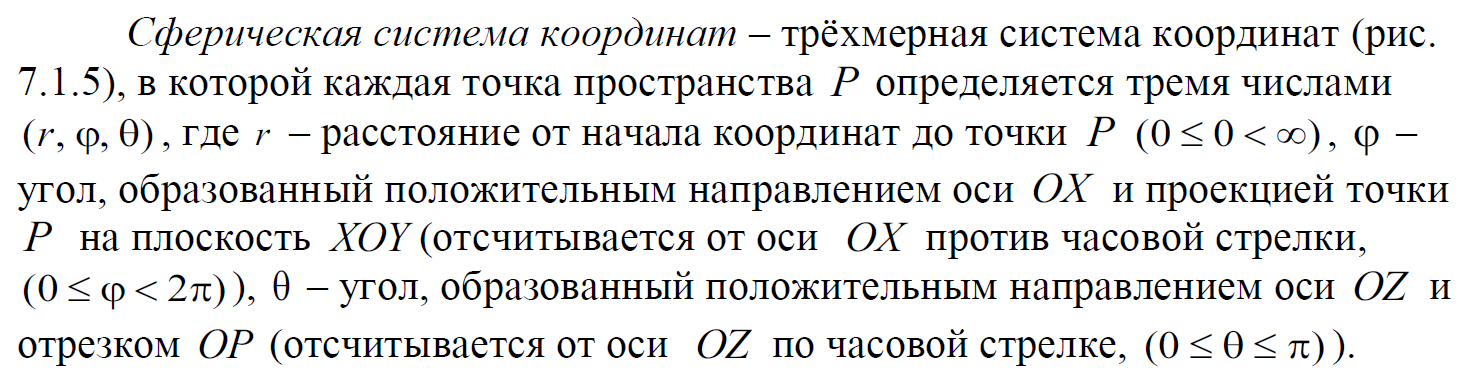
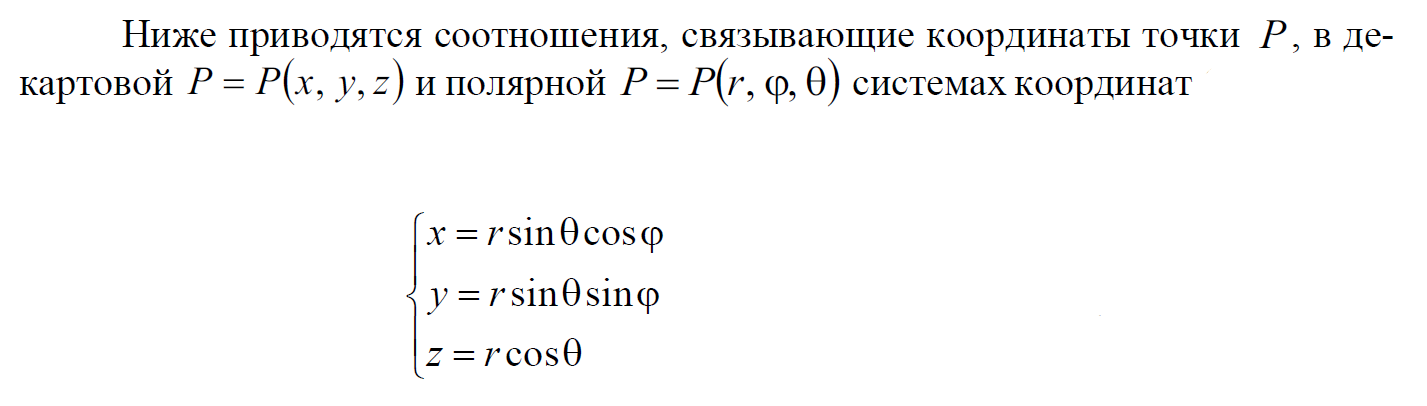
**Лабораторная работа № 1**

***Тема: Работа с классом CMatrix***

**1) Сферическая система координат. Переход из сферической в декартову систему и наоборот.**







**2) Карта сообщений. Дескрипторы окна.**

**Карта сообщений**

Макрос – это набор инструкций, которые сообщают программе (такой как Word или Excel), какие действия следует выполнить, чтобы достичь определенной цели.

