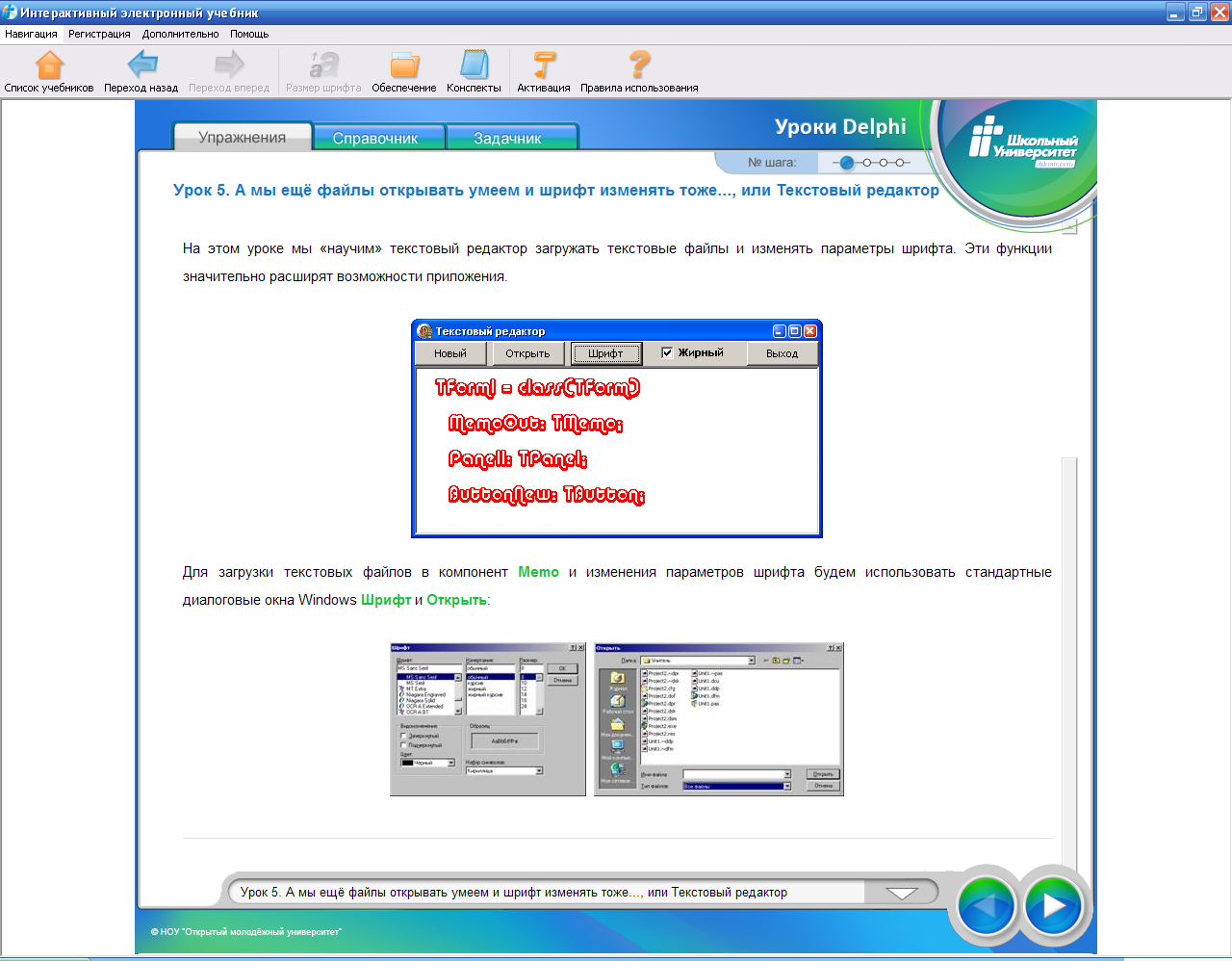
**Практическая работа №2 «Текстовый редактор»**

**Цель:** Научиться работать с компонентом **Memo**; с параметрами шрифта и компонентами **FontDialog**; загружать текстовые файлы в компонент **Memo** с использованием диалога **OpenDialog**; работать с компонентом **ChekBox** (переключатель).

