Лабораторная работа 6

УПРАВЛЕНИЕ ОТОБРАЖЕНИЕМ ДАННЫХ В ПРИЛОЖЕНИИ

Цель работы: рассмотреть аспекты представления графических данных содержащихся в *BLOB*-полях, а также механизмы сортировки на основе *SQL*-запросов и средствам *ADO*-компонент.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Организовать отображение изображений товаров в приложении.
3. Сделать сортировку записей с привязкой к событию **OnTitleClick**.
4. Установить подсветку записей в компоненте **DBGrid**.
5. Ответить на контрольные вопросы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Тип данных BLOB применяется для хранения любых бинарных данных, но чаще всего используется для хранения изображений, документов или других мультимедийных объектов.

6. РАБОТА С ИЗОБРАЖЕНИЯМИ В ПОЛЯХ BLOB

Рассмотрим в качестве примера хранение фотографий товаров (для этого ранее было создано поле **gcPhoto**).

Разместите на форму **Form4** (рис 6.1) панель на которую поместите компонент **TDBImage** для отображения в нем фотографий. Для панели сделаем выравнивание по правому краю (свойство **Align** со значением **alRight**).

Для компонента **DBImage** установим следующие свойства:

Свойство	Значение
Align	alTop
AlignWithMargins	True
Center	True
DataSource	DS_T_goods_catalog
DataField	gcPhoto
Name	Goods_Image
Stretch	True
Propotional	True

Свойство **Stretch** говорит о том, что отображаемый рисунок будет уменьшен до размеров области. Свойство **Propotional** указывает на выполнение пропорционального уменьшения размера. Свойство **Center** означает позиционирование отображаемого рисунка по центру области компонента.

После настройки области отображения фотографий на панель **PanelToInputValues** разместим две кнопки которым дадим следующие имена **B_SetImage** – **Назначить картинку** и **B_ClearImage** – **Очистить картинку**. Для того чтобы можно было выбрать картинку на форма необходимо разместить компонент **OpenPictureDialog** (имя по умолчанию **OpenPictureDialog1**). После размещения компонентов форма примет следующий вид (рис 6.1):

