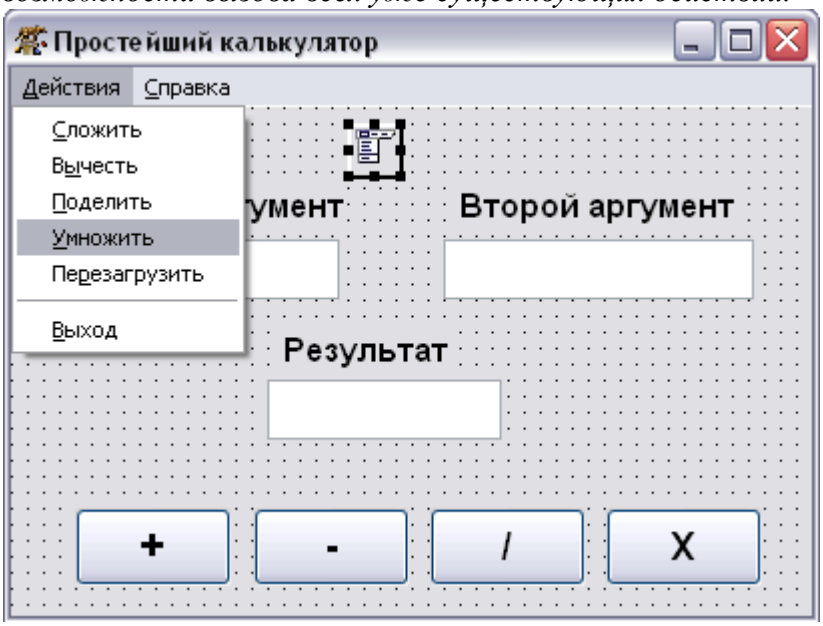
Лабораторная работа № ##
Часть 1. Компонент Меню (TMainMenu)

Вернемся к проекту «Простейший калькулятор» (Если проекта нет, создаем его заново).

Задача: *Добавить к приложению главное меню, которое реализует дополнительные возможности вызова всех уже существующих действий.*



Далее перетаскиваем с панели элементов управления на форму компонент **TMainMenu** . С помощью свойства Items, либо двойным щелчком на компоненте переходим в режим редактирования главного меню. Далее в **Редакторе меню**, щелкнув правой кнопкой мыши на элементе меню, раскроется список доступных действий. Чтобы создать такое меню, как показано на рисунке, нам понадобятся команды **Вставить новый пункт (после)**, а также **Создать подменю**.

После того, как все пункты меню созданы, необходимо для каждого пункта создать свои обработчики события **OnClick**. Для этого на форме в режиме конструктора можно выбрать необходимый пункт меню и кликнуть по нему.

Процедуры сложения, вычитания и т.д. уже написаны в нашей программе, поэтому нет необходимости в коде процедуры щелчка по пункту меню писать все эти действия заново, достаточно вызвать уже существующую процедуру нажатия по кнопке сложения, вычитания и т.д. Вот пример обработки события щелчка по пункту меню **Сложить**:

```
procedure TForm1.MenuItem2Click (Sender: TObject);    {Щелчок по пункту Сложить}
begin
Button1Click(Sender);          {Процедура щелчка по кнопке Button1 с заголовком «+»}
end;
```

Задание 1: Дополните программу действиями:

1) Все пункты меню должны работать, в пункте **Справка** вложен пункт **Помощь (F1).**
2) По нажатию на пункт **Перезагрузить** текстовые поля должны очищаться.
3) В процедуре выбора пункта Выход содержится следующий код:
   if CloseQuery then Close; *{Безопасный выход из приложения}* Такой способ выхода необходимо далее применять во всех разрабатываемых проектах.

Часть 2. Обработка событий, связанных с мышью.

Событие **OnMouseMove()**

Задача: *Создать событие движения курсора мыши по форме, в результате движения в заголовок формы постоянно выводятся текущие координаты курсора.*


```
procedure TForm1.FormMouseMove (Sender: TObject; Shift: TShiftState; X,  Y: Integer);
begin
end;
```

Обратите внимание на параметры **X** и **Y** целого типа в описании процедуры, это и есть координаты, которые нужно выводить в заголовок формы.

Параметр **Shift (переменная типа множество)** определяет ограниченное множество клавиш, которые могут быть нажаты в процессе перемещения мыши. Этот параметр может принимать несколько значений:

ssShift – нажата клавиша SHIFT
ssAlt – нажата клавиша ALT
ssCtrl – нажата клавиша CTRL
ssLeft – левая кнопка мыши
ssRight – правая кнопка мыши
ssMiddle – средняя кнопка мыши
ssDouble – выполнен двойной щелчок

Например, чтобы проверить в этой процедуре нажата ли клавиша SHIFT, нужно написать:

if Shift = [ssShift] then …

Или, например, необходимо проверить нажаты ли одновременно SHIFT и ALT:

if Shift = [ssShift, ssAlt] then …

Или проверка «нажата ли хотя бы одна из клавиш?»

if Shift <> [ ] then …

Задание 2 : Дополните программу действиями:

1) Координаты должны отображаться при движении курсора, при условии, что нажата клавиша CTRL
2) Координаты должны отображаться при движении курсора, при условии, что нажата либо клавиша CTRL, либо SHIFT, но не должны отображаться, если эти клавиши нажаты вместе и если ничто не нажато. *(сложное условие)*

События **OnMouseDown()** и **OnMouseUp().**

Эти два события сходны по своим параметрам, различие лишь в том, что **OnMouseDown()** возникает при нажатии кнопки мыши (и выполняется непрерывно пока кнопка остается нажатой), а **OnMouseUp()** выполняется когда кнопку мыши отпускают (выполняется единожды).

procedure *TForm1.FormMouseUp (Sender: TOBject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);*

begin

end;

С параметрами **X**, **Y** и **Shift** вы уже знакомы. Параметр **Button** здесь показывает какая из кнопок мыши была нажата: **mbLeft**, **mbRight** или **mbMiddle.** С помощью процедуры **OnMouseUp()** мы создадим возможность вызова всплывающего меню в приложении, по нажатию на правую кнопку мыши.

Часть 3. Компонент Контекстное меню (TPopupMenu).

Задача: *Обеспечить работу контекстного меню, которое должно содержать те же пункты, что и главное.*

Для выполнения поставленной задачи

1) Размещаем на форме элемент **TPopupMenu** 
2) Создаем необходимые пункты (редактирование аналогично **TMainMenu**)
3) Выделяем форму и создаем для формы событие **OnMouseUp(),** после которого и будет всплывать меню.

```
procedure TForm1.FormMouseUp(Sender: TOBject; Button: TMouseButton;  Shift: TShiftState; X, Y: Integer);
  var P: TPoint;
begin
  P := GetClientOrigin;
  if Button = mbRight then PopupMenu1.PopUp(P.X + X , P.Y + Y) ;
end;
```

Здесь мы описали дополнительную переменную **P** типом данных **TPoint** - это запись, в которой существует два поля **X** и **Y**. Фактически методом **GetClientOrigin** мы определили верхнюю левую координаты угла нашей формы и сохранили эти координаты в переменной **P.** Далее, проверили, если нажата правая кнопка мыши, то меню ***PopupMenu1*** всплывает с помощью метода **PopUp** в точке ***(P.X + X , P.Y + Y),*** где **X** и **Y** – координаты курсора мыши на форме, а **P.X** и **P.Y** – координаты верхнего левого края формы относительно экрана монитора. Координата точки для всплытия меню ***PopupMenu1*** отсчитывается от верхнего левого конца экрана, а не формы, поэтому нам нужны эти преобразования.

Задание 3:

1) Вместо P.X и P.Y подставьте значения свойств Left и Top формы, посмотрите что получится, и ответьте на вопрос: «Почему нельзя использовать свойства Left и Top, а нужно пользоваться методом **GetClientOrigin ?**

Лабораторная работа №
Часть 1. Компонент Текстовая область (TMemo)

Используйте известные вам приемы и методы работы со списками, для создания следующего приложения (см. рисунок).

Задача: *Создать приложение для простейших операций со списками. Программа должна уметь:*

1) ***Добавить*** *строку в список, введенную в поле* ***Текст****, если поле* ***Позиция*** *пусто, то новая строка добавляется в конец списка. Если же в поле* ***Позиция*** *число, то кнопка* ***Добавить*** *работает аналогично кнопке* ***Изменить*** *(см. пункт 3).*
2) ***Вставить*** *строку в список в указанную* ***Позиция****, если поле* ***Позиция*** *пусто, то вставка происходит в конец списка (см. пункт 1).*
3) ***Изменить*** *строку с индексом, введенным в поле* ***Позиция*** *(если пусто – ничего не происходит).*
4) ***Удалить*** *строке с индексом, введенным в поле* ***Позиция****(если пусто – ничего не происходит).*
5) ***Поиск*** *текста введенного в поле* ***Текст****, номер позиции найденного текста в списке выводится в поле* ***Позиция****. Если текст не найден, поле* ***Позиция*** *– пусто.*
6) ***Сохранить*** *список в файле, путь которого прописан в поле* ***Путь к файлу****.*
7) ***Загрузить*** *список из файла, путь которого прописан в поле* ***Путь к файлу****.*
8) ***Очистить*** *поле со списком (поле* ***Список****).*
9) ***Отсортировать*** *список в алфавитном порядке (необратимая операция).*
10) ***Индексировать*** *– в начале каждой строки в списке добавляется ее индекс (позиция). Например: ‘Нулевая строка’ -> ‘[0] – Нулевая строка’ (необратимая операция).*
11) ***Выход*** *из приложения (см. предыдущую лаб.раб. -> Часть 1 -> Задание 1 -> Пункт 3).*