Windows взаимодействует с приложением, *посылая ему сообщения*. Каждый раз, когда происходит такое событие, как нажатие клавиши или щелчок мышью, приложению, которое затем должно обработать это событие, отправляется сообщение. Поэтому обработка сообщений является ядром всех приложений. В традиционных приложениях Windows (написанных с использованием только API) каждое сообщение передается в качестве аргументов оконной функции. Там обычно с помощью большого оператора switch определяется тип сообщения, извлекается информация и производятся нужные действия. Это громоздкая и чреватая ошибками процедура. С помощью MFC все делается намного проще.

В модели программирования MFC «карты сообщений» используются для обозначения того, какие функции будут обрабатывать различные сообщения для определенного класса. Карты сообщений содержат один или несколько макросов, которые определяют, какие сообщения будут обрабатываться какими функциями.

Чтобы программа могла ответить на сообщение, в карту сообщений должна быть включена соответствующая макрокоманда. Названия макрокоманд соответствуют именам стандартных сообщений Windows, но дополнительно имеют префикс ON\_ и заканчиваются парой круглых скобок. Из этого правила есть исключение: сообщению WM\_COMMAND соответствует макрокоманда ON\_COMMAND(). Причина в том, что это сообщение обрабатывается особым образом.

Чтобы включить макрокоманду в очередь сообщений, нужно просто поместить ее между командами BEGIN\_MESSAGE\_MAP() и END\_MESSAGE\_MAP(). Например, если необходимо обработать в программе сообщение WM\_CHAR, то очередь должна выглядеть так:

BEGIN\_MESSAGE\_MAP(CMainWin, CFrameWnd)

ON\_WM\_CHAR()

END\_MESSAGE\_MAP()

Для организации обработки сообщений нужно выполнить следующие действия:

1. В карту сообщений программы должна быть включена команда соответствующего сообщения.
2. Прототип функции-обработчика должен быть включен в описание класса, ответственного за обработку данного сообщения.
3. В программу должна быть включена реализация функции-обработчика.

**Дескрипторы окна**

После создания окна, создающая функция возвращает дескриптор окна (window handle), который уникально идентифицирует окно. Прикладная программа использует этот дескриптор в других функциях, чтобы направить их действия на это окно.

**3) Реализация функций. Реализация пунктов меню**

**Иерархия классов MFC**

Библиотека MFC содержит большую иерархию классов, написанных на С++. В ее вершине находится класс СObject, который содержит различные функции, используемые во время выполнения программы и предназначенные, в частности, для предоставления информации о текущем типе во время выполнения, для диагностики, и для сериализации.

**Основные классы**

Некоторые классы порождаются непосредственно от CObject. Наиболее широко используемыми среди них являются CCmdTarget, CFile, CDC, CGDIObject и CMenu. Класс ССmdTarget предназначен для обработки сообщений. Класс СFile предназначен для работы с файлами. Класс CDC обеспечивает поддержку контекстов устройств. Об контекстах устройств мы будем говорить несколько позднее. В этот класс включены практически все функции графики GDI. CGDIObject является базовым классом для различных DGI-объектов, таких как перья, кисти, шрифты и другие. Класс СMenu предназначен для манипуляций с меню. От класса CCmdTarget порождается очень важный класс CWnd. Он является базовым для создания всех типов окон, включая масштабируемые ("обычные") и диалоговые, а также различные элементы управления. Наиболее широко используемым производным классом является CFrameWnd. Как Вы увидите в дальнейшем, в большинстве программ главное окно создается с помощью именно этого класса.

От класса CCmdTarget, через класс CWinThread, порождается, наверное, единственный из наиболее важных классов, обращение к которому в MFC-программах происходит напрямую: CWinApp. Это один из фундаментальных классов, поскольку предназначен для создания самого приложения. В каждой программе имеется один и только один объект этого класса. Как только он будет создан, приложение начнет выполняться.

