Лабораторная работа №1.

Тема: Настройка среды разработки, настройка проекта для работы с КОМПАС-3D API.

Введение

Интерфейс

Взаимодействие с системой КОМПАС-3D происходит при помощи интерфейсов. Проще говоря, интерфейс – это какой-либо класс, позволяющий одной программе взаимодействовать с другой.

Каждый из интерфейсов представляет собой класс, имеющий свои свойства и методы. Данные классы реализованы системой КОМПАС. Мы будем лишь обращаться к свойствам и методам данных классов.

Документация

Все интерфейсы (классы), которые предоставляет система КОМПАС, описаны в документации данной системы. Наибольший интерес представляет SDK – это справочный файл, в котором содержится описание интерфейсов. На моем ПК он находится: C:\Program Files\ASCON\KOMPAS-3D v22 Study\SDK.

Модули

В состав КОМПАС входит большое число модулей (заголовочных файлов) для различных языков программирования. В этих модулях содержится описание интерфейсов в соответствие с синтаксисом определенного языка. Располагаются эти модули: C:\Program Files\ASCON\KOMPAS-3D v22 Study\SDK\Include.

Среди всех модулей нас интересует лишь несколько:

- **KAPITypes** – библиотека для работы с различными типами в КОМПАС;
- **Kompas6API5 –** библиотека для работы с API-интерфейсами версии 5;
- **Kompas6Constants –** библиотека для работы с константами;
- **Kompas6Constants3D –** библиотека для работы с 3D константами.

1. Установка среды разработки

Скачиваем с официального сайта Visual Studio и устанавливаем. Скачать можно по ссылке https://visualstudio.microsoft.com/ru/vs/community/. Также нужно скачать и установить САПР-систему КОМПАС-3D с официального сайта по ссылке https://kompas.ru/kompas-educational/about/.

2. Подготовка и настройка проекта

Создаем проект с использованием шаблона ***Библиотека классов(.NET Framework),*** для удобства поиска шаблона можно выставить в параметрах поиска:

1) *Язык программирования – C#.*
2) *Платформа – Windows.*
3) *Тип проекта – Библиотека.*

Нужный нам шаблон показан на рисунке 1.

