit и LabeledEdit

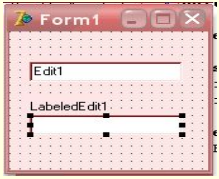
В компонентах `Edit` и `LabeledEdit` вводимый и выводимый текст содержится в свойстве `Text`. Это свойство можно устанавливать в процессе проектирования или задавать программно. Выравнивание текста, как это имело место в метках и панелях, невозможно. Перенос строк тоже невозможен. Окна редактирования снабжены многими функциями, свойственными большинству редакторов. Например, в них предусмотрены типичные комбинации «горячих» клавиш.

Свойство `AutoSelect` определяет, будет ли автоматически выделяться весь текст при передаче фокуса в окно редактирования.

Свойство `Modified`, доступное только во время выполнения, показывает, проводилось ли редактирование текста в окне.

Компонент `Edit` находится во вкладке `Standard`:

Компонент `LabeledEdit` – во вкладке `Additional`:

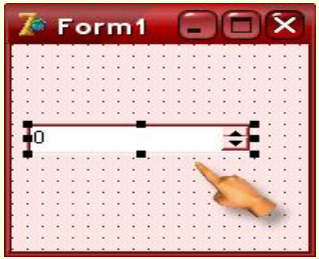


Компонент SpinEdit

Компонент `SpinEdit` представляет отдельный тип компонента, обеспечивающего ввод целых чисел.

Свойства компонента `SpinEdit` имеют следующие имена: `MinValue`, `MaxValue`, `Value`.

Компонент `SpinEdit` находится во вкладке `Samples`:



Компоненты ComboBox и ListBox

Компоненты отображают списки строк. Они отличаются друг от друга тем, что **ListBox** только отображает данные и позволяет пользователю выбрать из них то, что ему надо, а **ComboBox** позволяет также редактировать данные. **ListBox** отображает список в раскрытом виде и автоматически добавляет в список полосы прокрутки. **ComboBox** позволяет отображать список как в развернутом виде, так и в виде выпадающего списка.

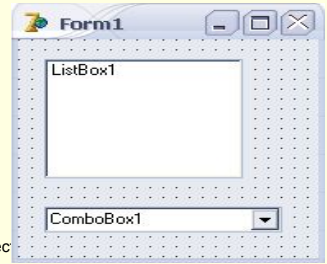
Основное свойство обоих компонентов, содержащее список строк, — **Items**, имеющее тип **TStrings**.

В компоненте **ListBox** имеются свойства **MultiSelect**, **Columns**, **Sorted**.

Стиль изображения компонента **ComboBox** определяется свойством **Style.**

Компонент **ComboBox** обладает следующими свойствами: **Text**, **ItemIndex**, **MaxLength**, **Sorted**.

Вкладка Standard:



Многострочное окно редактирования Memo

Компонент **Memo** является окном редактирования многострочного текста.

В компоненте **Memo** формат одинаков для всего текста и определяется свойством **Font**.

Основные свойства окна редактирования: **Alignment, WordWrap, ScrollBars, Lines**. Свойство только для чтения **Count** указывает число строк в тексте.

Для очистки текста в окне надо выполнить процедуру **Clear**. Для занесения новой строки в конец текста окна редактирования можно воспользоваться методами **Add** или **Append** свойства **Lines**.

Для загрузки текста из файла применяется метод **LoadFromFile**. Сохранение текста в фале осуществляется методом **SaveToFile**.

Таблица строк — компонент StringGrid

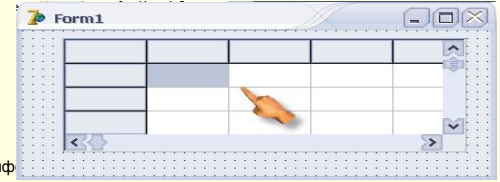
Компонент **StringGrid** представляет собой таблицу, содержащую строки. Данные таблицы могут быть только для чтения или редактируемыми. Таблица может иметь полосы прокрутки, причем заданное число первых строк и столбцов может быть фиксированным и не прокручиваться.

Основные свойства компонента:**Cells, Cols, Rows, Objects.** Свойства **ColCount** и **RowCount** определяют соответственно число столбцов и строк, свойства **FixedCols** и **FixedRows** — число фиксированных, не прокручиваемых столбцов и строк. Цвет фона фиксированных ячеек определяется свойством **FixedColor**.

Свойство **ScrollBars** определяет наличие в таблице полос прокрутки.

Среди множества событий компонента **StringGrid** следует отметить событие **OnSelectCell**, возникающее в момент выбора пользователем ячейки.

Компонент расположен во вкладке Additional:



Вывод в текстовый файл

Объявление файла

Файл — это именованная структура данных, представляющая собой последовательность элементов данных одного типа, причем количество элементов последовательности практически не ограничено. *Файл*, компонентами которого являются данные символьного типа, называется ***символьным,*** или ***текстовым***.

Описание текстового файла в общем виде выглядит так:

Имя: TextFile;

где: *имя* — имя файловой переменной; *TextFile* — обозначение типа, показывающее, что *Имя* — это файловая переменная, представляющая текстовый файл.