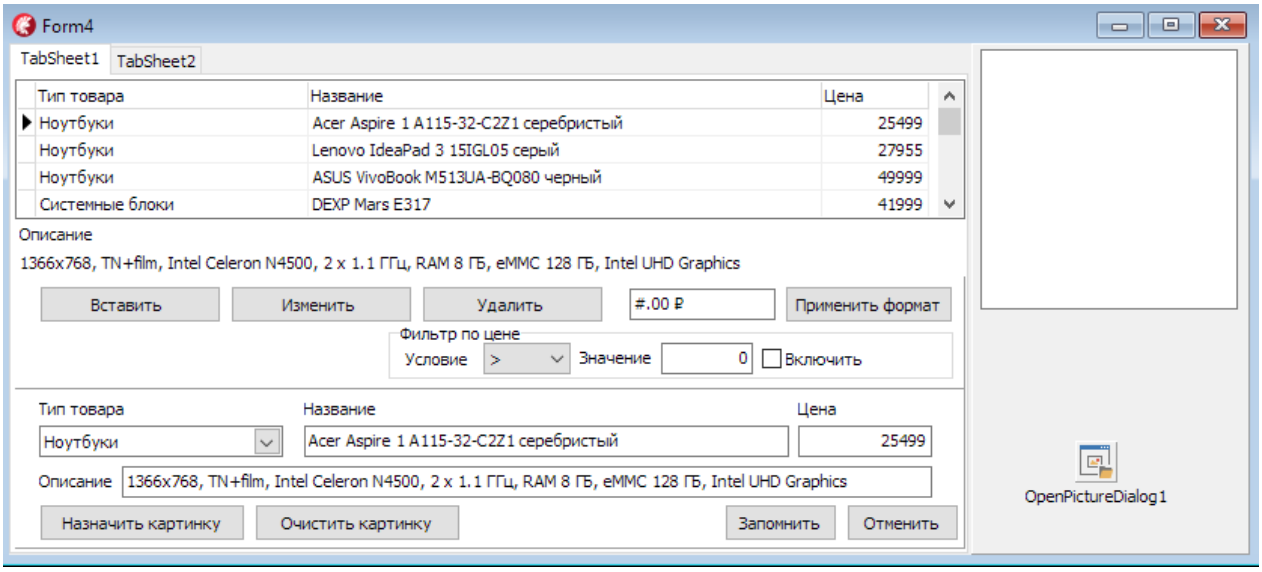

Рис. 6.1. Добавление изображения на форму

Теперь пропишите обработчик события **OnClick** кнопки **B_SetImage** для добавления картинки в базу данных и её отображения в компоненте **Goods_Image**:

```
procedure TForm4.B_SetImageClick(Sender: TObject);
begin
  // Проверка запустилось ли диалоговое окно
  if OpenPictureDialog1.Execute then
    // в случае если пользователь нажал на кнопку Ok в диалоговом окне
    // проверяем существует ли выбранный файл
    if FileExists(OpenPictureDialog1.FileName) then
    begin
      // загружаем выбранный файл в компонент Goods_Image
      Goods_Image.Picture.LoadFromFile(OpenPictureDialog1.FileName);
      // загружаем выбранный файл в поле gcPhoto
      DataModule2.T_goods_cataloggcPhoto.LoadFromFile
        (OpenPictureDialog1.FileName);
    end;
end;
```

Для удаления картинки из поля **gcPhoto** и из компонента **Goods_Image** будет использован обработчик события **OnClick** кнопки **B_ClearImage**:

```
procedure TForm4.B_ClearImageClick(Sender: TObject);
begin
   // вызываем метод очистки поля
  DataModule2.T_goods_cataloggcPhoto.Clear;
  // назначаем значение nil для картинки
  Goods_Image.Picture := nil;
end;
```

В связи с тем, что визуализация данных поля **gcPhoto** осуществляется с использованием отдельного компонента, то в режиме добавления записи нужно предусмотреть очистку картинки в компоненте **Goods_Image**.

Для этого в событие **OnClick** кнопки **InsertButton** необходимо добавить код: `Goods_Image.Picture := nil;`

Измененный код обработчика событий будет выглядеть следующим образом:

```
procedure TForm4.InsertButtonClick(Sender: TObject);
begin
  if DataModule2.T_goods_catalog.State = dsBrowse then
  begin
    // очистить отображаемую картинку
    Goods_Image.Picture := nil;
    // Отключить возможность изменения курсора НД
    // в компоненте DBGrid1
    DBGrid1.Enabled := False;
    // Сделать невидимой панель с кнопками Вставить, Изменить,
    // Удалить
    InsertEditDeletePanel.Visible := False;
    // Сделать видимой панель для ввода значения в запись
    PanelToInputValues.Visible := True;
    // Перевести НД в режим добавления записи, используя
    // метод Append
    DataModule2.T_goods_catalog.Append;
  end;
end;
```

После всех настроек запустите приложение и добавьте изображения к товарам через кнопку **Изменить** или **Вставить** (рис 6.2):