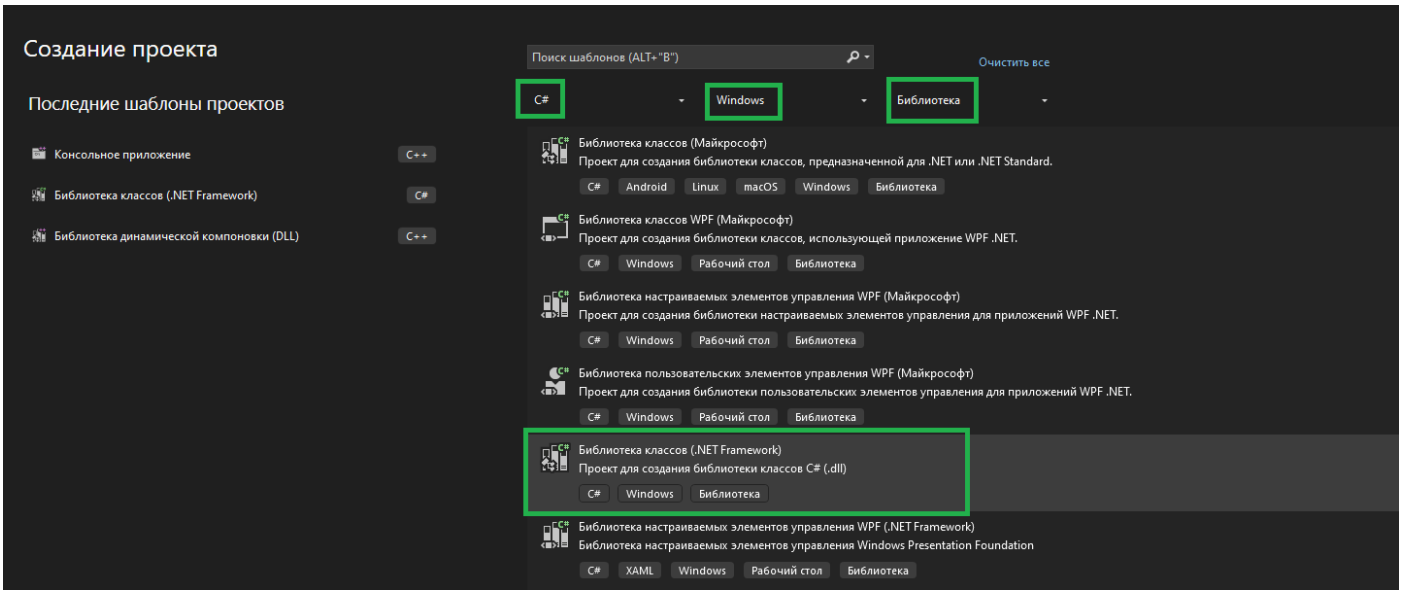

Рисунок 1 – Библиотека классов (.NET Framework)

Далее выбираем расположение нашего проекта. Важно чтобы путь и название проекта были на латинице. Также можно поставить галочку ***Поместить решение и проект в одном каталоге,*** но я не использую. (рисунок 2).

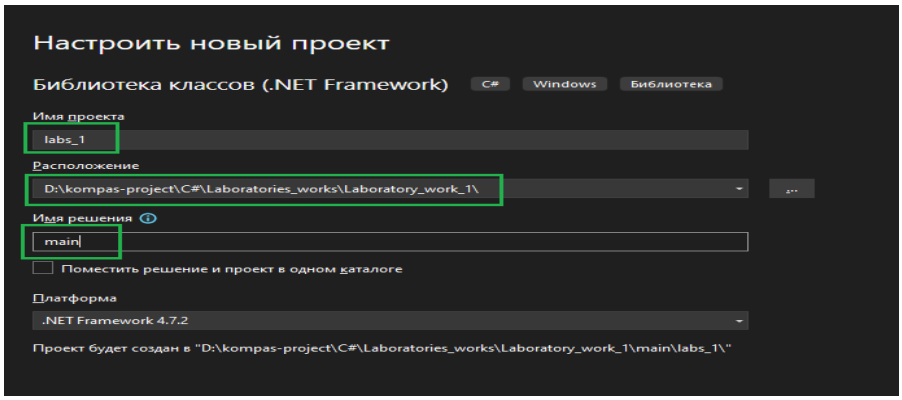

Рисунок 2 – Создание проекта.

После создания проекта нужно подключить нужные библиотеки:

- KAPITypes.dll
- Kompas6API5.dll
- Kompas6Constants.dll
- Kompas6Constants3D.dll
- KompasAPI7.dll

Подключаем данные библиотеки как показано на рисунке 3.