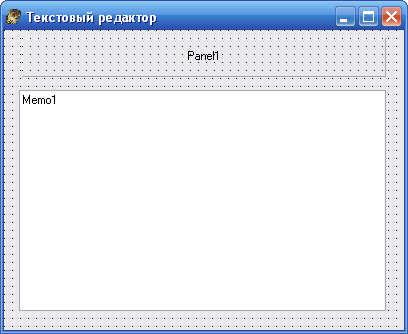
Приложения для работы с текстом играют важную роль в жизни большинства людей. С помощью таких приложений можно набирать и редактировать текстовые документы, сохранять их на диск, распечатывать и т.д.

Создадим простой текстовый редактор с возможностью очистки содержимого рабочей области. Этот редактор может пригодиться при работе с текстовыми документами.

В среде Delhi такое приложение сделать очень легко, так как имеющийся в ней компонент **Memo** уже умеет многое из того, что нужно для текстового редактора: работать с многострочным текстом, с буфером обмена, с диском, вставлять и удалять символы, выделять текст, изменять атрибуты шрифта.

***Дизайн приложения***

**1.** Поместите компонент **Memo** (вкладка **Standard**) на форму. Также добавьте компонент-контейнер **Panel**. В нём мы разместим кнопки, управляющие приложением. После этого очистите у компонента **Panel** свойство ***Caption*** и измените заголовок формы на надпись «Текстовый редактор»

Компонент **Panel** обладает очень важной особенностью. Он является компонентом – контейнером. Если на него поместить другие компоненты, то их перемещение будет ограничено границами компонента-контейнера. При перемещении компонента-контейнера по форме всё его содержимое будет перемещаться вместе с ним.

**2.** Чтобы понять принцип работы компонентов-контейнеров, посмотрите на содержимое докера ***Structure***. В нём хранится список всех компонентов, размещённых на форме. Из него видно, что главный контейнер – это форма (**Form1**). На ней расположены два компонента: **Memo1** и **Panel1**.

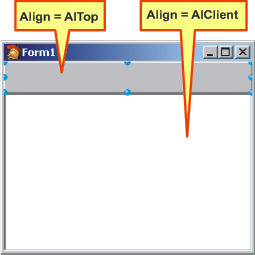
|  |
| --- |
|  |

**3.** Увеличьте размер панели до размеров, вдвое больших, чем у многострочного редактора.

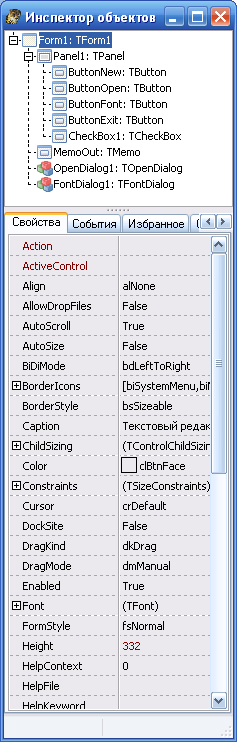
**4.** Сохраните проект.

**5.** Запустите проект и убедитесь, что вы можете легко набирать и редактировать текст в поле **Memo**. Но есть некоторые «минусы»: при загрузке приложения компонент **Memo** содержит совершенно ненужную надпись *Memo1*; при изменении размеров формы размер поля **Memo** не меняется, что выглядит некрасиво; при увеличении окна появляется много свободного места, а при уменьшении – появляются полосы прокрутки.

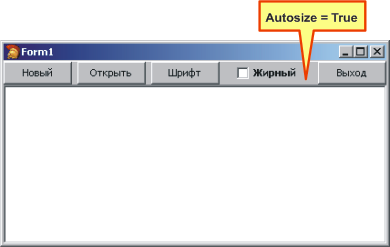
**6.** Чтобы удалить надпись *Memo1* в компоненте **Memo**, необходимо у него очистить свойство ***Lines***. Это аналог свойства ***Text*** компонента **Edit**, только в данном случае текст можно вводить многострочный. Итак, в компоненте **Memo** напротив свойства ***Lines*** нажмите на кнопку с тремя точками, откроется дополнительное окно. Удалите весь содержащийся там текст.