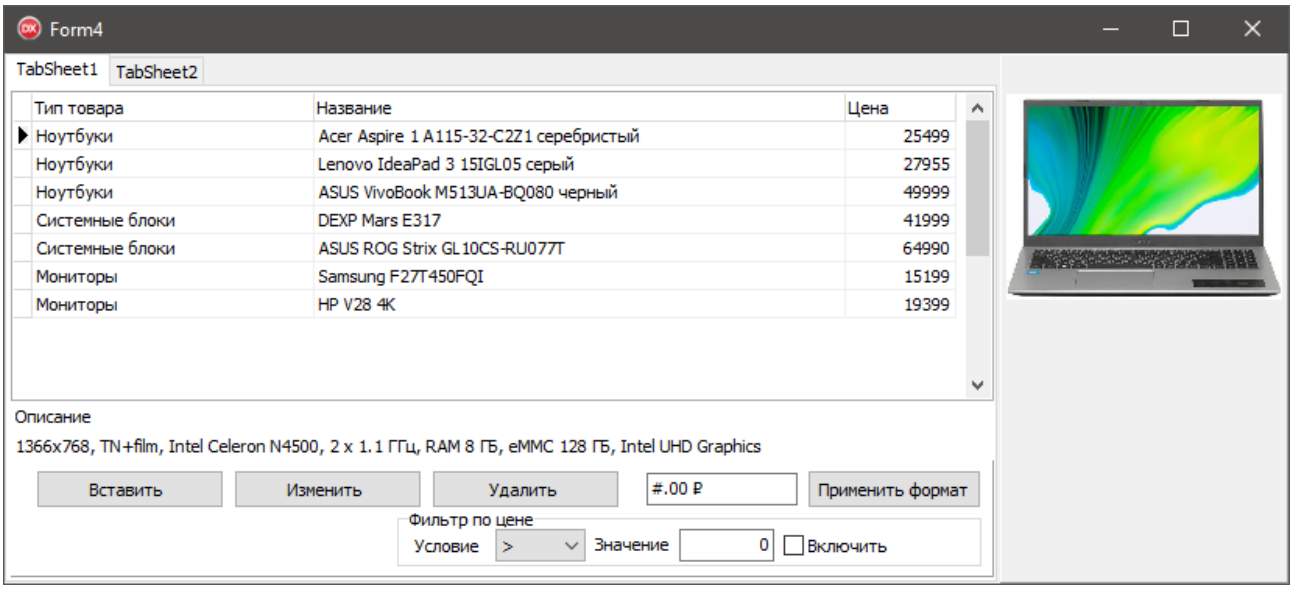

Рис. 6.2. Пример отображения фотографий товаров

6.1 Изменение сортировки полей во время выполнения приложения на основе события OnTitleClick

Часто при просмотре данных в компоненте **DBGrid** появляется необходимость изменить сортировку какого-нибудь столбца.

Наиболее оптимальным решением является написание обработчика события **OnTitleClick**, которое возникает, когда мы щелкаем мышью по заголовкам столбцов в **DBGrid**.

Рассмотрим реализацию сортировки на основе изменения *SQL*-кода на примере сортировки складских запасов (данные в **DBGrid3** на вкладке **Состояние склада**).

В SQL-запросах на выборку сортировка определяется при помощи оператора **ORDER BY**. Для задания сортировки необходимо указать имя поля (или его номер) и тип сортировки. По умолчанию данные сортируются по возрастанию (**ASC**), однако с помощью оператора **DESC** можно задать сортировку по убыванию.

Для реализации механизма сортировки нужно объявить глобальную переменную **sortFlag**, которая будет показывать, какой тип сортировки применялся до этого.

В блоке **var** основной формы (**Form1**) напишем объявление следующего вида:

```
sortFlag: Boolean = True;
```

Теперь создайте обработчик события **OnTitleClick** для **DBGrid3**:

```
procedure TForm1.DBGrid3TitleClick(Column: TColumn);
var
  sql: string;
begin
      DataModule2.Q_v_storage.Active := false;
      sql := 'SELECT * FROM view_storage ORDER BY ' +  Column.FieldName;

      if sortFlag then
        begin
          // Установить сортировку убыванию
          DataModule2.Q_v_storage.SQL.Text := sql + ' DESC';
          sortFlag := false;
        end
      else
        begin
          DataModule2.Q_v_storage.SQL.Text := sql + ' ASC';
          sortFlag := true;
        end;
```

```
      DataModule2.Q_v_storage.Active := true;
end;
```

6.2 Подсветка записей в компоненте DBGrid

Рассмотрим подсветку записей складских запасов в компоненте **DBGrid** значения которых меньше чем значение, которое определил пользователь.