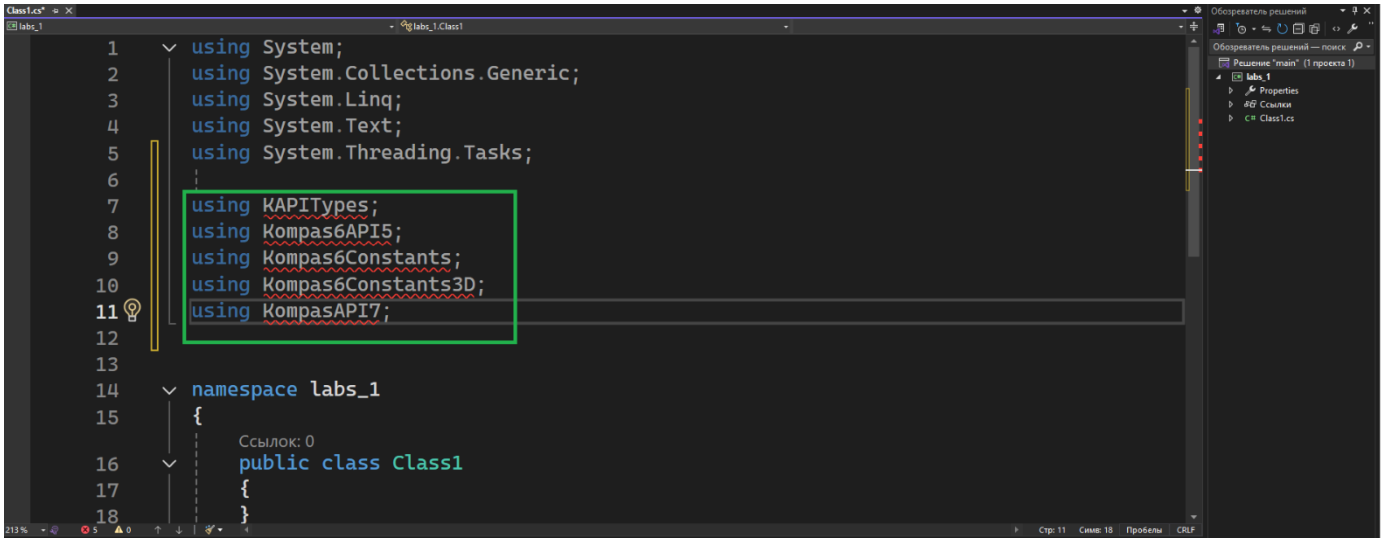

Рисунок 3 – Подключение библиотек.

Как видно из рисунка выше среда разработки подчеркивает данные библиотеки, что говорит об какой-то ошибке. Всё дело в том, что в наборе стандартных библиотек Visual Studio нет данных библиотек. Для того чтобы наш компилятор во время препроцесса смог увидите данные библиотеки нужно их подключить. Подключать их можно напрямую из каталога Компаса, но при таком подходе есть большой минус. Предположим, что вы работаете в команде и взаимодействуете с одним и тем же проектом, при этом у каждого из команды Компас может быть установлен в различных расположениях, что приведет к невозможности компиляции у других коллег. Чтобы этого избежать, можно перенести данные библиотеки в папку вашего проекта. Данные библиотеки лежат в Папке Ascon. Например, как у меня C:\Program Files\ASCON\KOMPAS-3D v22 Study\SDK\Include (рисунок 4).