**7.** Настроим компонент **Memo** так, чтобы его границы автоматически изменялись при изменении размеров формы с помощью свойства ***Align***(переводится как «выровнять»). Компонент **Panel** должен «прилипнуть» к верхней части окна (***Align=alTop***), а компонент **Memo** должен занять всю остальную площадь (***Align=alClient***)

**8.** Запустите приложение и поменяйте размер формы. Размеры компонентов автоматически поменяются в зависимости от размеров формы. Закройте приложение.

**9.** В компонент **Panel** добавьте четыре кнопки **Button** и измените их заголовки. Назначение кнопок понятно из их заголовков (свойство ***Captiоn***). Дайте им следующие имена (свойство ***Name***): **Новый – ButtonNew; Открыть – ButtonOpen; Шрифт – ButtonFont; Выход – ButtonExit; Memo1 – MemoOut.**

Обратите внимание на структуру проекта – кнопки д.б. компонентами компонента - контейнера **Panel1**

**10.** Добавьте из категории ***Standard*** в контейнер **Panel** компонент **CheckBox** (переключатель). Он будет выполнять функцию быстрого изменения атрибута шрифта ***Жирный***. Переименуйте его (измените свойство ***Name***) в **CheckBold** (Bold переводится как «жирный»). Измените заголовок переключателя (свойство ***caption***) на ***Жирный***.

**11.** Установите свойство ***AutoSize = True*** для панели **Panel**. Размер панели автоматически изменится так, чтобы вместить все находящиеся на ней компоненты.

**12.** Запустите приложение. У нас получился почти полноценный редактор текста. Но вот беда… Текст мы вводить можем, но ни одна кнопка не работает. Для исправления этого досадного обстоятельства необходимо **написать** **обработку события OnCliсk** **для каждой кнопки**.

**13.** Создайте обработчик события **OnClick** для кнопки **Выход**

**14.** Выберите кнопку **Новый** и создайте для неё обработчик события **OnClick**. В нём запишите всего один метод для компонента **Memo** – ***Clear***, который очищает содержимое этого компонента:

**Procedure TForm1.ButtonNewClick (Sender: TObject);**

**Begin**

**MemoOut.Clear;**

**End;**

***Открытие файла***

**1.** Из палитры компонентов, категория ***Dialogs***, поместите на форму невизуальные компоненты: **FontDialog**  и **OpenDialog**. В какое место формы мы их поместим – неважно, потому что они видны только на этапе разработки приложения.

* Компонент FontDialog – вызывает окно стандартного диалога выбора параметров шрифта Шрифт.
* Компонент OpenDialog – вызывает окно стандартного диалога выбора файлов Открыть.

Двойной щелчок по этим компонентам открывает стандартный диалог (а не событие, как это было у привычных для нас компонентов), Это диалоговое окно будет появляться при обращении к нему из работающего приложения (***Run Time***). Это очень удобно, так как в режиме конструирования (***Design Time***) можно посмотреть, как выглядит окно диалога.

**2.** Создайте обработчик события **OnClick** для кнопки **Открыть**. Он должен: вызывать стандартный диалог открытия файлов; загружать указанный пользователем файл в многострочный редактор (компонент **Memo**).

Запишите в обработчике следующий код:

**Procedure TForm1.ButtonOpenClick(Sender: TObject);**

**Begin**

*//Вызываем диалог открытия*

**OpenDialog1.Execute;**

**End;**

Метод ***Execute*** служит для вызова стандартного диалога Windows у всех компонентов (для каждого компонента свой диалог), находящихся на вкладке ***Dialogs***.

Обратите внимание на текст, находящийся после знаков **//**, в Delhi он по умолчанию выделен синим цветом. Это комментарии – второй по значимости элемент после кода программы. Чтобы не запутаться в записанной программе, необходимо комментировать практически каждую написанную вами строку!

**3.** Запустите приложение и убедитесь, что при выборе пункта **Открыть** появляется одноименное диалоговое окно. Однако какой бы файл вы не выбрали, ничего не произойдет. Это вполне логично – ведь в обработчике нигде не сказано, что нужно делать с выбранным файлом. Закройте приложение.