Вывод в текстовый файл

Указание имени файла

Для того, чтобы программа могла выводить данные в файл или считывать данные из файла, необходимо указать конкретный файл, т. е. связать файловую переменную с конкретным файлом (задать имя файла).

Имя файла задается вызовом процедуры **AssignFile**, связывающей файловую переменную с конкретным файлом.

Описание процедуры AssignFile выглядит следующим образом:

AssignFile(var f, ИмяФайла: string);

Примеры вызова процедуры AssignFiie:

AssignFile(f, 'a:\result.txt');
AssignFile(f, '\students\ivanov\korni.txt');
fname:=('otchet.txt'); AssignFiie(f,fname);

Вывод в текстовый файл

Открытие файла для вывода

Возможны следующие режимы открытия файла для записи в него данных:

- перезапись (запись нового файла поверх существующего или создание нового файла); чтобы открыть файл в режиме создания нового файла или замены существующего, необходимо вызвать процедуру ***Rewrite(f),*** где f — файловая переменная типа TextFile.
- добавление в существующий файл; чтобы открыть файл в режиме добавления к уже существующим данным, находящимся в этом файле, нужно вызвать процедуру ***Append (f),*** где f — файловая переменная типа TextFile.

Закрытие файла

Перед завершением работы программа должна закрыть все открытые файлы. Это делается вызовом процедуры closefile. Процедура closefile имеет один параметр — имя файловой переменной.

Пример использования процедуры:

CloseFile(f)

Вывод в текстовый файл

Вывод в файл

Непосредственно вывод в текстовый файл осуществляется при помощи инструкции write или writeln:

write (ФайловаяПеременная, СписокВывода) ;

writeln(ФайловаяПеременная, СписокВывода);

где: ***ФайловаяПеременная*** — переменная, идентифицирующая файл, в который выполняется вывод; ***СписокВывода*** - разделенные запятыми имена переменных, значения которых надо вывести в файл. Помимо имен переменных в список вывода можно включать строковые константы.

Например, если переменная f является переменной типа TextFile, то инструкция вывода значений переменных x1 и x2 в файл может быть такой:

write(f, 'Корни уравнения', xl, x2);

Вывод в текстовый файл: пример

```
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile; // файл
fName: String; // имя файла
i: integer;
begin
 fName := Editl.Text;
 AssignFile(f, fName);
 Rewrite(f); // открыть для записи
 / / запись в файл
  writeln(f, ….);
 CloseFile(f); // закрыть файл
 MessageDlg('Данные записаны в файл ‘+fName,mtlnformation,[mbOk],0);
end;
```

Ввод из файла

Открытие файла для ввода (чтения) выполняется вызовом процедуры ***Reset***. С помощью функции ***AssignFile*** файловая переменная должна быть связана с конкретным файлом.

Например, следующие инструкции открывают файл для ввода:

```
AssignFile(f, 'c:\data.txt');
Reset(f);
```

Чтение из файла выполняется при помощи инструкций read и readln, которые в общем виде записываются следующим образом:

```
read(ФайловаяПеременная, СписокПеременных);
readln(ФайловаяПеременная, СписокПеременных) ;
```

Где ФайловаяПеременная — переменная типа TextFile;
СписокПеременных — имена переменных, разделенные запятыми.

Чтение чисел

В текстовом файле находятся не числа, а их изображения. Действие, выполняемое инструкциями **read** или **readln**, состоит из двух: сначала из файла читаются символы до появления разделителя, затем прочитанные символы преобразуются в число, и полученное значение присваивается переменной, имя которой указано в качестве параметра инструкции **read** или **readln**.

Пример: текстовый файл a: \data. txt содержит следующие строки: 23 15 45 28 56 71. В результате выполнения инструкций:

```
AssignFile(f, 'a:\data.txt');
Reset(f); // открыть для чтения
read(f, a); read(f, b, c); read(f, d);
```

значения переменных будут: a = 23, b = 15, c = 45, d = 28.

Отличие инструкции readln от read состоит в том, что после считывания указатель чтения из файла автоматически перемещается в начало следующей строки файла.

Ввод из файла

Конец файла

Пусть на диске есть некоторый текстовый файл. Нужно в диалоговое окно вывести содержимое этого файла. Решение задачи довольно очевидно: надо открыть файл, прочитать первую строку, затем вторую, третью и т. д. до тех пор, пока не будет достигнут конец файла. Но как определить, что прочитана последняя строка, достигнут конец файла?

Для определения конца файла можно воспользоваться функцией EOF (End of File — конец файла). У функции EOF один параметр — файловая переменная. Значение функции EOF равно False, если прочитанный элемент данных не является последним в файле, т. е. возможно дальнейшее чтение. Если прочитанный элемент данных является последним, то значение EOF равно True.

Значение функции EOF можно проверить сразу после открытия файла. Если при этом оно окажется равным True, то это значит, что файл не содержит ни одного элемента данных, т. е. является пустым (размер такого файла равен нулю).