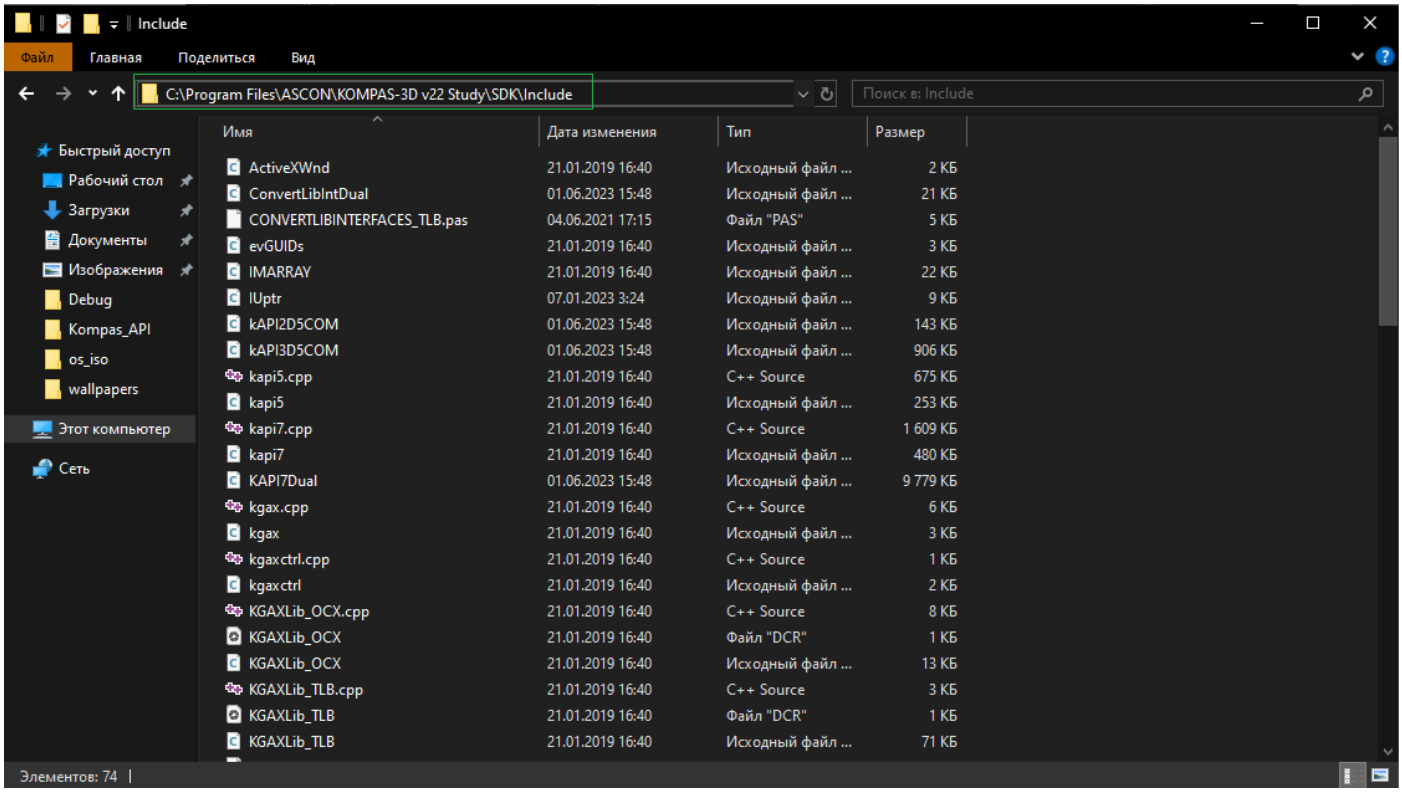

Рисунок 4 – Расположение библиотек предоставляемых КОМПАС

В вашем случае может отличаться: *буква физического диска, версия самого КОМПАС*. Из данной папки копируем нужные библиотеки в папку нашего проект, заранее в папке проекта можно создать папку например, ***include*** (рисунок 5).

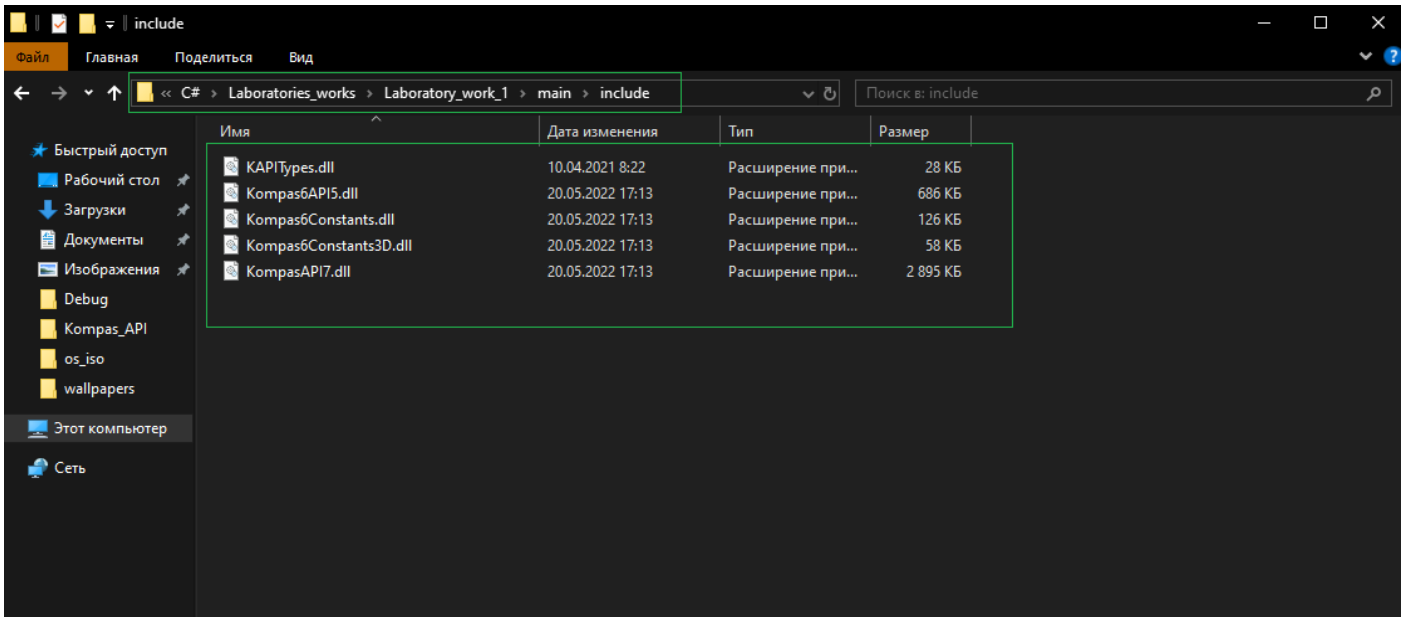

Рисунок 5 – Скопированные библиотеки в нужный проект

Далее нужно добавить ссылки на данные библиотеки в нашем проекте. Нажимаем на ссылки в дереве проекта и нажимаем добавить (рисунки 6-7).