Расширим панель по высоте на вкладке **Состояние склада** и разместим компоненты **Label** и **SpinEdit** (имя по умолчанию **SpinEdit1**). Для компонента **SpinEdit1** установим следующие свойства: **MinValue** в значение **1**, **MaxValue** в значение **1000**, **Value** в значение **1**. В свойство **Caption** компонента **Label** пропишите поясняющий текст. Пример расположения элементов и результата их настройки приведен на рис 6.3:

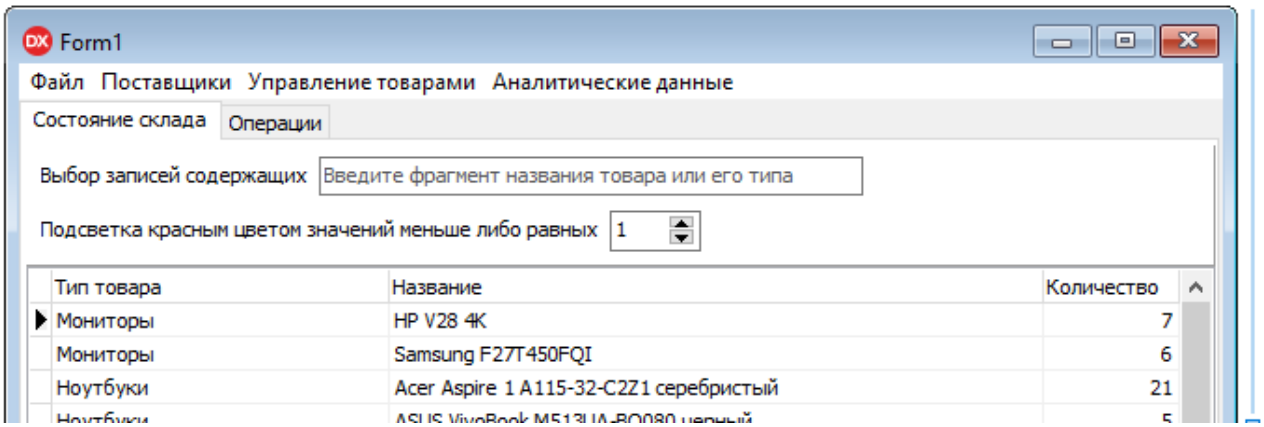

Рис. 6.3. Размещение необходимых компонентов

Необходимо вывести белым шрифтом на красном фоне те строки компонента НД **Q_v_storage**, у которых поле **sCount** содержит значение меньше либо равное значению указанного в **SpinEdit1**.

Для этого будем использовать обработчик события **OnDrawColumnCell** компонента **DBGrid3**, в котором происходит отображение складских запасов:

```
procedure TForm1.DBGrid3DrawColumnCell(Sender: TObject; const Rect: TRect;
  DataCol: Integer; Column: TColumn; State: TGridDrawState);
begin
  with DBGrid3.Canvas do
  begin
    // Проверить, не меньше ли значение поля sCount
    // чем установлено в SpinEdit1
    if (DataModule2.Q_v_storagesCount.value <= SpinEdit1.value) then
    // Если условие выполняется, то задать формат вывода
```

```
    begin
      // Установка красного фона
      Brush.Color := clRed;
      // Если строка (запись) выбрана
      if (gdSelected in State) then
        // Установка бордовый цвет фона
        Brush.Color := clMaroon;
      // Установка белого шрифта
      Font.Color := clWhite;
    end;
  end; // With
  // Отрисовка поля стандартным методом
  DBGrid3.DefaultDrawColumnCell(Rect, DataCol, Column, State);
end;
```

Для того чтобы выделение перерисовывалось при смене значения в **SpinEdit1** необходимо прописать обработчик:

```
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  DBGrid3.Invalidate;
end;
```

Запустите приложение (рис 6.4):