**4.** Для того чтобы выбранный файл загружался в многострочный редактор, допишите в обработчик следующий код:

Procedure TForm1.ButtonOpenClick(Sender: TObject);

Begin

*//Вызываем диалог открытия*

OpenDialog1.Execute;

*//Загружаем содержимое файла в компонент MemoOut*

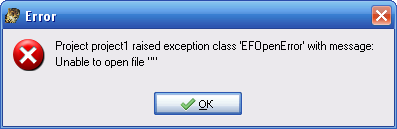
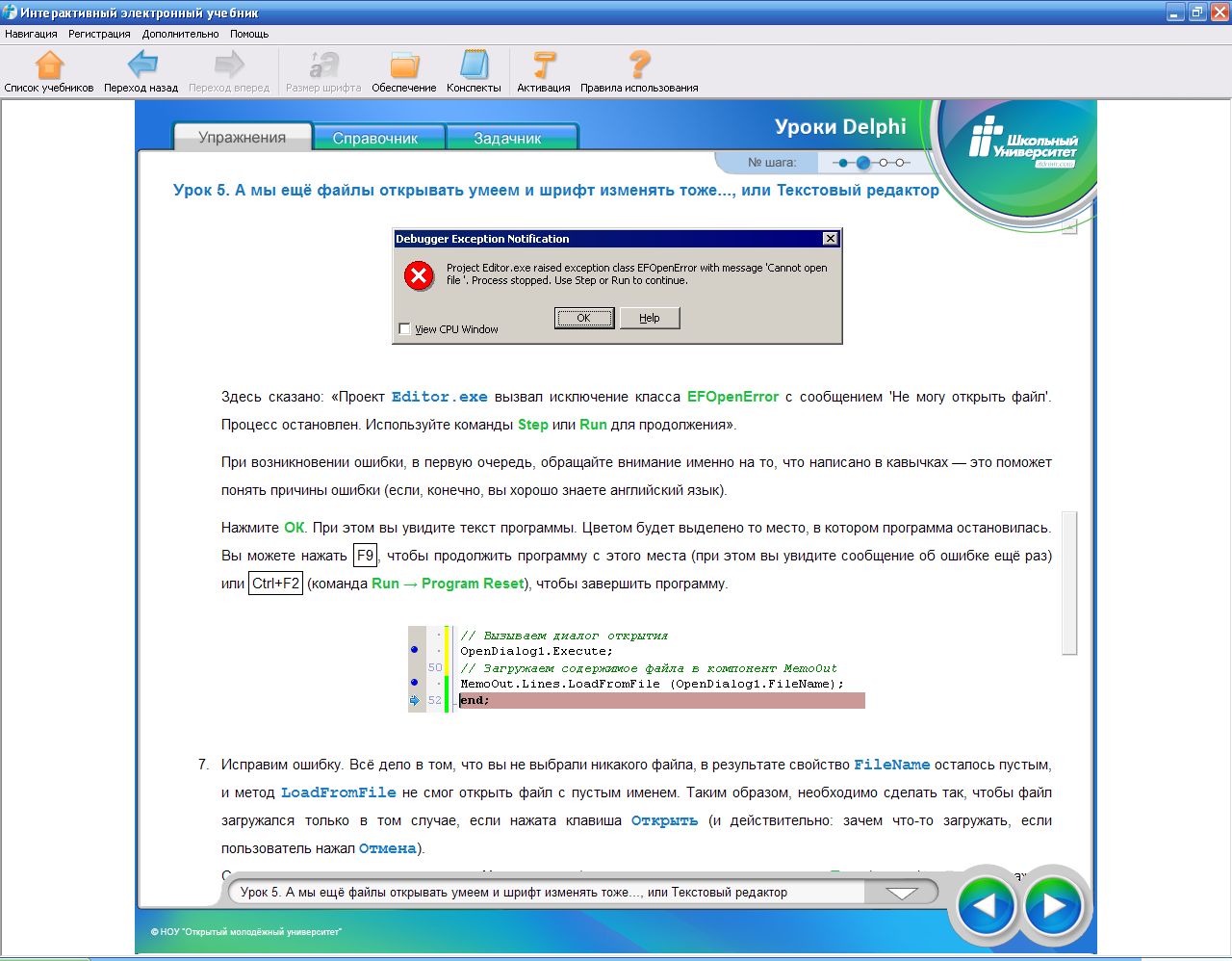
**MemoOut.Lines.LoadFromFile (OpenDialog1.FileName);**

End;

Что это означает? После того как пользователь выбирает файл и нажимает кнопку **Открыть**, имя файла и путь к нему помещаются в свойство ***FileName*** компонента **OpenDialog1**, Потом содержимое этого файла загружается в многострочный редактор. Для этого используется метод ***LoadFromFile*** класса **TStrings**. Для доступа к этому классу используется свойство компонента ***Lines***.

**5.** Запустите приложение и откройте любой файл (например *demo.txt*). Его содержимое появится в поле компонента **Memo**, и вы сможете его редактировать.

Закройте приложение и снова запустите. Опять выберите пункт **Открыть**, но в диалоговом окне нажмите кнопку **Отмена**. В результате появится сообщение об ошибке:

Здесь сказано: «Проект вызвал исключение класса EFOpenError с сообщением ‘Не могу открыть файл’. Процесс остановлен. Используйте команды Step или Run для продолжения»

При возникновении ошибки, в первую очередь, обращайте внимание на то, что написано в кавычках – это поможет понять причины ошибки.

Нажмите **ОК**. При этом вы увидите текст программы, в которой цветом будет выделено то место, в котором программа остановилась. Вы можете нажать клавишу **F9**, чтобы продолжить программу с этого места (при этом вы увидите сообщение об ошибке ещё раз) или нажать **Ctrl+F2** (команда Run → Program Reset), чтобы завершить программу.

**6.** Исправим ошибку. Всё дело в том, что вы не выбрали никакого файла, в результате свойство ***FileName*** осталось пустым, и метод ***LoadFromFile*** не смог открыть файл с пустым именем. Таким образом, необходимо сделать так, чтобы файл загружался только в том случае, если нажата клавиша **Открыть** (и действительно: зачем что-то загружать, если пользователь нажал **Отмена**).

Сделать такую проверку очень легко. Метод ***Execute*** возвращает логическое значение True (Истина), если была нажата кнопка **Открыть**, и False (Ложь), если была нажата **Отмена**. Остаётся только проверить это значение с помощью оператора условного перехода:

Procedure TForm1.ButtonOpenClick(Sender: TObject);

Begin

*//Вызываем диалог открытия и проверяем условие нажатия на кнопку «ОК»*

**If** OpenDialog1.Execute **then**

*//Загружаем содержимое файла в компонент MemoOut*

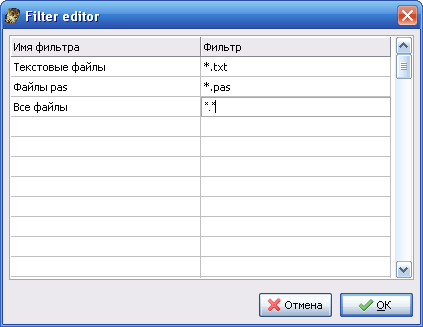
MemoOut.Lines.LoadFromFile (OpenDialog1.FileName);

End;

Обратите внимание, что в условии написано ***OpenDialog1.Execute***, вместо ***OpenDialog1.Execute=True***. Эти две записи полностью идентичны, но первая более простая и рекомендуется использовать именно её.

**7.** Запустите приложение и убедитесь, что теперь всё работает корректно.

Заметьте, что при вызове диалога открытия появляется список всех файлов, которые есть в выбранном каталоге. Но нам не нужен весь список. В приложениях типа Word, Photoshop и много других в нижней части диалога открытия файлов есть выпадающий список **Тип файлов**. С его помощью можно выбрать только те типы файлов, которые будут отображаться в окне.

**8.** Для того чтобы пользователь мог выбирать тип открываемых файлов, у компонента **OpenDialog** есть свойство ***Filter***. Откройте редактор фильтра, нажав на кнопку с тремя точками. В левой его части, построчно, записываются имена фильтров, которые будут отображаться в поле *Тип файлов*, а в правой непосредственно сам фильтр (правила задания фильтра стандартные для Windows):

***Форматирование текста***

**1.** При нажатии на кнопку **Шрифт** должны происходить следующие действия:

* все параметры шрифта (название, размер, цвет, стиль и пр.) компонента **MemoOut** передаются компоненту **FontDialog**. При этом в стандартном диалоговом окне **Шрифт** автоматически выставляюся требуемы параметры;
* открывается стандартный диалог **Шрифт**;
* проверяется, что пользователь нажал кнопку **ОК**, чтобы программе не прогонять оставшиеся два пункта «вхолостую»;
* параметры шрифта из компонента **FontDialog** передаются компоненту **MemoOu**t;
* изменяются, если необходимо, состояния переключателя **CheckBold**.

Создайте обработчик события **OnClick** для кнопки **Шрифт** и запишите в нём следующий код:

**Procedure TForm1.ButtonFontClick(Sender: TObject);**

**Begin**

*//Передаём компоненту FontDialog параметры шрифта из компонента Memo*

**FontDialog1.Font := MemoOut. Font;**

*//Открываем диалог шрифта*

**If** FontDialog1.Execute **then**

**begin**

*//Передаём компоненту Memo параметры шрифта из компонента FontDialog*

**MemoOut. Font := FontDialog1.Font;**

*// Изменяем, если необходимо, состояние переключателя*

**CheckBold.Checked := fsbold in FontDialog1.Font.Style;**

**End;**

**End;**

Поясним код.

Замечательной особенностью ООП является возможность присвоения значений одного компонента другому, если они принадлежат одному классу. Например, свойство ***Font*** (размер шрифта, цвет и пр) класса **TFont** присутствует в объектах **Memo** и **FontDialog**, поэтому достаточно написать: FontDialog1.Font := MemoOut. Font, чтобы передать все параметры свойства ***Font*** от объекта **Memo** объекту **FontDialog**.

Отметим особенность установки переключателя Жирный: fsbold in FontDialog1.Font.Style. Дело в том, что свойство ***Style*** у класса **TFont** имеет тип «множество». И данная конструкция проверяет: входит ли значение ***fsbold*** в множество значений ***Style***, и если входит, то возвращает значение True, иначе - False.

**2.** Теперь запустите приложение и порадуйтесь тому, как легко изменять параметры шрифта с помощью стандартного диалога, специально для этого предназначенного.

**3.** Переключатель **CheckBox** с именем **CheckBold** на панели **Panel** предназначен для более быстрой установки жирного шрифта. Создайте обработчик события **OnClick** для переключателя и запишите в нём следующий код:

**Procedure TForm1.CheckBoldClick (Sender: TObject);**

**Begin**

*//Проверка состояния переключателя CheckBold*

*//Если включен, то текст в MemoOut делаем жирным*

**If** CheckBold.Checked **then** MemoOut.Font.Style := MemoOut.Font.Style +[fsBold]

*//если выключен, то текст в MemoOut делаем светлым*

**Else** MemoOut.Font.Style := MemoOut.Font.Style - [fsBold]

**End;**

**4.** Запускаем приложение. Всё работает!

***Контрольные вопросы***

**1.** Какой метод служит для открытия диалога всех компонентов, расположенных в категории Dialogs палитры компонентов? Значения какого типа он возвращает?

**2.** Какой тип имеет свойство Style класса TFont?

**3.** В случае появления сообщения об ошибке как можно продолжить работу приложения? Как остановить?

**4.** Как в диалоговом окне открытия файлов можно задать фильтры для типов файлов?

***Задания для самостоятельного выполнения***

**1.** Добавить к созданному текстовому редактору функцию сохранения текстового файла через компонент **SaveDialog**, категории ***Dialogs***. Для этого добавьте новую кнопку **Сохранить** рядом с кнопкой **Открыть**. (Сохранение текста осуществляется вызовом метода ***SaveFromFile*** свойства ***Lines*** компонента **Memo**)

**2.** Добавить к созданному текстовому редактору функцию выбора курсивного шрифта через компонент **CheckBox**, аналогично выбору жирного шрифта. Для этого добавьте на панель новый переключатель.

**3.** Добавить к созданному текстовому редактору функцию выбора цвета фона у компонента **Memo** через компонент **ColorBox**.

*Литература:*

*Интерактивный электронный учебник «Школьный университет» ТУСУР, г.Томск*