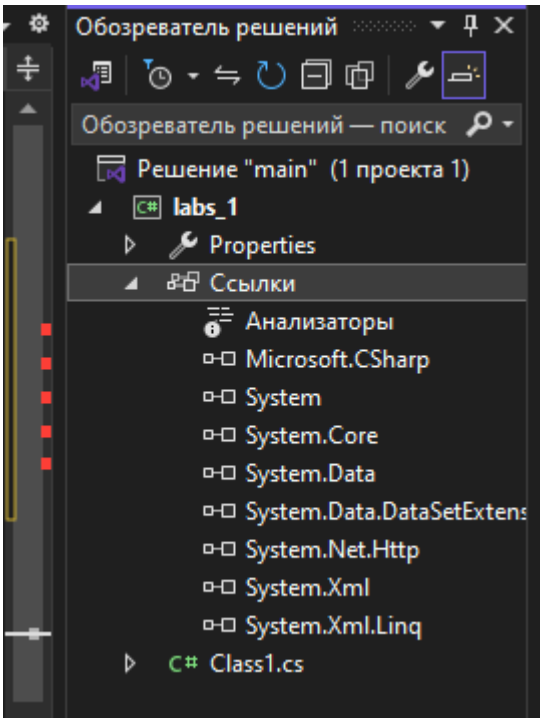

Рисунок 6 – Ссылки в обозревателе решений

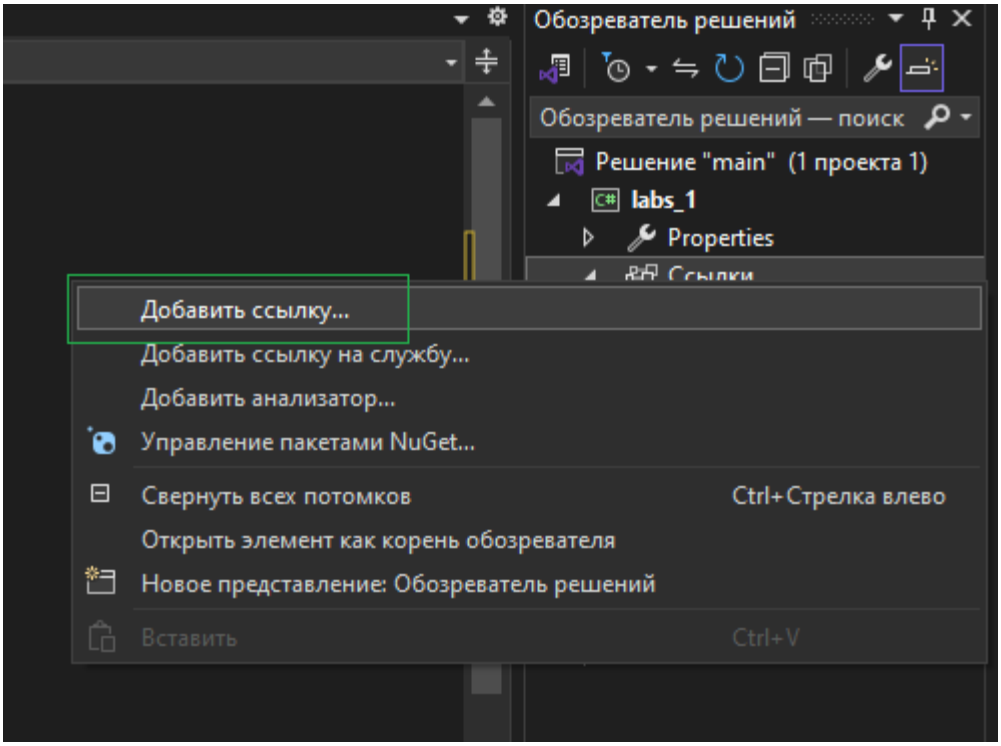

Рисунок 7 – Добавить ссылку

Далее нажимаем обзор, переходим в папку include в папке нашего проекта и нажимаем добавить (рисунок 8).