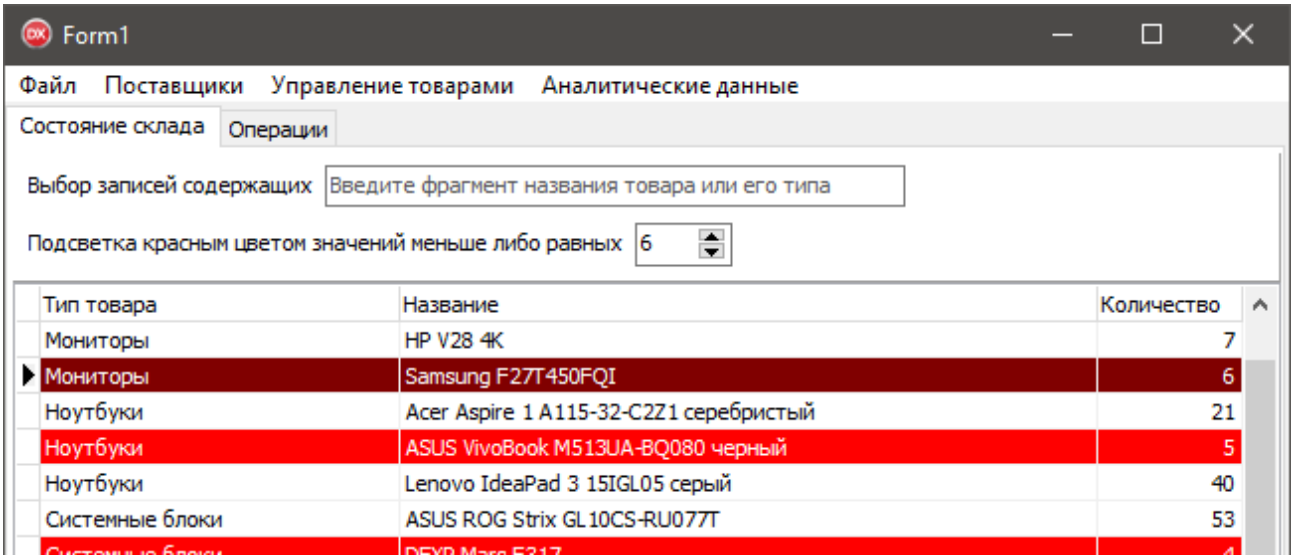

Рис. 6.4. Пример работы подсветки записей

6.3 Настройка шрифтов и цвета записей в DBGrid

Рассмотрим пример реализации динамического изменения свойств подсветки записей. Для этого разместим на форме **Form1** невизуальные компоненты, обеспечивающие диалоги выбора шрифта **FontDialog** и цвета **ColorDialog** (они находятся на вкладке **Dialogs** палитры компонентов), а

также набор компонентов который будет отвечать за настройку выделения записей.

Понадобятся следующие компоненты: **GroupBox** – для группировки элементов, **Panel** для имитации вида подсвеченной записи, а также две кнопки для выбора цвета фона и шрифта (рис 6.5.).

Два компонента **Button** назовем **B_ColorSel** и **B_FontSel**, в свойство **Caption** которых пропишем **Выбор цвета** и **Выбор шрифта** соответственно. Компонент **Panel** для демонстрации оформления назовем **p_ColorRecord**. У нее зададим следующие свойства:

Свойство	Значение
BevelInner	bvLowered
Caption	Цвет
Color	clRed
Font.Color	clWhite
Name	p_ColorRecord

После этого изменим значение свойства **Caption** компонента **Label** отвечающего за пояснение. Пример настройки размещения компонентов приведен на рис 6.5:

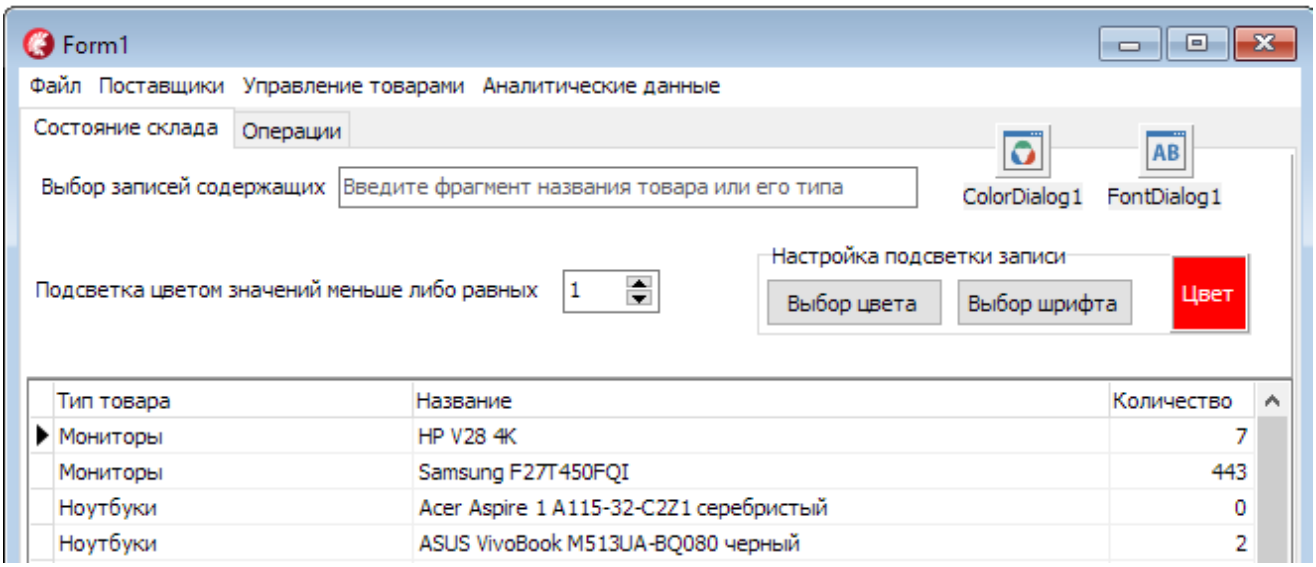

Рис 6.5 Пример размещения компонентов и результата их настройки

Зададим обработчики событий нажатия кнопок, реализующих возможности настройки отображения:

```
procedure TForm1.B_ColorSelClick(Sender: TObject);
begin
  // заносим значение текущего цвета в диалог
  ColorDialog1.Color := p_ColorRecord.Color;
```

```
    // Если запускается диалог выбора цвета
    if ColorDialog1.Execute then
      // Назначить цвет панели примера - p_NA_record
      p_ColorRecord.Color := ColorDialog1.Color;
    //Обновление DBGrid3
    DBGrid3.Invalidate;
  end;

  procedure TForm1.B_FontSelClick(Sender: TObject);
  begin
    // заносим значение текущего шрифта в диалог
    FontDialog1.Font := p_ColorRecord.Font;
    // Если запускается диалог выбора цвета
    if FontDialog1.Execute then
      // Назначить цвет панели примера - p_NA_record
      p_ColorRecord.Font := FontDialog1.Font;
    //Обновление DBGrid3
    DBGrid3.Invalidate;
  end;
```

Для того чтобы учитывать параметры подсветки записей изменим обработчик события **OnDrawColumnCell** для компонента **DBGrid3**.

```
procedure TForm1.DBGrid3DrawColumnCell(Sender: TObject; const Rect: TRect;
  DataCol: Integer; Column: TColumn; State: TGridDrawState);
begin
  with DBGrid3.Canvas do
  begin
    // Проверить, не меньше ли значение поля sCount
    // чем установлено в SpinEdit1
    if (DataModule2.Q_v_storagesCount.value <= SpinEdit1.value) then
    // Если условие выполняется, то задать формат вывода
    begin
      // Установка параметров шрифта и фона не активной записи
      Font := p_ColorRecord.Font;
      Brush.Color := p_ColorRecord.Color;
      // Если строка (запись) выбрана
      if (gdSelected in State) then
        // Установка бордовый цвет фона
        Brush.Color := clMaroon;
      // Установка белого шрифта
      Font.Color := clWhite;
    end;
  end; // With
  // Отрисовка поля стандартным методом
  DBGrid3.DefaultDrawColumnCell(Rect, DataCol, Column, State);
end;
```

