Семинар 4.

Примеры задач.

*Читайте коды программ! Там много комментариев с тонкостями.*

№1. На форму вывести надпись, буквы которой будут последовательно исчезать (начиная с конца). Как только надпись исчезнет, форма должна постепенно становиться все более прозрачной и постепенно «растаять». Затем изображение формы постепенно восстанавливается, но надпись на форме уже другая. Весь процесс начинается с нажатия кнопки в изначальной форме. Программа демонстрирует возможности компонента **Timer**, и свойства **Opacity**, определяющего прозрачность формы.

**Описание класса Form**:

<http://msdn.microsoft.com/ru-ru/library/system.windows.forms.form.aspx>

**Описание Button**:

<http://msdn.microsoft.com/ru-ru/library/system.windows.forms.button.button.aspx>

№2. Разработать Windows-приложение. В поле **ListBox** вывести в виде списка элементы массива строк. Выделяя элемент списка, удалять его и из списка, и из массива. Обеспечить возможность восстановления начального состояния списка и массива. Некоторые свойства элементов задавать в конструкторе форм, другие – в коде программы.

**Описание ListBox:**

<http://msdn.microsoft.com/ru-ru/library/system.windows.forms.listbox.aspx>

№3. Последовательность событий при работе с формой

• Создайте пустую форму.

• Добавьте в нее обработчики событий: **Activated**, **Deactivate**, **FormClosed**, **FormClosing**, **Load**, **Paint**, **Resize**.

• В каждый обработчик включите оператор, изменяющий текст заголовка формы, и оператор, добавляющий в общую строку название события.

• В обработчике события **Form1\_FormClosed()** поместите вызов диалогового окна, где выведите список событий, произошедших при выполнении программы.

• В каждый обработчик событий, кроме **Activated**, **Deactivate**, поместите вывод в диалоговое окно названия события.

• Запустите программу на выполнение.

• Запишите названия произошедших событий, изменяемые заголовки формы.

• Сравните с результатом, выведенным в обработчике события **FormClosed**.

№4. В библиотеке классов **Mathematics** находятся:

• Класс **Expression** - представляет математическое выражение.

Поле **ex** - ссылка на метод-выражение, **exEvent** – событие, происходящее при смене выражения, **ExVal** – метод вычисления значения выражения для заданного значения аргумента, конструктор.

• Класс **ValueStore** - хранит значение выражения.

Поле **exp** – ссылка на выражение, **x0** – значение аргумента, **expCurrValue** – значение выражения, **CurrVal** – ссылка на **expCurrValue**, Конструктор: **ValueStore** (*Expression* **e**, *double* **x**)

• Классы находятся **Expression** и **ValueStore** в отношении агрегации.

В основной программе создать объект **me** класса **Expression**, использовать ссылку **me** в конструкторе объекта **vs** класса **ValueStore**. Задавая разные выражения поля **me.ex**, выводить значения **vs.expCurrValue**.