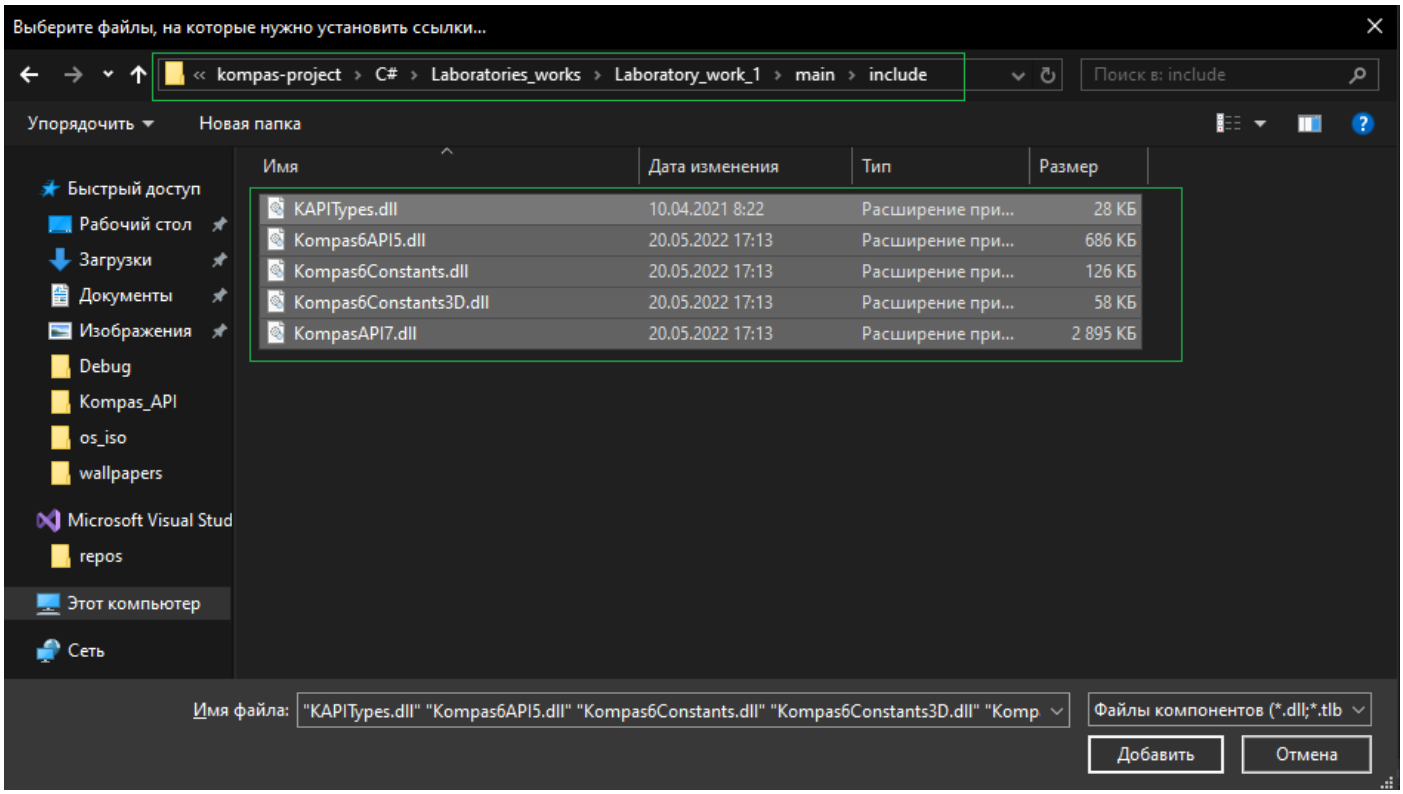

Рисунок 8 – Выбор нужных библиотек.

После данных действий компилятор видит данные библиотеки.

3. Подключение к КОМПАСу

Для того чтобы работать с данной САПР нужно как-то к ней подключиться. Подключаться можно уже к запущенному экземпляру КОМПАС при помощи функции ***GetActiveObject(),*** если в момент вызова данной функции не будет запущенно ни одного экземпляра КОМПАС, функция генерирует исключение с классом ***ESysError***. Код подключения к уже запущенному экземпляру приложения КОМПАС выглядит так:

```
KompasObject kompas = (KompasObject)Marshal.GetActiveObject(
    "Kompas.Application.5");
kompas.Visible = true;
```

KompasObject – является ключевым интерфейсом системы, из которого могут быть получены остальные интерфейсы.

Marshal – это класс предоставляющий методы для выделения неуправляемой памяти, копирования блоков неуправляемой памяти и преобразования управляемых типов в неуправляемые, а также прочих разнообразных методов, используемых при взаимодействии с неуправляемым кодом.

GetActiveObject() – получает выполняющийся экземпляр указанного объекта из таблицы запущенных объектов (ROT). При помощи данного метода мы и подключаемся к запущенному экземпляру программы Компаса.

Таблицы запущенных объектов (ROT) – это таблицы, в которых объекты могут регистрировать себя. Данные таблицы позволяют проверять запущен ли объект.

*"**Kompas.Application.5**"* – является записью приложения КОМПАС из ***таблицы ROT.***

При помощи такого способа чаще всего и запускаю ***ddl*** написанные при помощи ***API КОМПАС.*** Так ка чаще всего данные библиотеки запускаются уже из запущенного экземпляра программы.

Но существуют и другие задачи решаемые при помощи ***API-интерфейсов.*** Например, написание программы для построения детали *"Молочная гайка"*. Данная деталь является типовой в узконаправленной отрасли, такое как молочное производство. Следовательно, в общем смысле гайка является типовой деталью, но не молочная гайка. В таких случаях чаще всего приложение с интерфейсом для построения данной детали с заданными параметрами под различные стандарты. Чтобы не нагружать основной экземпляр программы запускается данное приложение, которое запускает новый экземпляр приложения, в котором строится данная деталь и сохраняется куда нужно пользователю и закрывается, главное качество такого подхода построение детали происходит в фоновом режиме.

Для такого подхода нужно запустить новый экземпляр приложения КОМПАС, сделать это можно при помощи:

```
Process.Start("kStudy");
```

Process – класс, который предоставляет доступ к локальным и удаленным процессам и позволяет запускать и останавливать локальные системные процессы.

Start() – это функция класса ***Start,*** которая запускает процессы в системе.

"kStudy" – название .exe файлы, который запускает студенческую версию КОМПАСа.

Также, чтобы экземпляр запущенный в фоновом режиме не нагружал систему, его нужно закрыть (завершить процесс). Сделать это можно при помощи:

```
kompas.Quit();
```

Функция интерфейса ***KompasObject*** которая закрывает экземпляр КОМПАСа и завершает процессы, связанные с ним.

Для примера напишем с вами программу, которая должна выводить в окне Windows сообщение с информацией о версии КОМПАСа, программа выглядит так:

```
int Major, Minor, Release, Build;
string str;

kompas = (KompasObject)Marshal.GetActiveObject
    ("Kompas.Application.5");
kompas.Visible = true;
```

```
kompas.ksGetSystemVersion(out Major, out Minor,
    out Release, out Build);
str = Major.ToString() + "." + Minor.ToString() + " Release:"
    + Release.ToString() + " Build: " + Build.ToString();
MessageBox.Show(str);
```

Функция **ksGetSystemVersion(int Major, int Minor, int Release, int Build) –** данная функция передает в переменные информацию о версии компаса, переменные:

- **Major** – в данную переменную записывается старшее слово версии.
- **Minor** – в данную переменную записывается младшее слово версии.
- **Release** – в данную переменную записывается номер выпуска.
- **Build** – в данную переменную записывается номер сборки.
- **Str** – данная переменная нужна для вывода информации о версии компаса.

Функция **Show()** – является функцией класса **MessageBox,** данная функция передает значение в окно сообщения Windows и выводит его.

Задание 1.

Напишите такую же программу, **которая будет запускать экземпляр КОМПАСа, а не сразу подключаться к запущенному экземпляру**, также нужно сделать вывод информации о версии КОМПАСа **не используя** дополнительную переменную **str. Важно:** после того как вы запустите новый экземпляр КОМПАСа, вам всё равно придется подключаться к его активному интерфейсу.

Проблема, с которой вы вероятнее всего столкнетесь:

При запуске вашей программы будет запускаться новый экземпляр КОМПАСа, и после этого среда разработки будет выдавать ошибку: **System.Runtime. InteropServices.COMException:"Операция недоступна (Исключение из HRESULT: 0x800401E3 (MK_E_UNAVAILABLE))"**

Дело в том, что программа КОМПАС запускается не мгновенно, следовательно после того как программа подает команду запуска экземпляра КОМПАСа, она следом (**практически моментально**) пытается подключиться к активному интерфейсу данного экземпляра через имя в таблице **ROT**. При этом экземпляр КОМПАСа ещё не успевает прописать имя своего процесса в данную таблицу или же КОМПАС в принципе ещё не является активным.