**Функции-члены в MFC**

Большинство функций, вызываемых в MFC-программе, являются членами одного из классов, определенных в библиотеке. Большинство функций API доступны через функции-члены MFC. Тем не менее, всегда можно обращаться к функциям API напрямую. Иногда это бывает необходимым, но все же в большинстве случаев удобнее использовать функции-члены MFC.

**Глобальные функции в MFC**

В библиотеке есть ряд глобальных функций. Все они начинаются с префикса Afx. (Когда MFC только разрабатывалась, то проект назывался AFX, Application Framework. После ряда существенных изменений AFX была переработана в MFC, но прежнее название сохранилось во многих идентификаторах библиотеки и в названиях файлов.) Например, очень часто используется функция AfxMessageBox(), отображающая заранее определенное окно сообщения. Но есть и член-функция MessageBox(). Таким образом, часто глобальные функции перекрываются функциями-членами.

**Файл AFXWIN.H**

Все MFC-программы включают заголовочный файл AFXWIN.H. В нем, а также в различных вспомогательных файлах, содержатся описания классов, структур, переменных и других объектов MFC. Он автоматически подключает большинство заголовочных файлов, относящихся к MFC, в том числе и WINDOWS.H, в котором определены все функции Windows API и другие объекты, которые используются при традиционном программировании на С и "чистом" API.

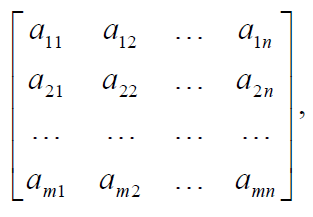
**Каркас MFC-программы**

В простейшем случае программа, написанная с помощью MFC, содержит два класса, порождаемые от классов иерархии библиотеки: класс, предназначенный для создания приложения, и класс, предназначенный для создания окна. Другими словами, для создания минимальной программы необходимо породить один класс от CWinApp, а другой - от CFrameWnd. Эти два класса обязательны для любой программы.

Кроме создания вышеупомянутых классов, в программе также должна быть организована обработка всех сообщений, поступающих от Windows.

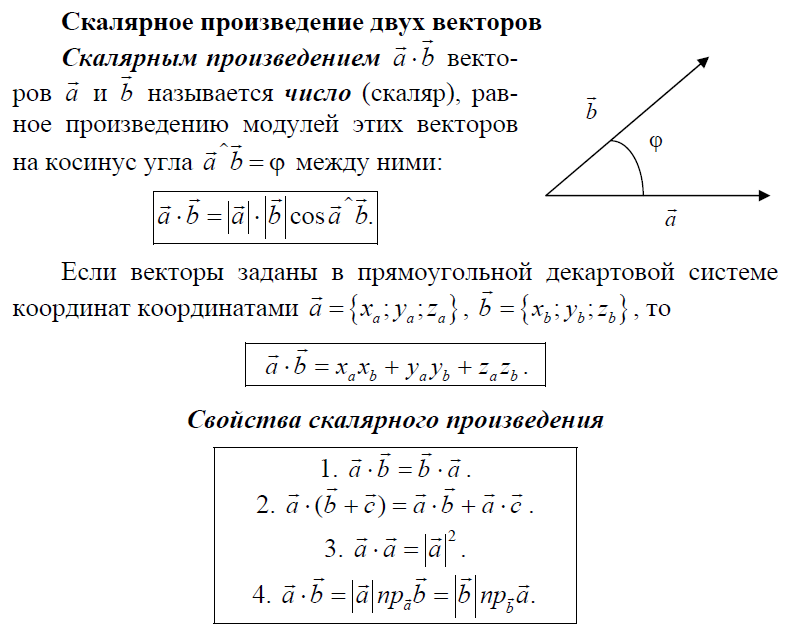
**4) Матрицы и векторы. Операции над ними**