После изменения кода можно проверить, как работает приложение, задав во время работы иные оттенки цвета для шрифта. Пример функционирования приведен на рис. 6.6:

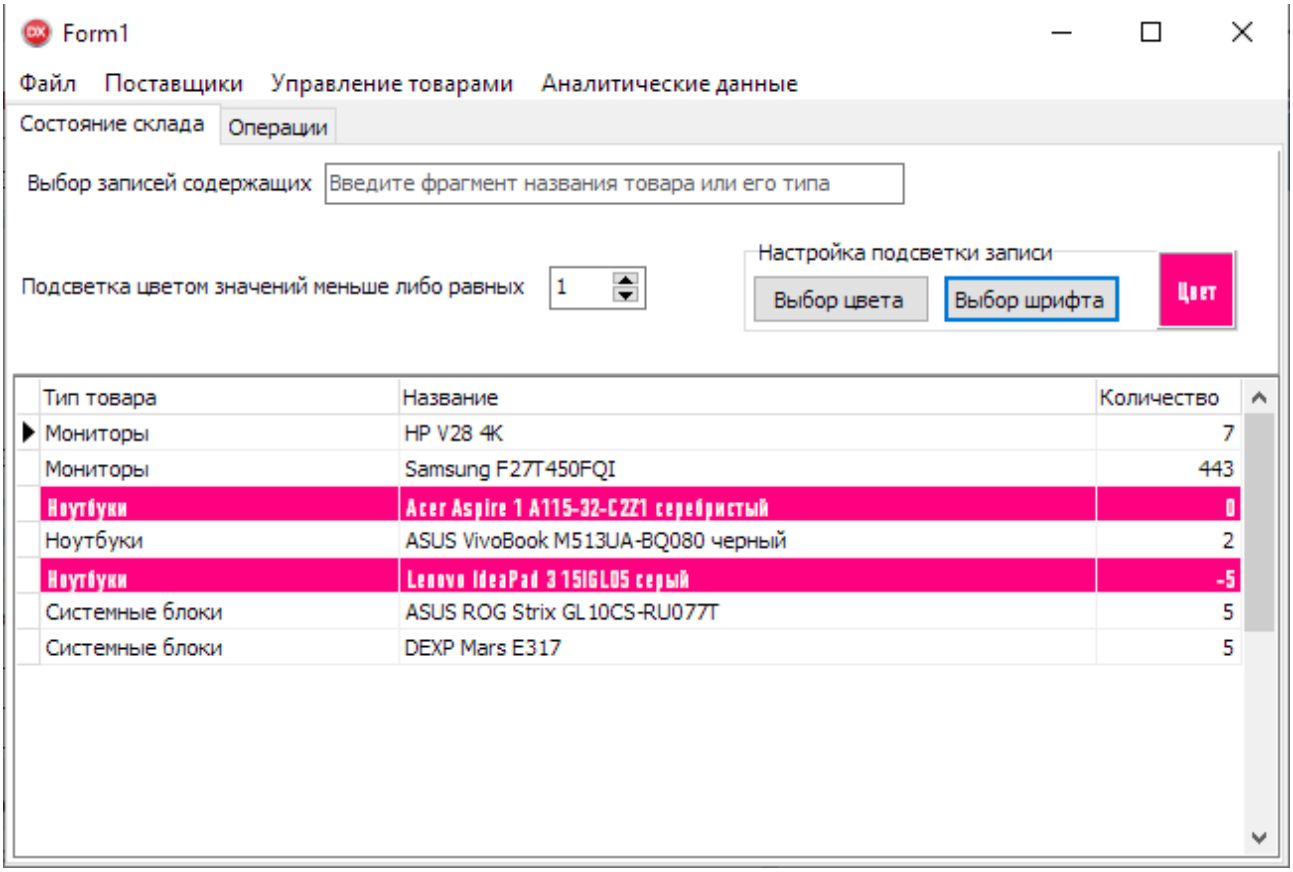

Рис. 6.6. Пример работы подсветки во время выполнения приложения