Чтобы избежать данной ошибки нужно между запуском экземпляра и подключения к его интерфейсу выждать какое-то время. *Для этого можно использовать функцию **Sleep()** класса **Thread**, который находится в пространстве имен **System.Threading.***

4. Создание чертежа. Параметры документа

Интерфейс **ksDocumentParam** описывает параметры графического документа. Получить данный интерфейс можно с помощью метода **GetParamStruct** нашего главного интерфейса **KompasObject** (который мы неоднократно будем использовать) для этого в качестве аргумента нужно передать значение **ko_DocumentParam**.

Свойств и методов у данного интерфейса не много.

Свойства:

- **author** – строка с именем автора документа.
- **comment** – комментарий к документу.
- **fileName** – строка с именем файла документа.
- **regime** – режим редактирования. Если данное свойство равное единице, то редактируемый документ не отображается на экране. Если же оно равно нулю, то документ отображается на экране пользователя.
- **type** - тип документа.

В таблице представленной ниже показаны основные типы документов системы КОМПАС:

Таблица 1. Основные типы документов

Имя типа	Описание
ksDocumentUnknown	Неизвестный тип
ksDocumentDrawing	Чертеж
ksDocumentFragment	Фрагмент
ksDocumentSpecification	Спецификация
ksDocumentPart	Деталь
ksDocumentAssembly	Сборка
ksDocumentTextual	Текстовый документ

Из методов данного интерфейса мы разберем лишь один – метод **Init().** Данный метод сбрасывает все настройки чертежа. Он не имеет входных параметров и возвращает **true** в случае успешного завершения или **false**, если произошла какая-то ошибка.

Создание нового чертежа

Создать новый чертеж можно при помощи:

```
ksDocument2D doc_2d;
  ksDocumentParam doc_param;

  doc_param = (ksDocumentParam)kompas.GetParamStruct
((short)Kompas6Constants.StructType2DEnum.ko_DocumentParam);
  doc_param.Init();
  doc_param.type = (short)DocumentTypeEnum.ksDocumentDrawing;
  doc_2d = (ksDocument2D)kompas.Document2D();
  doc_2d.ksCreateDocument(doc_param);
```

Объявляем 2 переменные, которые содержат в себе интерфейсы: **ksDocument2D**, **ksDocumentParam.** Заполняем интерфейс параметра документа, очищаем параметры. Затем присваиваем документу тип **ksDocumentDrawing**, следовательно это означает, что документом будет являться чертеж. Далее получаем интерфейс данного документа и создаем его.

Открытие документа

Часто встречаются задачи, где нужно открыть како-то шаблон чертежа, заполнить его и закрыть. Для того, чтобы открыть документ используется метод **ksOpenDocument**, который предусмотрен интерфейсом **ksDocument2D**. Пример кода для открытия документа:

```
ksDocument2D doc_2d1 = (ksDocument2D)kompas.Document2D();
doc_2d1.ksOpenDocument("D:/kompas-project/Kompas
project/test_doc.cdw", false);
kompas.Visible = true;
```

ksOpenDocument принимает два параметра:

- **string docName** (путь и название нужного документа)

- **bool** regim (если true, то файл открывается в невидимом режиме, иначе в видимом режиме)

Закрытие и сохранение документа

Из своей программы вы можете также закрыть документ. Для этого используется метод **ksCloseDocument** все тот же интерфейса **ksDocument2D**. Данный метод не имеет входных параметров. Этот метод закрывает документа, не сохраняя его и не предлагая сохранить его. Поэтому важно перед закрытием сохранить документ, а только потом закрывать его

Для сохранения чертежа предусмотрено два метода **ksSaveDocument** и **ksSaveDocumentEx** интерфейса **Document2D**.

Метод **ksSaveDocument()** имеет всего лишь один параметр – это строка с именем файла в который следует сохранить данный чертеж.

Метод **ksSaveDocument()** – может сохранять документ в предыдущих версиях Компаса, но это не особо важный метод.

Задание 2.

Написать программу, которая будет в невидимом режиме создавать чертеж, заполняя: имя автора, название чертежа, комментарий, а также сохранять его. А также напишите вторую программу, которая будет открывать документ созданный вами раннее, сохранять его под другим именем и закрывать.