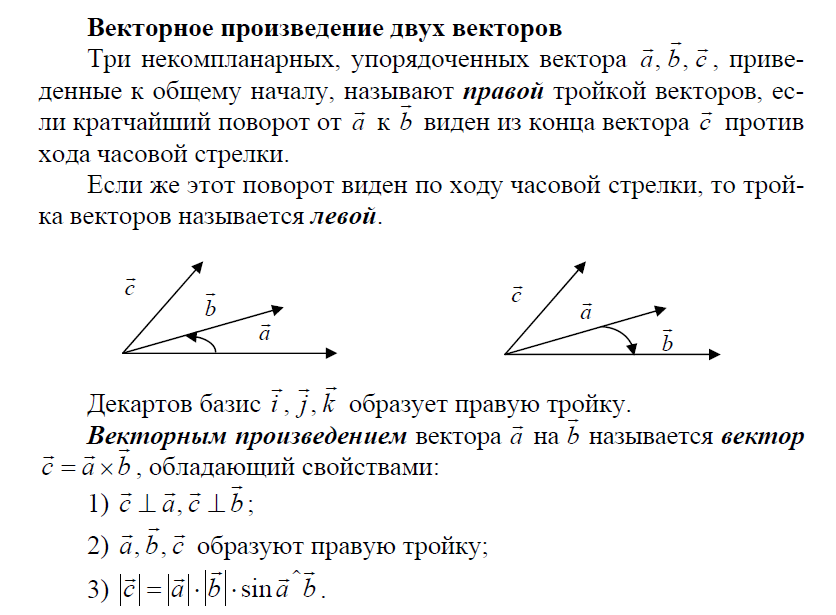
**Матрицей** размера m×n называется прямоугольная таблица чисел (или других математических объектов) – элементов матрицы, расположенных в m строках и n столбцах.



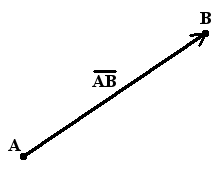
**Вектор** представляет собой массив из одной строки и n столбцов (вектор-строка) или из одного столбца и n строк (вектор-столбец), а матрица — это массив из m строк и n столбцов.

Под вектором на плоскости или в пространстве будем понимать направленный отрезок, который можно переносить параллельно самому себе (свободный вектор).





Длина направленного отрезка определяет числовое значение вектора и называется длиной вектора или **модулем вектора **.

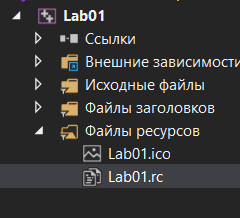


Основное соотношение. Длина вектора |a| в прямоугольных декартовых координатах равна квадратному корню из суммы квадратов его координат.

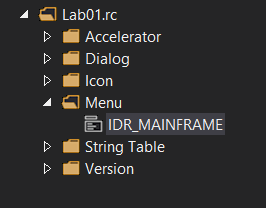
Чтобы найти **косинус угла между векторами** нужно, скалярное произведение этих векторов разделить на произведение их длин.

**5. Создать пункт меню, выести свое имя в окно**

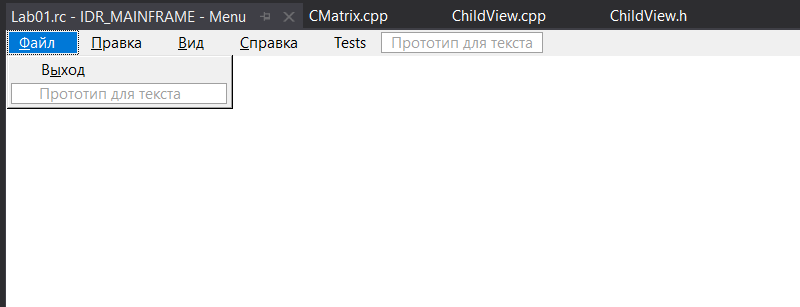
Файлы ресурсов → Lab01.rc:



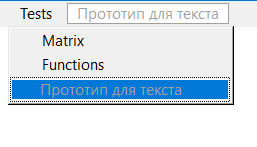
Menu → IDR\_MAINFRAME:



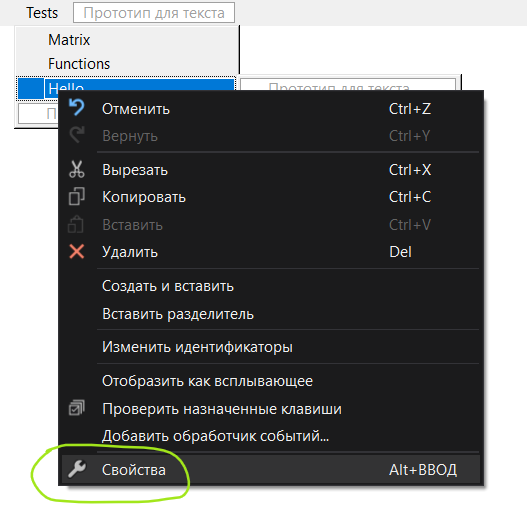
Откроется конструктор меню:



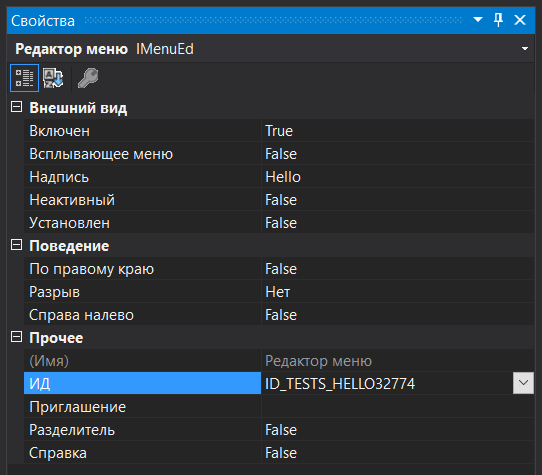
Нажимаем левой кнопкой мыши на Test, в выпадающем меню щелкаем на строку «Прототип для текста» и вводим текст, закончив ввод нажимаем Enter:



Затем в контекстном меню созданной строки открываем свойства:



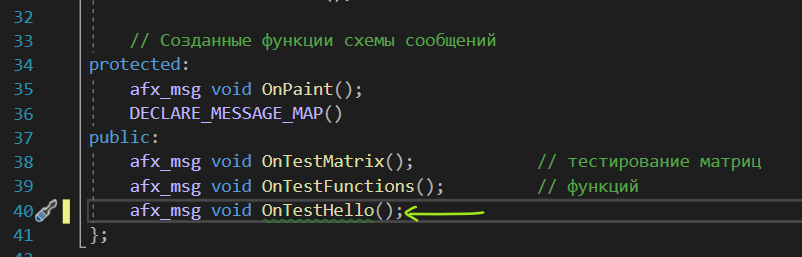
В окне свойств находим ID элемента:



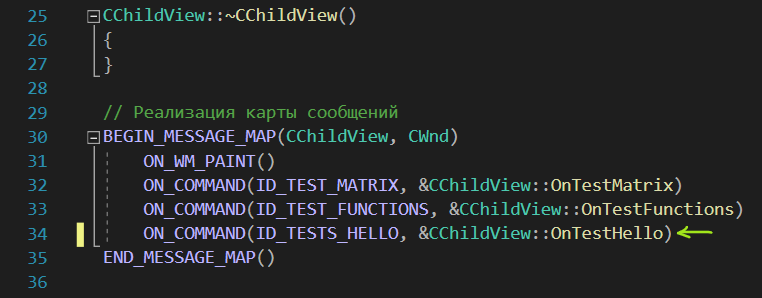
Его можно поменять:



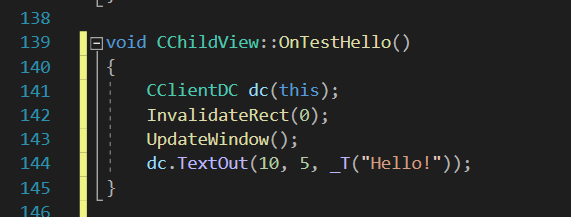
Переходим в файл ChildView.h. Там задаем прототип функции для обработки нажатия на элемент «Hello»:



Переходим в файл ChildView.cpp. Помещаем команду в карту сообщений. Первым параметром передаем ID элемента, вторым – функцию обработки:



В том же файле прописываем реализацию функции. Например, так:



Данная функция выведет «Hello!» в окно.

Результат:

