**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**“Московский политехнический университет”**

**КАФЕДРА ИНФОКОГНИТИВНЫХ ТЕХНОЛОГИЙ**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

**ПРОГРАММИРОВАНИЕ В САПР**

Задание № 7

**ПОДСИСТЕМА 3D-МОДЕЛИРОВАНИЯ ПЛАНЕТАРНО-ШНЕКОВОГО СМЕСИТЕЛЯ**

Студент: Беседин А.В.

Группа: 171-333

Преподаватель: к.т.н., доцент Лянг В.Ф.

Москва - 2018

**Задание**

1. Разработать подсистему на языке C# под Autodesk Inventor по параметрическому 3D-моделированию смесителя (рис. 1), состоящего из конического корпуса, плоской крышки, шнека, корпуса устройства для выгрузки материала, боковых опор (тип 2 в приложении 2) и патрубков (А.С. Тимонин. Основы конструирования и расчета химико-технологического оборудования и природоохранного оборудования. Справочник в 2 т. Т 2. – Калуга: Издательство Н. Бочкаревой, 2002 – с. 202).
2. Разместить в диалоговом окне приложения поясняющие рисунки, надписи и поля ввода исходных данных. Размеры деталей должны задаваться в мм.
3. Организовать в программе проверку по активному состоянию Inventor.
4. Предусмотреть в подсистеме сообщения о неправильно введенных данных.
5. Провести сборку аппарата методом сопряжения поверхностей сборочных единиц.



Рис. 1. Общий вид планетарно-шнекового смесителя

**Содержание**

[Введение 4](#_Toc514698061)

[1. Интерфейс пользователя 5](#_Toc514698062)

[1.1. Окно запуска Inventor 5](#_Toc514698063)

[1.2. Окна построения деталей 6](#_Toc514698064)

[1.3. Кнопка сборки 9](#_Toc514698065)

[2. Создание 3D-модели планетарно-шнекового смесителя 10](#_Toc514698066)

[2.1. Создание конического корпуса 10](#_Toc514698067)

[2.2. Создание крышки 10](#_Toc514698068)

[2.3. Создание шнека 11](#_Toc514698069)

[2.4. Создание корпуса для выгрузки материалов 11](#_Toc514698070)

[2.5. Создание боковых опор 12](#_Toc514698071)

[2.6. Выполнение сборки из построенных деталей 12](#_Toc514698072)

[3. Дополнительный функционал программы 14](#_Toc514698073)

[3.1. Сохранение деталей 14](#_Toc514698074)

[3.2. Вывод сообщений о различных событиях 14](#_Toc514698075)

[3.2.1. Сообщения об успешности 14](#_Toc514698076)

[3.2.2. Сообщение-указание 15](#_Toc514698077)

[3.2.3. Сообщения о проблемах 15](#_Toc514698078)

[Заключение 16](#_Toc514698079)

[Список используемой литературы 17](#_Toc514698080)

[Приложение 1 18](#_Toc514698081)

# **Введение**

**Цель**

Целью проекта является разработка подсистемы параметрического 3D-моделирования планетарно-шнекового смесителя на языке высокого уровня С# в среде программирования Microsoft Visual Studio под Autodesk Inventor.

**Поставленные задачи**

1. Разработка интерфейса программного продукта, состоящего из чертежей деталей с настраиваемыми размерам и необходимых элементов управления.
2. Программное построение деталей сборки: корпуса, плоской крышки, шнека, корпуса устройства для выгрузки материала и боковых опор.
3. Выполнение сборки из построенных деталей.

# **1. Интерфейс пользователя**

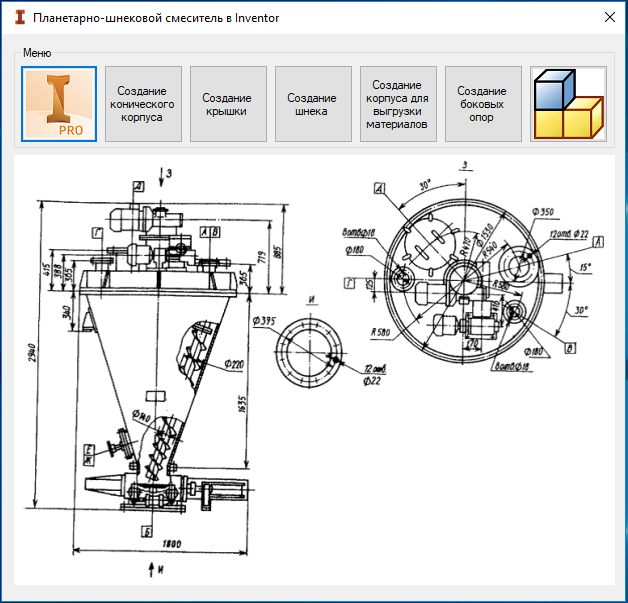


рис. 1.1. Основное окно

## **1.1. Окно запуска Inventor**

Запуск программы Inventor реализован в отдельном окне, которое появляется либо при вызове с основного окна программы, либо при попытке взаимодействовать с Inventor. На окне присутствует выпадающий список со доступными версиями и кнопка непосредственного запуска. Так же присутствует надпись, показывающая запущен ли Inventor. Если Inventor запущен, все элементы управления заблокированы, не позволяя пользователю запустить лишний процесс Inventor.

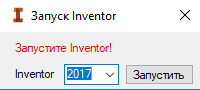


Рис. 1.2. Окно запуска Inventor

## **1.2. Окна построения деталей**

На данных окнах пользователь может увидеть все необходимые чертежи для построения деталей, может задать необходимые ему размеры в мм, запустить непосредственное построение деталей, а также их сохранение. Существуют окна создания: конического корпуса (рис. 1.3), крышки (рис. 1.4), шнека (рис. 1.5), корпуса для выгрузки материалов (рис. 1.6), боковых опор (рис. 1.7).

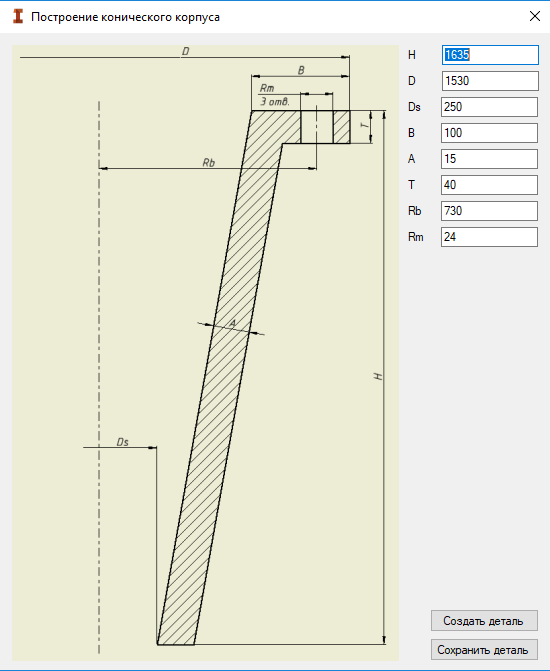


рис. 1.3. Окно построения конического корпуса

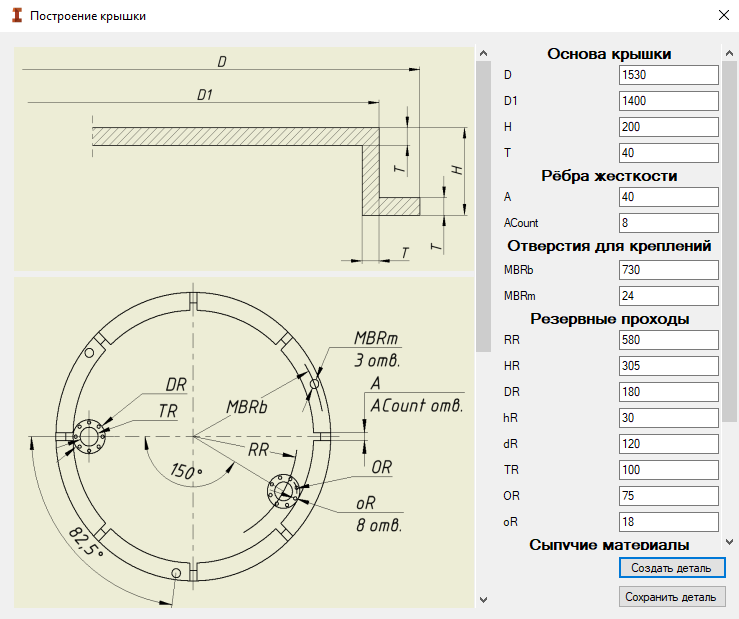


рис. 1.4. Окно построения крышки

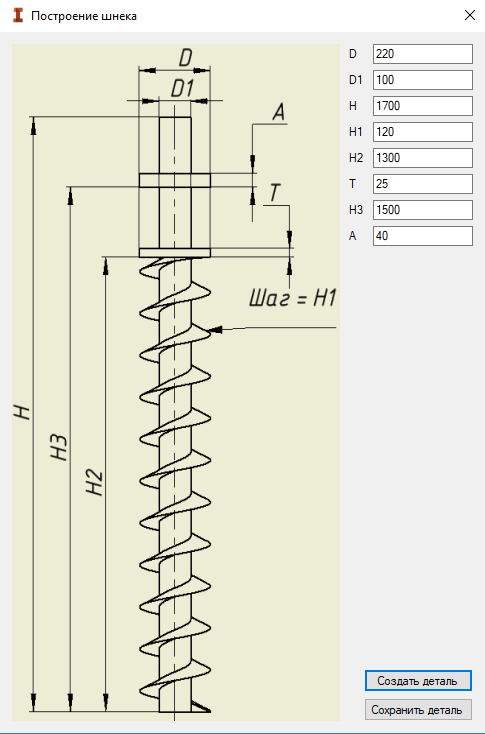


рис. 1.5. Окно построения шнека

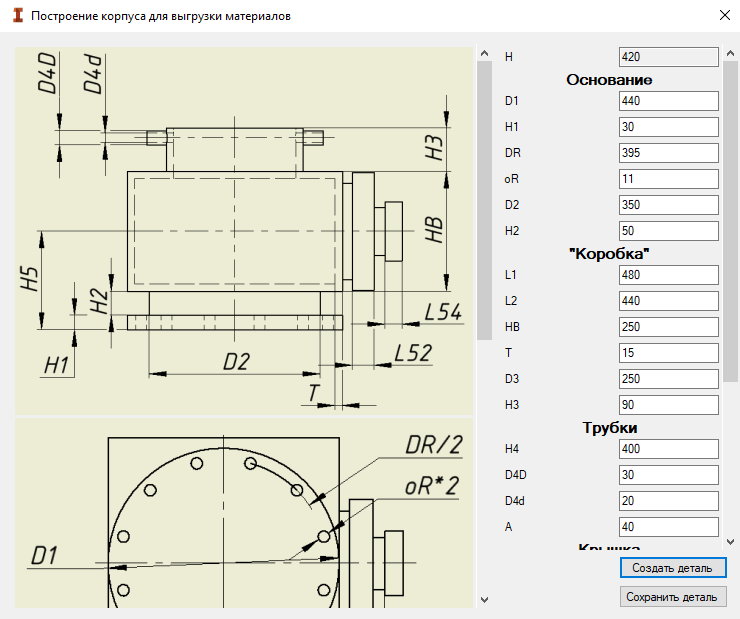


рис. 1.6. Окно построения корпуса для выгрузки материалов

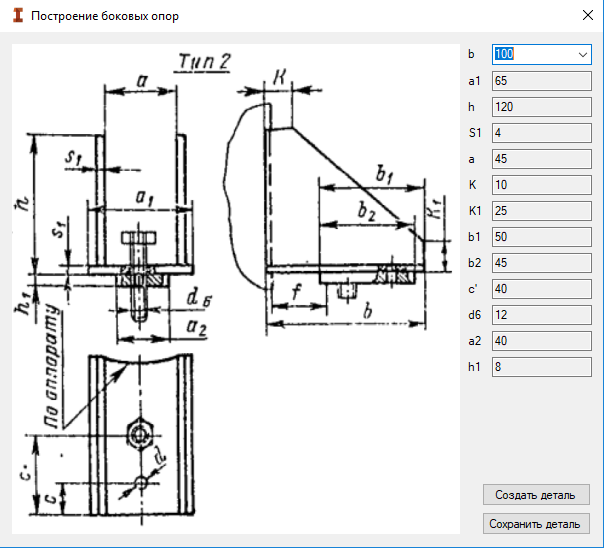


рис. 1.7. Окно построения боковых опор

## **1.3. Кнопка сборки**

Создание сборки лишено отдельного диалогового окна. При нажатии на кнопку сборки пользователю предлагается только открытие уже созданных деталей (рис. 1.8), после чего программа сама производит сборку.

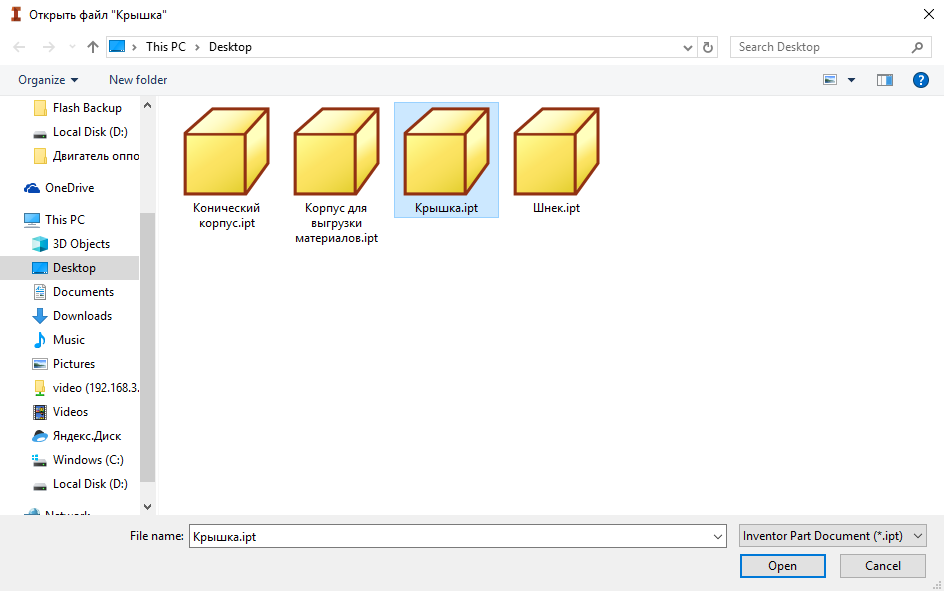


рис. 1.8. Открытие готовых деталей для дальнейшей сборки

# **2. Создание 3D-модели планетарно-шнекового смесителя**

Для создания планетарно-шнекового смесителя пользователю необходимо при помощи данной программы создать все необходимые детали и, при помощи этой же программы, произвести непосредственную сборку.

## **2.1. Создание конического корпуса**

Для построения конического корпуса создается эскиз, на котором располагаются точки, согласно чертежу, после чего происходит соединение точек, пока не образуется замкнутый контур. Далее выполняется функция вращения, после чего создаются отверстия при помощи следующего выдавленного эскиза и использования кругового массива.

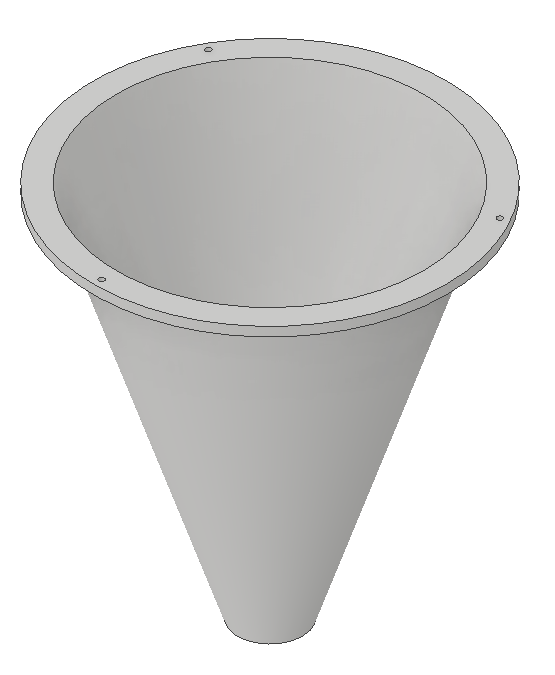


рис. 2.1. Пример 3D-модели конического корпуса

## **2.2. Создание крышки**

Для построения крышки создается эскиз на основе чертежа, далее замкнутый контур вращается, образуя основную часть крышки. После чего создается эскиз ребра жесткости, выдавливается и множится при помощи кругового массива. Далее создается рабочая плоскость на определенной высоте от верхней части крышки и на ней создаются эскизы окружностей, которые выдавливаются в люк и резервные проходы.

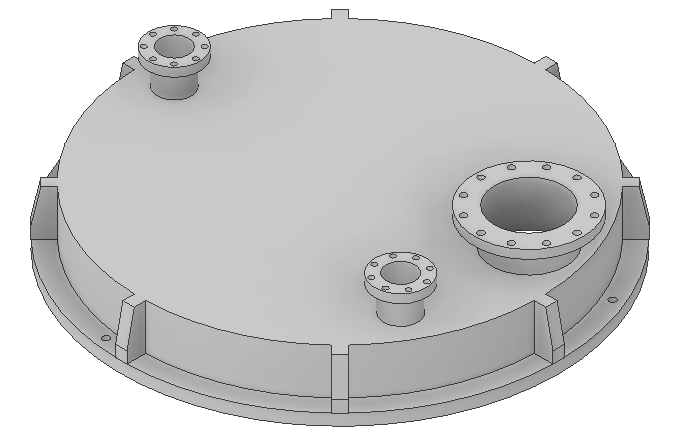


рис. 2.2. Пример 3D-модели крышки

## **2.3. Создание шнека**

Для создания данной детали происходит вращение основного эскиза, согласно чертежу. После чего создается эскиз треугольника, который при помощи инструмента «Пружина» создает основную часть шнека. Далее создается крепление шнека, что позволяет детали обрести максимально правдоподобный вид.

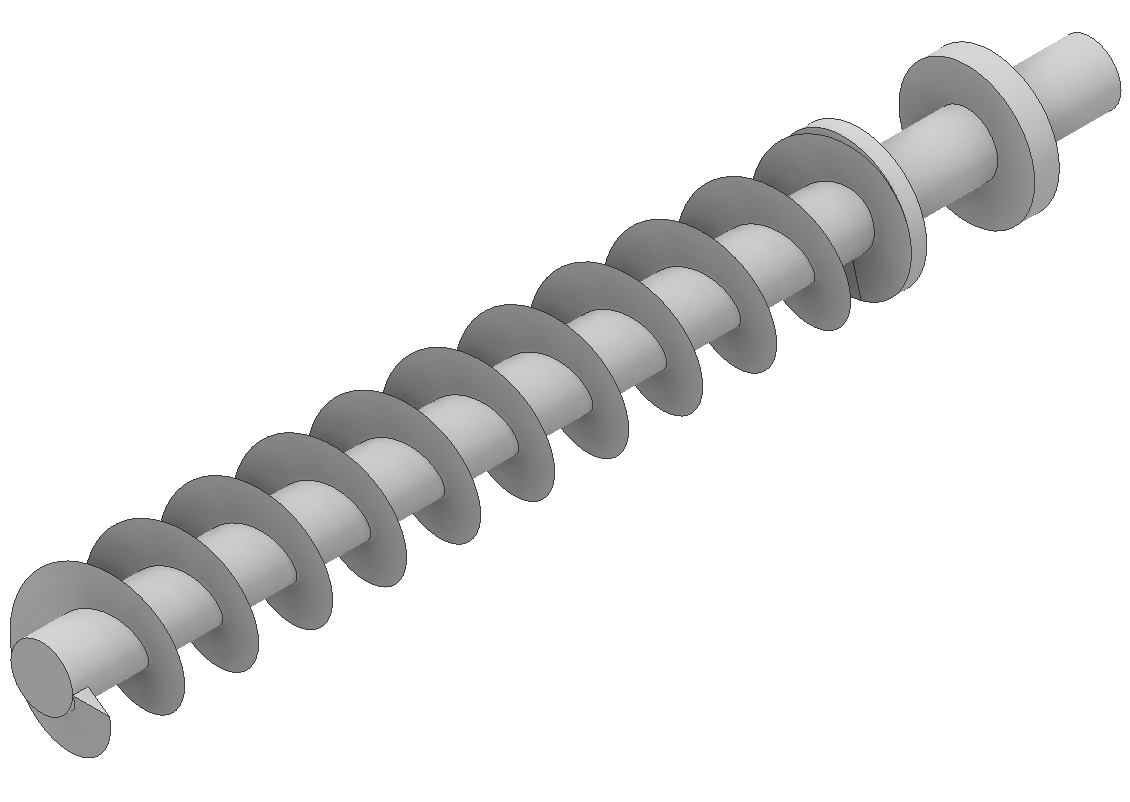


рис. 2.3. Пример 3D-модели шнека

## **2.4. Создание корпуса для выгрузки материалов**

Для создания данной детали создается эскиз окружности, которая выдавливается тем самым образуя нижнее основание, далее поверх полученной модели создается ещё один эскиз, который так же выдавливается, образуя тем самым ножку. После чего при помощи нескольких выдавливании эскизов прямоугольников создается «коробка», а поверх неё создается цилиндрическая часть, которая на сборке будет переходить в конический корпус. Далее на ней создаются трубки, после чего сбоку получившейся «коробки» создается крышечка. У основания создаются отверстия.

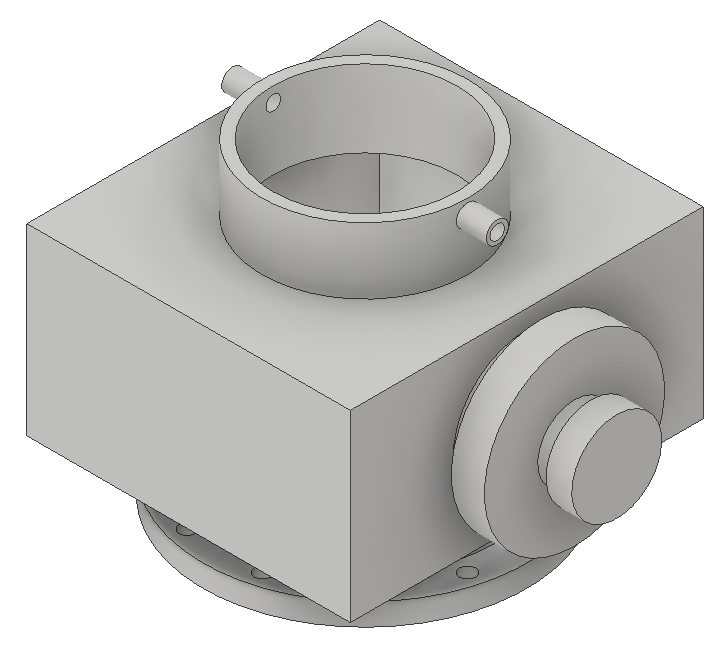


рис. 2.4. Пример 3D-модели корпуса для выгрузки материалов

## **2.5. Создание боковых опор**

Программа создает необходимые эскизы и выдавливает их. Всё это происходит в соответствии с чертежами ГОСТ и размерами, выбранными пользователем (программа не позволяет выбрать размер не из набора ГОСТ).

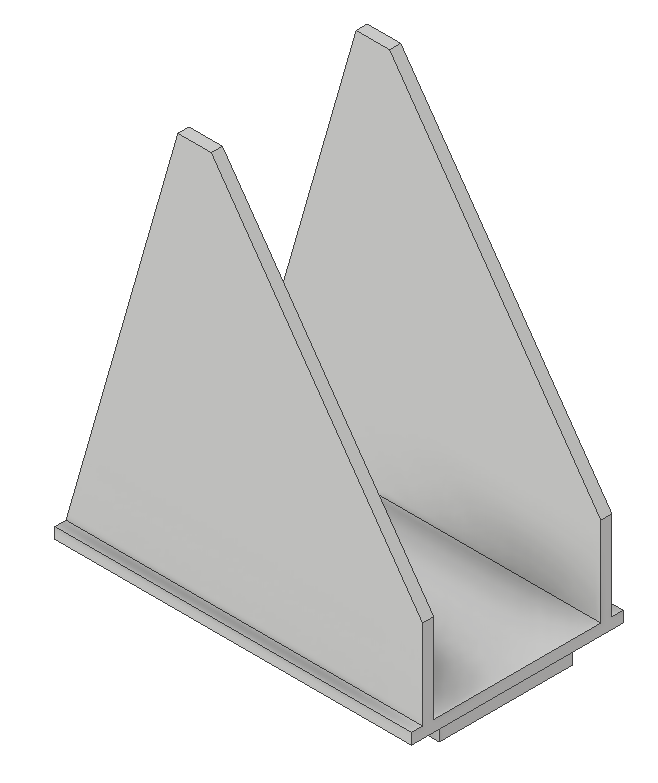


рис. 2.5. Пример 3D-модели боковой опоры

## **2.6. Выполнение сборки из построенных деталей**

После построения и сохранения полученных деталей, можно использовать кнопку сборки, которая откроет все детали снова, после чего соберет. Сначала выполнится присоединение крышки к коническому корпусу путем совмещения фундаментальных отверстий и соответствующих поверхностей. Далее программа выровняет конический корпус и шнек по заранее выбранной базовой оси и базовой плоскости после чего задаст угол между другими базовыми плоскостями обеих деталей, чтобы шнек располагался так, как это предусмотрено сборочным чертежом. Далее корпус для выгрузки материалов и конический корпус совмещаются по базовой оси и соответствующим плоскостям. После данных манипуляций обе детали окончательно выравниваются по базовой плоскости для жесткой заделки. После чего программа поместит боковую опору на коническом корпусе и при помощи функции кругового массива размножит их до 4-х штук.

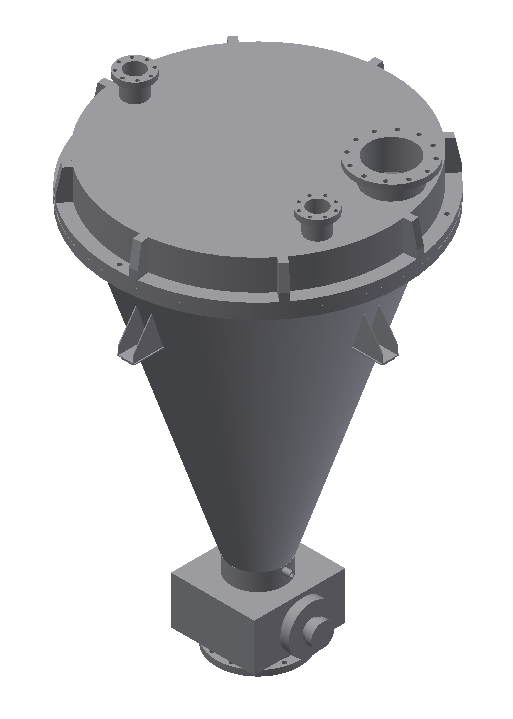


рис. 2.6. Пример собранной 3D-модели

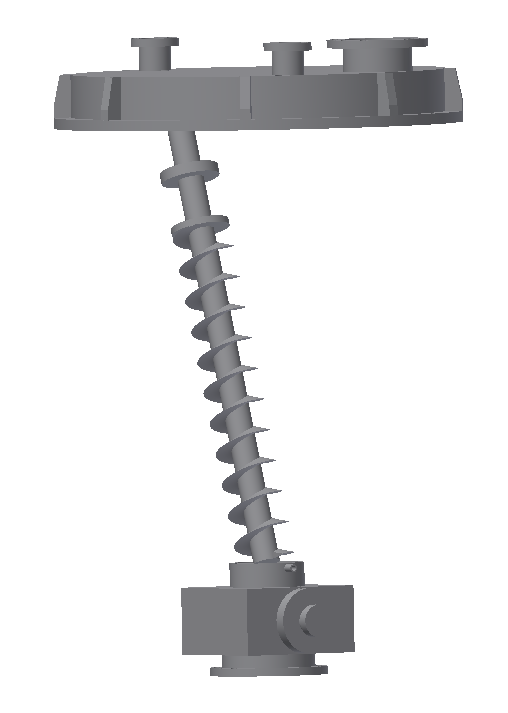


рис. 2.7. Пример собранной 3D-модели со скрытым коническим корпусом

# **3. Дополнительный функционал программы**

## **3.1. Сохранение деталей**

Программа позволяет сохранять созданные детали для дальнейшего использования.

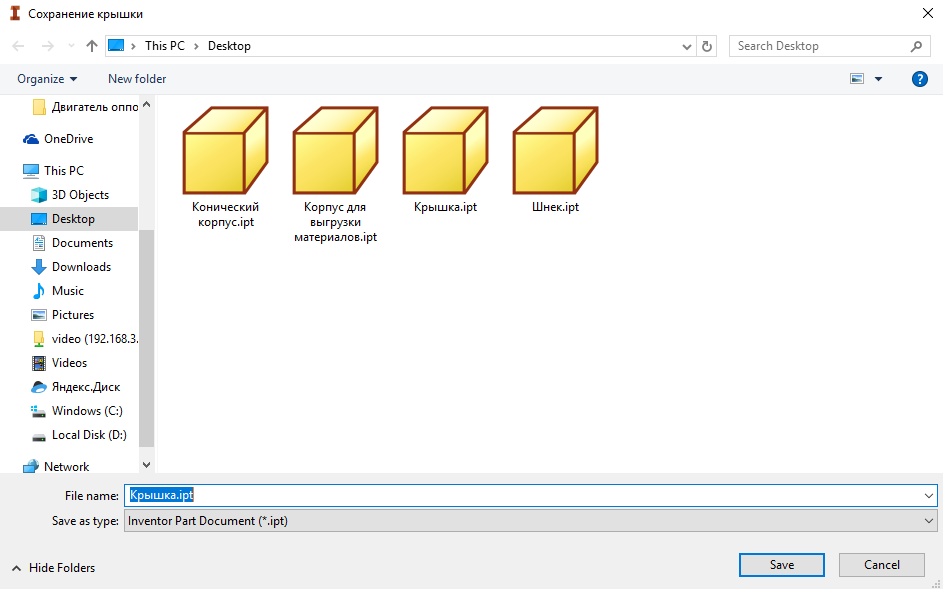


рис. 3.1. Пример окна сохранения

## **3.2. Вывод сообщений о различных событиях**

В определенных случаях программа сообщает, если что-то произошло. Или, наоборот, не произошло.

### **3.2.1. Сообщения об успешности**

Когда программа завершает построение детали (рис. 3.2) или сборку (рис. 3.3), она показывает пользователю соответствующее сообщение.

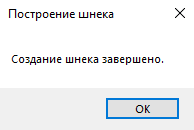


рис. 3.2. Пример сообщения об успешном построении детали

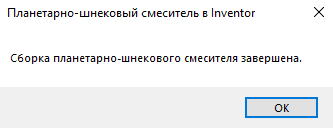


рис. 3.3. Пример сообщения об успешной сборке

### **3.2.2. Сообщение-указание**

В случае, если пользователь вводит значение размера меньше нуля, то программа выводит сообщение.

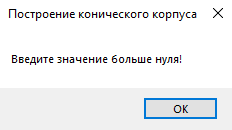


рис. 3.4. Пример сообщения о неверном размере

### **3.2.3. Сообщения о проблемах**

В случае, если что-то идёт не так, как этого хотел бы пользователь, то программа непременно сообщает об этом. Например, если программа не смогла сохранить деталь (рис. 3.5) или собрать установку (рис. 3.6)

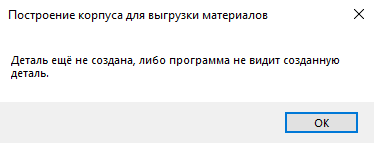


рис. 3.5. Пример сообщения о неуспешном сохранении детали

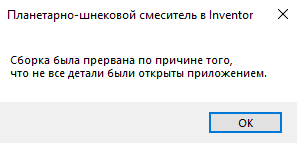


рис. 3.6. Пример сообщения о неуспешной сборке

# **Заключение**

В результате проделанной работы было создано приложение на языке высокого уровня C# в среде программирования Microsoft Visual Studio, которое автоматизирует постройку планетарно-шнекового смесителя в Autodesk Inventor. От пользователя данного приложения требуется ввести необходимые размеры для автоматической постройки каждой детали, после чего программа выполнит построение. Далее пользователю необходимо нажать одну кнопку, открыть детали, которые программа помогает открыть, после чего происходит полноценная сборка установки.

# **Список используемой литературы**

* А.С. Тимонин. Основы конструирования и расчета химико-технологического оборудования и природоохранного оборудования. Справочник в 2 т. Т 2. – Калуга: Издательство Н. Бочкаревой, 2002 – с. 202 - 203

# Приложение 1

using System;  
using System.Collections.Generic;  
using System.Windows.Forms;  
using Inventor;  
  
namespace BesedinCoursework  
{  
    public partial class MainWindow : Form  
    {  
        public MainWindow()  
        {  
            InitializeComponent();  
        }  
        private Inventor.Application ThisApplication = null; // Проверка активности Inventor  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>(); // Имена документов деталей  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>(); // Ссылки на документы деталей  
        private AssemblyComponentDefinition oAssCompDef;  
        private void Plane(int OccurrenceOne, int PartPlaneOne, int OccurrenceTwo, int PartPlaneTwo, string Offset = "0", bool MateOrFlush = false) // Совмещение базовых плоскостей  
        {  
            ComponentOccurrence oOcc1 = oAssCompDef.Occurrences[OccurrenceOne];  
            ComponentOccurrence oOcc2 = oAssCompDef.Occurrences[OccurrenceTwo];  
            PartComponentDefinition oPartComDef = (PartComponentDefinition)oOcc1.Definition;  
            WorkPlane oPartPlane1 = oPartComDef.WorkPlanes[PartPlaneOne];  
            oPartComDef = (PartComponentDefinition)oOcc2.Definition;  
            WorkPlane oPartPlane2 = oPartComDef.WorkPlanes[PartPlaneTwo];  
            oOcc1.CreateGeometryProxy(oPartPlane1, out object oAsmPlane1Obj);  
            WorkPlaneProxy oAsmPlane1 = (WorkPlaneProxy)oAsmPlane1Obj;  
            oOcc2.CreateGeometryProxy(oPartPlane2, out object oAsmPlane2Obj);  
            WorkPlaneProxy oAsmPlane2 = (WorkPlaneProxy)oAsmPlane2Obj;  
            if (MateOrFlush == true) oAssCompDef.Constraints.AddMateConstraint(oAsmPlane1, oAsmPlane2, Offset);  
            else oAssCompDef.Constraints.AddFlushConstraint(oAsmPlane1, oAsmPlane2, Offset);  
        }  
        private void Axis(int OccurrenceOne, int PartAxisOne, int OccurrenceTwo, int PartAxisTwo, string Offset = "0") // Совмещение базовых осей  
        {  
            ComponentOccurrence oOcc1 = oAssCompDef.Occurrences[OccurrenceOne];  
            ComponentOccurrence oOcc2 = oAssCompDef.Occurrences[OccurrenceTwo];  
            PartComponentDefinition oPartComDef = (PartComponentDefinition)oOcc1.Definition;  
            WorkAxis oPartAxis1 = oPartComDef.WorkAxes[PartAxisOne];  
            oPartComDef = (PartComponentDefinition)oOcc2.Definition;  
            WorkAxis oPartAxis2 = oPartComDef.WorkAxes[PartAxisTwo];  
            oOcc1.CreateGeometryProxy(oPartAxis1, out object oAsmAxis1Obj);  
            WorkAxisProxy oAsmAxis1 = (WorkAxisProxy)oAsmAxis1Obj;  
            oOcc2.CreateGeometryProxy(oPartAxis2, out object oAsmAxis2Obj);  
            WorkAxisProxy oAsmAxis2 = (WorkAxisProxy)oAsmAxis2Obj;  
            oAssCompDef.Constraints.AddMateConstraint(oAsmAxis1, oAsmAxis2, Offset);  
        }  
        private void PlaneAngle(int OccurrenceOne, int PartPlaneOne, int OccurrenceTwo, int PartPlaneTwo, string Offset = "0") // Угол между базовыми плоскостями  
        {  
            ComponentOccurrence oOcc1 = oAssCompDef.Occurrences[OccurrenceOne];  
            ComponentOccurrence oOcc2 = oAssCompDef.Occurrences[OccurrenceTwo];  
            PartComponentDefinition oPartComDef = (PartComponentDefinition)oOcc1.Definition;  
            WorkPlane oPartPlane1 = oPartComDef.WorkPlanes[PartPlaneOne];  
            oPartComDef = (PartComponentDefinition)oOcc2.Definition;  
            WorkPlane oPartPlane2 = oPartComDef.WorkPlanes[PartPlaneTwo];  
            oOcc1.CreateGeometryProxy(oPartPlane1, out object oAsmPlane1Obj);  
            WorkPlaneProxy oAsmPlane1 = (WorkPlaneProxy)oAsmPlane1Obj;  
            oOcc2.CreateGeometryProxy(oPartPlane2, out object oAsmPlane2Obj);  
            WorkPlaneProxy oAsmPlane2 = (WorkPlaneProxy)oAsmPlane2Obj;  
            oAssCompDef.Constraints.AddAngleConstraint(oAsmPlane1, oAsmPlane2, Offset);  
        }  
        private void Surface(int OccurrenceOne, int PartFaceOne, int OccurrenceTwo, int PartFaceTwo, double Offset = 0, bool MateOrFlush = true) // Совмещение поверхностей  
        {  
            ComponentOccurrence oOcc1 = oAssCompDef.Occurrences[OccurrenceOne];  
            ComponentOccurrence oOcc2 = oAssCompDef.Occurrences[OccurrenceTwo];  
            Face oFace1 = oOcc1.SurfaceBodies[1].Faces[PartFaceOne];  
            Face oFace2 = oOcc2.SurfaceBodies[1].Faces[PartFaceTwo];  
            if (MateOrFlush == true) oAssCompDef.Constraints.AddMateConstraint(oFace1, oFace2, Offset);  
            else oAssCompDef.Constraints.AddFlushConstraint(oFace1, oFace2, Offset);  
        }  
        private void InventorControl\_Click(object sender, EventArgs e) // Запуск окна "Запуск Inventor"  
        {  
            InventorControl IC = new InventorControl();  
            IC.ShowDialog();  
        }  
        private void MainBody\_Click(object sender, EventArgs e) // Запуск окна "Построение конического корпуса"  
        {  
            MainBody MB = new MainBody();  
            MB.ShowDialog();  
        }  
        private void Top\_Click(object sender, EventArgs e) // Запуск окна "Построение крышки"  
        {  
            Top T = new Top();  
            T.ShowDialog();  
        }  
        private void Screw\_Click(object sender, EventArgs e) // Запуск окна "Построение шнека"  
        {  
            Screw W = new Screw();  
            W.ShowDialog();  
        }  
        private void Bottom\_Click(object sender, EventArgs e) // Запуск окна "Построение корпуса для выгрузки материалов"  
        {  
            Bottom B = new Bottom();  
            B.ShowDialog();  
        }  
        private void Hold\_Click(object sender, EventArgs e) // Запуск окна "Построение боковых опор"  
        {  
            Hold H = new Hold();  
            H.ShowDialog();  
        }  
        private void Build\_Click(object sender, EventArgs e) // Выполнение сборки  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            string[] PartIndex = { "КК", "К", "Шнек", "КВ", "БО" };  
            string[] PartName = { "Конический корпус", "Крышка", "Шнек", "Корпус для выгрузки материалов", "Боковые опоры" };  
            for (int I = 0; I < PartIndex.Length; I++)  
            {  
                openFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                openFileDialog1.Title = "Открыть файл \"" + PartName[I] + "\"";  
                if (openFileDialog1.ShowDialog() == DialogResult.OK)  
                {  
                    if (!string.IsNullOrWhiteSpace(openFileDialog1.FileName))  
                    {  
                        oPartDoc[PartIndex[I]] = (PartDocument)ThisApplication.Documents.Open(openFileDialog1.FileName, true);  
                        oPartDoc[PartIndex[I]].DisplayName = "" + PartName[I] + "";  
                        oFileName[PartIndex[I]] = openFileDialog1.FileName;  
                    }  
                }  
                else  
                {  
                    MessageBox.Show("Сборка была прервана по причине того, \nчто не все детали были открыты приложением.", "Планетарно-шнековой смеситель в Inventor");  
                    return;  
                }  
            }  
            // Инициализация сборки  
            AssemblyDocument oAssDoc = (AssemblyDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kAssemblyDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kAssemblyDocumentObject));  
            oAssCompDef = oAssDoc.ComponentDefinition;  
            oAssDoc.DisplayName = "Планетарно-шнековый смеситель";  
            for (int I = 0; I < PartIndex.Length; I++) oAssDoc.ComponentDefinition.Occurrences.Add(oFileName[PartIndex[I]], ThisApplication.TransientGeometry.CreateMatrix());  
            // Присоединение крышки  
            Surface(1, 1, 2, 19);  
            Surface(1, 2, 2, 20);  
            Surface(1, 9, 2, 114);  
            // Присоединение шнека  
            Axis(1, 3, 3, 3);  
            Plane(1, 3, 3, 3);  
            PlaneAngle(1, 1, 3, 1, Convert.ToString(BesedinCoursework.MainBody.Degree));  
            // Присоединение корпуса для выгрузки материалов  
            Axis(1, 2, 4, 2);  
            Surface(1, 9, 4, 28, -BesedinCoursework.MainBody.H / 10, false);  
            Plane(1, 3, 4, 3);  
            // Присоединение боковых опор  
            Plane(1, 3, 5, 3);  
            Plane(1, 2, 5, 2, Convert.ToString(BesedinCoursework.MainBody.H - BesedinCoursework.MainBody.H / 4));  
            Plane(1, 1, 5, 1, Convert.ToString((BesedinCoursework.MainBody.H - (BesedinCoursework.MainBody.H - BesedinCoursework.MainBody.H / 4)) \* 1.32));  
            ObjectCollection oB = ThisApplication.TransientObjects.CreateObjectCollection();  
            oB.Add(oAssCompDef.Occurrences[5]);  
            oAssCompDef.OccurrencePatterns.AddCircularPattern(oB, oAssCompDef.WorkAxes[2], true, "90 degree", 4);  
            MessageBox.Show("Сборка планетарно-шнекового смесителя завершена.", "Планетарно-шнековый смеситель в Inventor");  
        }  
    }

public partial class InventorControl : Form  
    {  
        public InventorControl()  
        {  
            InitializeComponent();  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
                if (ThisApplication != null) label1.Text = "Inventor запущен.";  
                InventorVersions.Enabled = false;  
                InventorLaunch.Enabled = false;  
            }  
            catch  
            {  
                label1.Text = "Запустите Inventor!";  
                InventorVersions.Enabled = true;  
                InventorLaunch.Enabled = true;  
            }  
            this.InventorVersions.Items.AddRange(new object[] { "2015", "2016", "2017", "2018", "2019" });  
            InventorVersions.Text = "2017";  
        }  
        private Inventor.Application ThisApplication = null; // Проверка активности Inventor  
        private void InventorLaunch\_Click(object sender, EventArgs e) // Кнопка запуска Inventor  
        {  
            try  
            {  
                System.Diagnostics.Process.Start("C://Program Files/Autodesk/Inventor " + InventorVersions.Text.ToString() + "/Bin/Inventor.exe");  
                Close();  
            }  
            catch  
            {  
                try  
                {  
                    System.Diagnostics.Process.Start("D://Program Files/Autodesk/Inventor " + InventorVersions.Text.ToString() + "/Bin/Inventor.exe");  
                    Close();  
                }  
                catch  
                {  
                    try  
                    {  
                        System.Diagnostics.Process.Start("E://Program Files/Autodesk/Inventor " + InventorVersions.Text.ToString() + "/Bin/Inventor.exe");  
                        Close();  
                    }  
                    catch  
                    {  
                        MessageBox.Show("Данная версия Inventor не была найдена на дисках C, D, E.", "Запуск Inventor");  
                    }  
                }  
            }  
        }  
    }

public partial class MainBody : Form  
    {  
        public MainBody()  
        {  
            InitializeComponent();  
            textBox1.Text = Convert.ToString(H);  
            textBox2.Text = Convert.ToString(D);  
            textBox3.Text = Convert.ToString(B);  
            textBox4.Text = Convert.ToString(A);  
            textBox5.Text = Convert.ToString(T);  
            textBox7.Text = Convert.ToString(Rb);  
            textBox8.Text = Convert.ToString(Rm);  
            textBox10.Text = Convert.ToString(Ds);  
        }  
        private Inventor.Application ThisApplication = null;  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>();  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>();  
        private Dictionary<string, PartComponentDefinition> oCompDef = new Dictionary<string, PartComponentDefinition>();  
        private Dictionary<string, TransientGeometry> oTransGeom = new Dictionary<string, TransientGeometry>();  
        private Dictionary<string, Transaction> oTrans = new Dictionary<string, Transaction>();  
        public static double H = 1635, Degree = 18;  
        double D = 1530, B = 100, A = 15, T = 40, Rb = 730, Rm = 24, Ds = 250;  
        private void Build\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            oPartDoc["КК"] = (PartDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kPartDocumentObject));  
            oCompDef["КК"] = oPartDoc["КК"].ComponentDefinition;  
            oTransGeom["КК"] = ThisApplication.TransientGeometry;  
            oFileName["КК"] = null;  
            oPartDoc["КК"].DisplayName = "Конический корпус";  
            Parameters oParameters = oCompDef["КК"].Parameters; // Хранит параметры детали  
            SketchPoint[] point = new SketchPoint[8];  
            SketchLine[] line = new SketchLine[7];  
            SketchCircle[] circle = new SketchCircle[1];  
            // Создание конической основы  
            PlanarSketch oSketch = oCompDef["КК"].Sketches.Add(oCompDef["КК"].WorkPlanes[3]);  
            point[0] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(0.1, 0.1), false);  
            point[1] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(0.1, 0.2), false);  
            line[0] = oSketch.SketchLines.AddByTwoPoints(point[0], point[1]);  
            point[2] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(D / 2 - B, H), false);  
            point[3] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(D / 2, H), false);  
            point[4] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(D / 2, H - T), false);  
            point[5] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(D / 2 - B + 1, H - T), false);  
            point[6] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(Ds + 1, 0), false);  
            point[7] = oSketch.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(Ds, 0), false);  
            line[1] = oSketch.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[2] = oSketch.SketchLines.AddByTwoPoints(point[3], point[4]);  
            line[3] = oSketch.SketchLines.AddByTwoPoints(point[4], point[5]);  
            line[4] = oSketch.SketchLines.AddByTwoPoints(point[5], point[6]);  
            line[5] = oSketch.SketchLines.AddByTwoPoints(point[6], point[7]);  
            line[6] = oSketch.SketchLines.AddByTwoPoints(point[7], point[2]);  
            Point2d SketchSize = oTransGeom["КК"].CreatePoint2d(-1, -1); // Место для выноса размеров  
            oSketch.GeometricConstraints.AddVertical((SketchEntity)line[0]); // Выравнивание линии по вертикали  
            oSketch.GeometricConstraints.AddHorizontal((SketchEntity)line[1]); // Выравнивание линии по горизонтали  
            oSketch.GeometricConstraints.AddVertical((SketchEntity)line[2]);  
            oSketch.GeometricConstraints.AddHorizontal((SketchEntity)line[3]);  
            oSketch.GeometricConstraints.AddHorizontal((SketchEntity)line[5]);  
            oSketch.GeometricConstraints.AddHorizontalAlign(point[0], point[7]); // Выравнивает точки по горизонтали  
            oSketch.GeometricConstraints.AddParallel((SketchEntity)line[4], (SketchEntity)line[6]); // Добавление параллельности линиям  
            oSketch.DimensionConstraints.AddTwoPointDistance(point[0], point[7], DimensionOrientationEnum.kHorizontalDim, SketchSize); // Ds  
            oSketch.DimensionConstraints.AddTwoPointDistance(point[0], point[3], DimensionOrientationEnum.kVerticalDim, SketchSize); // H (по вертикали)  
            oSketch.DimensionConstraints.AddTwoPointDistance(point[0], point[3], DimensionOrientationEnum.kHorizontalDim, SketchSize); // D (по горизонтали)  
            oSketch.DimensionConstraints.AddTwoPointDistance(point[2], point[3], DimensionOrientationEnum.kHorizontalDim, SketchSize); // B (по горизонтали)  
            oSketch.DimensionConstraints.AddOffset(line[4], (SketchEntity)line[6], SketchSize, false); // A  
            oSketch.DimensionConstraints.AddTwoPointDistance(point[3], point[4], DimensionOrientationEnum.kVerticalDim, SketchSize); // T  
            oParameters["d0"].Expression = "" + Ds / 2 + " mm";  
            oParameters["d1"].Expression = "" + H + " mm";  
            oParameters["d2"].Expression = "" + D / 2 + " mm";  
            oParameters["d3"].Expression = "" + B + " mm";  
            oParameters["d4"].Expression = "" + A + " mm";  
            oParameters["d5"].Expression = "" + T + " mm";  
            point[0].MoveTo(oTransGeom["КК"].CreatePoint2d(0, 0)); // Выравнивание осевой линии центра  
            oSketch.DimensionConstraints.AddTwoLineAngle(line[6], line[0], oTransGeom["КК"].CreatePoint2d(10, 40),true);  
            Degree = oParameters["d6"].\_Value \* (180 / Math.PI);  
            Profile oProfile = (Profile)oSketch.Profiles.AddForSolid();  
            RevolveFeature revolvefeature = oCompDef["КК"].Features.RevolveFeatures.AddFull(oProfile, line[0], PartFeatureOperationEnum.kJoinOperation);  
            // Отверстия  
            PlanarSketch oSketch2 = oCompDef["КК"].Sketches.Add(oCompDef["КК"].WorkPlanes[2]);  
            point[0] = oSketch2.SketchPoints.Add(oTransGeom["КК"].CreatePoint2d(0, Rb / 10), false);  
            circle[0] = oSketch2.SketchCircles.AddByCenterRadius(point[0], Rm / 10 / 2);  
            Profile oProfile2 = (Profile)oSketch2.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef2 = oCompDef["КК"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile2, H / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile2);  
            ObjectCollection objCollection2 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection2.Add(oExtrudeDef2);  
            CircularPatternFeature CircularPatternFeature2 = oCompDef["КК"].Features.CircularPatternFeatures.Add(objCollection2, oCompDef["КК"].WorkAxes[2], true, 3, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            MessageBox.Show("Создание конического корпуса завершено.", "Построение конического корпуса");  
        }  
        private void Save\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            try  
            {  
                saveFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                saveFileDialog1.Title = "Сохранение конического корпуса";  
                saveFileDialog1.FileName = oPartDoc["КК"].DisplayName;  
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
                    if (!string.IsNullOrWhiteSpace(saveFileDialog1.FileName))  
                    {  
                        oPartDoc["КК"].SaveAs(saveFileDialog1.FileName, false);  
                        oFileName["КК"] = saveFileDialog1.FileName;  
                    }  
                Close();  
            }  
            catch  
            {  
                MessageBox.Show("Деталь ещё не создана, либо программа не видит созданную деталь.", "Построение конического корпуса");  
            }  
        }  
        private void error\_check(System.Windows.Forms.TextBox textBox, double Number) // Проверка ошибки в textBox  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox.Text) <= 0)  
                {  
                    MessageBox.Show("Введите значение больше нуля!", "Построение конического корпуса");  
                    textBox.Text = Convert.ToString(Number);  
                }  
            }  
            catch  
            {  
                textBox.Text = Convert.ToString(Number);  
            }  
        }  
        private void textBox1\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox1, 1635);  
            H = Convert.ToDouble(textBox1.Text);  
        }  
        private void textBox2\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox2, 1530);  
            D = Convert.ToDouble(textBox2.Text);  
        }  
        private void textBox3\_TextChanged(object sender, EventArgs e)  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox3.Text) <= 0 | Convert.ToDouble(textBox3.Text) > D / 2)   
                {  
                    MessageBox.Show("Введите значение больше нуля и меньше " + Convert.ToString(D / 2) + "!", "Построение конического корпуса");  
                    B = D / 2 / 2;  
                    textBox3.Text = Convert.ToString(B);  
                }  
                else B = Convert.ToDouble(textBox3.Text);  
            }  
            catch  
            {  
                B = D / 2 / 2;  
                textBox3.Text = Convert.ToString(B);  
                return;  
            }  
        }  
        private void textBox4\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox4, 15);  
            A = Convert.ToDouble(textBox4.Text);  
        }  
        private void textBox5\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox5, 40);  
            T = Convert.ToDouble(textBox5.Text);  
        }  
        private void textBox7\_TextChanged(object sender, EventArgs e)  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox7.Text) < D / 2 - B - Rm | Convert.ToDouble(textBox7.Text) >= D / 2 + Rm)  
                {  
                    MessageBox.Show("Введите значение от " + Convert.ToString(D / 2 - B) + " до " + Convert.ToString(D / 2) + "!", "Построение конического корпуса");  
                    Rb = 730;  
                    textBox7.Text = Convert.ToString(Rb);  
                }  
                else Rb = Convert.ToDouble(textBox7.Text);  
            }  
            catch  
            {  
                Rb = 730;  
                textBox7.Text = Convert.ToString(Rb);  
                return;  
            }  
        }  
        private void textBox8\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox8, 24);  
            Rm = Convert.ToDouble(textBox8.Text);  
        }  
        private void textBox10\_TextChanged(object sender, EventArgs e)  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox10.Text) > D / 2 - B | Convert.ToDouble(textBox10.Text) == 0)  
                {  
                    MessageBox.Show("Введите значение от нуля до " + (D / 2 - B) + "!", "Построение конического корпуса");  
                    Ds = (D / 2 - B) / 2;  
                    textBox10.Text = Convert.ToString(Ds);  
                }  
                else Ds = Convert.ToDouble(textBox10.Text);  
            }  
            catch  
            {  
                Ds = (D / 2 - B) / 2;  
                textBox10.Text = Convert.ToString(Ds);  
                return;  
            }  
        }  
    }

public partial class Top : Form  
    {  
        public Top()  
        {  
            InitializeComponent();  
            textBox1.Text = Convert.ToString(D);  
            textBox2.Text = Convert.ToString(D1);  
            textBox3.Text = Convert.ToString(H);  
            textBox4.Text = Convert.ToString(T);  
            textBox5.Text = Convert.ToString(A);  
            textBox6.Text = Convert.ToString(ACount);  
            textBox7.Text = Convert.ToString(MBRb);  
            textBox8.Text = Convert.ToString(MBRm);  
            textBox9.Text = Convert.ToString(RR);  
            textBox10.Text = Convert.ToString(HR);  
            textBox11.Text = Convert.ToString(DR);  
            textBox12.Text = Convert.ToString(hR);  
            textBox13.Text = Convert.ToString(dR);  
            textBox14.Text = Convert.ToString(TR);  
            textBox15.Text = Convert.ToString(OR);  
            textBox16.Text = Convert.ToString(oR);  
            textBox17.Text = Convert.ToString(R1R);  
            textBox18.Text = Convert.ToString(D1R);  
            textBox19.Text = Convert.ToString(d1R);  
            textBox20.Text = Convert.ToString(T1R);  
            textBox21.Text = Convert.ToString(O1R);  
            textBox22.Text = Convert.ToString(o1R);  
        }  
        private Inventor.Application ThisApplication = null;  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>();  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>();  
        private Dictionary<string, PartComponentDefinition> oCompDef = new Dictionary<string, PartComponentDefinition>();  
        private Dictionary<string, TransientGeometry> oTransGeom = new Dictionary<string, TransientGeometry>();  
        double D = 1530, D1 = 1400, T = 40, H = 200, A = 40, ACount = 8; // Основа крышки  
        double HR = 305, hR = 30, RR = 580, DR = 180, dR = 120, OR = 75, oR = 18, TR = 100; // Резервные проходы  
        double MBRb = 730, MBRm = 24; // Отверстия под болты  
        double D1R = 380, d1R = 260, T1R = 240, o1R = 22, O1R = 168, R1R = 540; // Загрузка сыпучих материалов  
        private void Build\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            D /= 10; D1 /= 10; T /= 10; H /= 10;  
            oPartDoc["К"] = (PartDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kPartDocumentObject));  
            oCompDef["К"] = oPartDoc["К"].ComponentDefinition;  
            oTransGeom["К"] = ThisApplication.TransientGeometry;  
            oFileName["К"] = null;  
            oPartDoc["К"].DisplayName = "Крышка";  
            Parameters oParameters = oCompDef["К"].Parameters;  
            SketchPoint[] point = new SketchPoint[8];  
            SketchLine[] line = new SketchLine[8];  
            SketchCircle[] circle = new SketchCircle[1];  
            // Основа корпуса  
            PlanarSketch oSketch = oCompDef["К"].Sketches.Add(oCompDef["К"].WorkPlanes[3]);  
            point[0] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, H - T), false);  
            point[1] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, H), false);  
            point[2] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D1 / 2, H), false);  
            point[3] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D1 / 2, T), false);  
            point[4] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D / 2, T), false);  
            point[5] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D / 2, 0), false);  
            point[6] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D1 / 2 - T, 0), false);  
            point[7] = oSketch.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D1 / 2 - T, H - T), false);  
            line[0] = oSketch.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch.SketchLines.AddByTwoPoints(point[3], point[4]);  
            line[4] = oSketch.SketchLines.AddByTwoPoints(point[4], point[5]);  
            line[5] = oSketch.SketchLines.AddByTwoPoints(point[5], point[6]);  
            line[6] = oSketch.SketchLines.AddByTwoPoints(point[6], point[7]);  
            line[7] = oSketch.SketchLines.AddByTwoPoints(point[7], point[0]);  
            Profile oProfile = default(Profile);  
            oProfile = (Profile)oSketch.Profiles.AddForSolid();  
            RevolveFeature revolvefeature = default(RevolveFeature);  
            revolvefeature = oCompDef["К"].Features.RevolveFeatures.AddFull(oProfile, line[0], PartFeatureOperationEnum.kJoinOperation);  
            // Ребра жесткости  
            PlanarSketch oSketch1 = oCompDef["К"].Sketches.Add(oCompDef["К"].WorkPlanes[3]);  
            point[0] = oSketch1.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D1 / 2 - T, H), false);  
            point[1] = oSketch1.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(point[0].Geometry.X + T \* 2, H), false);  
            point[2] = oSketch1.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D / 2, T \* 2), false);  
            point[3] = oSketch1.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(D / 2, T), false);  
            point[4] = oSketch1.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(point[0].Geometry.X, T), false);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[3], point[4]);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[4], point[0]);  
            Profile oProfile1 = (Profile)oSketch1.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef1 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile1, A / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile1);  
            ObjectCollection objCollection1 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection1.Add(oExtrudeDef1);  
            CircularPatternFeature CircularPatternFeature2 = oCompDef["К"].Features.CircularPatternFeatures.Add(objCollection1, oCompDef["К"].WorkAxes[2], true, ACount, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            // Резервные проходы (2 штуки)  
            WorkPlane oWorkPlane2 = oCompDef["К"].WorkPlanes.AddByPlaneAndOffset(oCompDef["К"].WorkPlanes[2], HR / 10, false);  
            oWorkPlane2.Visible = false;  
            PlanarSketch oSketch2 = oCompDef["К"].Sketches.Add(oWorkPlane2, false);  
            circle[0] = oSketch2.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(RR / 10, 0), DR / 10 / 2);  
            Profile oProfile2 = (Profile)oSketch2.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef2 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile2, hR / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile2);  
            PlanarSketch oSketch3 = oCompDef["К"].Sketches.Add(oWorkPlane2, false);  
            circle[0] = oSketch3.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(RR / 10, 0), dR / 10 / 2);  
            Profile oProfile3 = (Profile)oSketch3.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef3 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile3, HR / 10 - H, PartFeatureExtentDirectionEnum.kNegativeExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile3);  
            PlanarSketch oSketch4 = oCompDef["К"].Sketches.Add(oWorkPlane2, false);  
            circle[0] = oSketch4.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(RR / 10, 0), TR / 10 / 2);  
            Profile oProfile4 = (Profile)oSketch4.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef4 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile4, 1000, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile4);  
            PlanarSketch oSketch5 = oCompDef["К"].Sketches.Add(oWorkPlane2, false);  
            circle[0] = oSketch5.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(RR / 10, OR / 10), oR / 10 / 2);  
            Profile oProfile5 = (Profile)oSketch5.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef5 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile5, hR / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile5);  
            WorkAxis Axis5 = oCompDef["К"].WorkAxes.AddByRevolvedFace(oExtrudeDef4.Faces[1]);  
            Axis5.Visible = false;  
            ObjectCollection objCollection5 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection5.Add(oExtrudeDef5);  
            CircularPatternFeature CircularPatternFeature5 = oCompDef["К"].Features.CircularPatternFeatures.Add(objCollection5, Axis5, true, 8, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            ObjectCollection objCollection6 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection6.Add(oExtrudeDef2);  
            objCollection6.Add(oExtrudeDef3);  
            objCollection6.Add(oExtrudeDef4);  
            objCollection6.Add(oExtrudeDef5);  
            objCollection6.Add(CircularPatternFeature5);  
            CircularPatternFeature CircularPatternFeature6 = oCompDef["К"].Features.CircularPatternFeatures.Add(objCollection6, oCompDef["К"].WorkAxes[2], true, 2, "150 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            // Отверстия под болты  
            PlanarSketch oSketch7 = oCompDef["К"].Sketches.Add(oCompDef["К"].WorkPlanes[2]);  
            point[0] = oSketch7.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch7.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, MBRb / 10), false);  
            point[2] = oSketch7.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(MBRb / 10, 0), false);  
            line[0] = oSketch7.SketchLines.AddByTwoPoints(point[1], point[0]);  
            line[1] = oSketch7.SketchLines.AddByTwoPoints(point[0], point[2]);  
            oSketch7.DimensionConstraints.AddTwoLineAngle(line[0], line[1], oTransGeom["К"].CreatePoint2d(1, 1));  
            oSketch7.GeometricConstraints.AddHorizontal((SketchEntity)line[1]);  
            circle[0] = oSketch7.SketchCircles.AddByCenterRadius(point[1], MBRm / 10 / 2);  
            oParameters["d21"].Expression = "82.5 degree";  
            oSketch7.GeometricConstraints.AddCoincident((SketchEntity)circle[0].CenterSketchPoint, (SketchEntity)point[1]);  
            Profile oProfile7 = (Profile)oSketch7.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef7 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile7, 1000, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile7);  
            ObjectCollection objCollection7 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection7.Add(oExtrudeDef7);  
            CircularPatternFeature CircularPatternFeature7 = oCompDef["К"].Features.CircularPatternFeatures.Add(objCollection7, oCompDef["К"].WorkAxes[2], true, 3, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            // Загрузка сыпучих материалов  
            PlanarSketch oSketch8 = oCompDef["К"].Sketches.Add(oWorkPlane2, false);  
            point[0] = oSketch8.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch8.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(0, -R1R / 10), false);  
            point[2] = oSketch8.SketchPoints.Add(oTransGeom["К"].CreatePoint2d(-R1R / 10, 0), false);  
            line[0] = oSketch8.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch8.SketchLines.AddByTwoPoints(point[0], point[2]);  
            oSketch8.DimensionConstraints.AddTwoLineAngle(line[0], line[1], oTransGeom["К"].CreatePoint2d(1, 1));  
            oSketch8.GeometricConstraints.AddVertical((SketchEntity)line[0]);  
            circle[0] = oSketch8.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(-R1R / 10, -1), D1R / 10 / 2);  
            oParameters["d27"].Expression = "75 degree";  
            oSketch8.GeometricConstraints.AddCoincident((SketchEntity)circle[0].CenterSketchPoint, (SketchEntity)point[2]);  
            Profile oProfile8 = (Profile)oSketch8.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef8 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile8, hR / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile8);  
            PlanarSketch oSketch9 = oCompDef["К"].Sketches.Add(oExtrudeDef8.Faces[1], false);  
            circle[0] = oSketch9.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(0, 0), d1R / 10 / 2);  
            Profile oProfile9 = (Profile)oSketch9.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef9 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile9, HR / 10 - H, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile9);  
            PlanarSketch oSketch10 = oCompDef["К"].Sketches.Add(oExtrudeDef8.Faces[1], false);  
            circle[0] = oSketch10.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(0, 0), T1R / 10 / 2);  
            Profile oProfile10 = (Profile)oSketch10.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef10 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile10, 1000, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile10);  
            PlanarSketch oSketch11 = oCompDef["К"].Sketches.Add(oExtrudeDef8.Faces[1], false);  
            circle[0] = oSketch11.SketchCircles.AddByCenterRadius(oTransGeom["К"].CreatePoint2d(0, O1R / 10), o1R / 10 / 2);  
            Profile oProfile11 = (Profile)oSketch11.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef11 = oCompDef["К"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile11, hR / 10, PartFeatureExtentDirectionEnum.kNegativeExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile5);  
            WorkAxis Axis11 = oCompDef["К"].WorkAxes.AddByRevolvedFace(oExtrudeDef10.Faces[1]);  
            Axis11.Visible = false;  
            ObjectCollection objCollection11 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection11.Add(oExtrudeDef11);  
            CircularPatternFeature CircularPatternFeature11 = oCompDef["К"].Features.CircularPatternFeatures.Add(objCollection11, Axis11, true, 12, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            D \*= 10; D1 \*= 10; T \*= 10; H \*= 10;  
            MessageBox.Show("Создание крышки завершено.", "Построение крышки");  
        }  
        private void Save\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            try  
            {  
                saveFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                saveFileDialog1.Title = "Сохранение крышки";  
                saveFileDialog1.FileName = oPartDoc["К"].DisplayName;  
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
                {  
                    if (!string.IsNullOrWhiteSpace(saveFileDialog1.FileName))  
                    {  
                        oPartDoc["К"].SaveAs(saveFileDialog1.FileName, false);  
                        oFileName["К"] = saveFileDialog1.FileName;  
                    }  
                }  
                Close();  
            }  
            catch  
            {  
                MessageBox.Show("Деталь ещё не создана, либо программа не видит созданную деталь.", "Построение крышки");  
            }  
        }  
        private void error\_check(System.Windows.Forms.TextBox textBox, double Number) // Проверка ошибки в textBox  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox.Text) <= 0)  
                {  
                    MessageBox.Show("Введите значение больше нуля!", "Построение корпуса для выгрузки материалов");  
                    textBox.Text = Convert.ToString(Number);  
                }  
            }  
            catch  
            {  
                textBox.Text = Convert.ToString(Number);  
            }  
        }  
        private void textBox1\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox1, 1530);  
            D = Convert.ToDouble(textBox1.Text);  
        }  
        private void textBox2\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox2, 1340);  
            D1 = Convert.ToDouble(textBox2.Text);  
        }  
        private void textBox3\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox3, 200);  
            H = Convert.ToDouble(textBox3.Text);  
        }  
        private void textBox4\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox4, 40);  
            T = Convert.ToDouble(textBox4.Text);  
        }  
        private void textBox5\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox5, 40);  
            A = Convert.ToDouble(textBox5.Text);  
        }  
        private void textBox6\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox6, 8);  
            ACount = Convert.ToDouble(textBox6.Text);  
        }  
        private void textBox7\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox7, 730);  
            MBRb = Convert.ToDouble(textBox7.Text);  
        }  
        private void textBox8\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox8, 24);  
            MBRm = Convert.ToDouble(textBox8.Text);  
        }  
        private void textBox9\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox9, 580);  
            RR = Convert.ToDouble(textBox9.Text);  
        }  
        private void textBox10\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox10, 305);  
            HR = Convert.ToDouble(textBox10.Text);  
        }  
        private void textBox11\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox11, 180);  
            DR = Convert.ToDouble(textBox11.Text);  
        }  
        private void textBox12\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox12, 30);  
            hR = Convert.ToDouble(textBox12.Text);  
        }  
        private void textBox13\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox13, 120);  
            dR = Convert.ToDouble(textBox13.Text);  
        }  
        private void textBox14\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox14, 100);  
            TR = Convert.ToDouble(textBox14.Text);  
        }  
        private void textBox15\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox15, 75);  
            OR = Convert.ToDouble(textBox15.Text);  
        }  
        private void textBox16\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox16, 18);  
            oR = Convert.ToDouble(textBox16.Text);  
        }  
        private void textBox17\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox17, 540);  
            R1R = Convert.ToDouble(textBox17.Text);  
        }  
        private void textBox18\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox18, 380);  
            D1R = Convert.ToDouble(textBox18.Text);  
        }  
        private void textBox19\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox19, 260);  
            d1R = Convert.ToDouble(textBox19.Text);  
        }  
        private void textBox20\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox20, 240);  
            T1R = Convert.ToDouble(textBox20.Text);  
        }  
        private void textBox21\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox21, 168);  
            O1R = Convert.ToDouble(textBox21.Text);  
        }  
        private void textBox22\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox22, 22);  
            o1R = Convert.ToDouble(textBox22.Text);  
        }  
    }

public partial class Screw : Form  
    {  
        public Screw()  
        {  
            InitializeComponent();  
            textBox1.Text = Convert.ToString(D);  
            textBox2.Text = Convert.ToString(D1);  
            textBox3.Text = Convert.ToString(H);  
            textBox4.Text = Convert.ToString(H1);  
            textBox5.Text = Convert.ToString(H2);  
            textBox6.Text = Convert.ToString(T);  
            textBox7.Text = Convert.ToString(H3);  
            textBox8.Text = Convert.ToString(A);  
        }  
        private Inventor.Application ThisApplication = null;  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>();  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>();  
        private Dictionary<string, PartComponentDefinition> oCompDef = new Dictionary<string, PartComponentDefinition>();  
        private Dictionary<string, TransientGeometry> oTransGeom = new Dictionary<string, TransientGeometry>();  
        double H = 1700, D = 220, D1 = 100, H1 = 120, H2 = 1300, T = 25, H3 = 1500, A = 40;  
        private void Build\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            H /= 10; D /= 10; D1 /= 10; H1 /= 10; H2 /= 10; T /= 10; H3 /= 10; A /= 10;  
            oPartDoc["Шнек"] = (PartDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kPartDocumentObject));  
            oCompDef["Шнек"] = oPartDoc["Шнек"].ComponentDefinition;  
            oTransGeom["Шнек"] = ThisApplication.TransientGeometry;  
            oFileName["Шнек"] = null;  
            oPartDoc["Шнек"].DisplayName = "Шнек";  
            SketchPoint[] point = new SketchPoint[4];  
            SketchLine[] line = new SketchLine[4];  
            // Создание цилиндра  
            PlanarSketch oSketch = oCompDef["Шнек"].Sketches.Add(oCompDef["Шнек"].WorkPlanes[3]);  
            point[0] = oSketch.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, H), false);  
            point[2] = oSketch.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D1 / 2, H), false);  
            point[3] = oSketch.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D1 / 2, 0), false);  
            line[0] = oSketch.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile = (Profile)oSketch.Profiles.AddForSolid();  
            RevolveFeature revolvefeature = oCompDef["Шнек"].Features.RevolveFeatures.AddFull(oProfile, line[0], PartFeatureOperationEnum.kJoinOperation);  
            // Создание пружины  
            PlanarSketch oSketch1 = oCompDef["Шнек"].Sketches.Add(oCompDef["Шнек"].WorkPlanes[3]);  
            point[0] = oSketch1.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D1 / 2, 0), false);  
            point[1] = oSketch1.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D / 2, 0), false);  
            point[3] = oSketch1.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D1 / 2, T), false);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch1.SketchLines.AddByTwoPoints(point[1], point[3]);  
            line[2] = oSketch1.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile1 = (Profile)oSketch1.Profiles.AddForSolid();  
            CoilFeature coil = oCompDef["Шнек"].Features.CoilFeatures.AddByPitchAndHeight(oProfile1, oCompDef["Шнек"].WorkAxes[2], H1, H2, PartFeatureOperationEnum.kJoinOperation, false, false, 0, false, 0, 0, true);  
            // Верхняя граница пружины  
            PlanarSketch oSketch2 = oCompDef["Шнек"].Sketches.Add(oCompDef["Шнек"].WorkPlanes[3]);  
            point[0] = oSketch2.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, H2), false);  
            point[1] = oSketch2.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, H2 + T), false);  
            point[2] = oSketch2.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D / 2, H2 + T), false);  
            point[3] = oSketch2.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D / 2, H2), false);  
            line[0] = oSketch2.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch2.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch2.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch2.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile2 = (Profile)oSketch2.Profiles.AddForSolid();  
            RevolveFeature revolvefeature2 = oCompDef["Шнек"].Features.RevolveFeatures.AddFull(oProfile2, line[0], PartFeatureOperationEnum.kJoinOperation);  
            // Крепление  
            PlanarSketch oSketch3 = oCompDef["Шнек"].Sketches.Add(oCompDef["Шнек"].WorkPlanes[3]);  
            point[0] = oSketch3.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, H3), false);  
            point[1] = oSketch3.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(0, H3 + A), false);  
            point[2] = oSketch3.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D / 2, H3 + A), false);  
            point[3] = oSketch3.SketchPoints.Add(oTransGeom["Шнек"].CreatePoint2d(D / 2, H3), false);  
            line[0] = oSketch3.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch3.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch3.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch3.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile3 = (Profile)oSketch3.Profiles.AddForSolid();  
            RevolveFeature revolvefeature3 = oCompDef["Шнек"].Features.RevolveFeatures.AddFull(oProfile3, line[0], PartFeatureOperationEnum.kJoinOperation);  
            H \*= 10; D \*= 10; D1 \*= 10; H1 \*= 10; H2 \*= 10; T \*= 10; H3 \*= 10; A \*= 10;  
            MessageBox.Show("Создание шнека завершено.", "Построение шнека");  
        }  
        private void Save\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            try  
            {  
                saveFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                saveFileDialog1.Title = "Сохранение шнека";  
                saveFileDialog1.FileName = oPartDoc["Шнек"].DisplayName;  
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
                    if (!string.IsNullOrWhiteSpace(saveFileDialog1.FileName))  
                    {  
                        oPartDoc["Шнек"].SaveAs(saveFileDialog1.FileName, false);  
                        oFileName["Шнек"] = saveFileDialog1.FileName;  
                    }  
                Close();  
            }  
            catch  
            {  
                MessageBox.Show("Деталь ещё не создана, либо программа не видит созданную деталь.", "Построение шнека");  
            }  
        }  
        private void error\_check(System.Windows.Forms.TextBox textBox, double Number) // Проверка ошибки в textBox  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox.Text) <= 0)  
                {  
                    MessageBox.Show("Введите значение больше нуля!", "Построение корпуса для выгрузки материалов");  
                    textBox.Text = Convert.ToString(Number);  
                }  
            }  
            catch  
            {  
                textBox.Text = Convert.ToString(Number);  
            }  
        }  
        private void textBox1\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox1, 220);  
            D = Convert.ToDouble(textBox1.Text);  
        }  
        private void textBox2\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox2, 100);  
            D1 = Convert.ToDouble(textBox2.Text);  
        }  
        private void textBox3\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox3, 1600);  
            H = Convert.ToDouble(textBox3.Text);  
        }  
        private void textBox4\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox4, 120);  
            H1 = Convert.ToDouble(textBox4.Text);  
        }  
        private void textBox5\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox5, 1300);  
            H2 = Convert.ToDouble(textBox5.Text);  
        }  
        private void textBox6\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox6, 25);  
            T = Convert.ToDouble(textBox6.Text);  
        }  
        private void textBox7\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox7, 1500);  
            H3 = Convert.ToDouble(textBox7.Text);  
        }  
        private void textBox8\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox8, 40);  
            A = Convert.ToDouble(textBox8.Text);  
        }  
    }

public partial class Bottom : Form  
    {  
        public Bottom()  
        {  
            InitializeComponent();  
            textBox1.Text = Convert.ToString(H1 + H2 + HB + H3);  
            textBox2.Text = Convert.ToString(D1);  
            textBox3.Text = Convert.ToString(H1);  
            textBox4.Text = Convert.ToString(DR);  
            textBox5.Text = Convert.ToString(oR);  
            textBox6.Text = Convert.ToString(D2);  
            textBox7.Text = Convert.ToString(H2);  
            textBox8.Text = Convert.ToString(L1);  
            textBox9.Text = Convert.ToString(L2);  
            textBox10.Text = Convert.ToString(HB);  
            textBox11.Text = Convert.ToString(T);  
            textBox12.Text = Convert.ToString(D3);  
            textBox13.Text = Convert.ToString(H3);  
            textBox14.Text = Convert.ToString(H4);  
            textBox15.Text = Convert.ToString(D4D);  
            textBox16.Text = Convert.ToString(D4d);  
            textBox17.Text = Convert.ToString(A);  
            textBox18.Text = Convert.ToString(H5);  
            textBox19.Text = Convert.ToString(D51);  
            textBox20.Text = Convert.ToString(L51);  
            textBox21.Text = Convert.ToString(D52);  
            textBox22.Text = Convert.ToString(L52);  
            textBox23.Text = Convert.ToString(D53);  
            textBox24.Text = Convert.ToString(L53);  
            textBox25.Text = Convert.ToString(D54);  
            textBox26.Text = Convert.ToString(L54);  
        }  
        private Inventor.Application ThisApplication = null;  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>();  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>();  
        private Dictionary<string, PartComponentDefinition> oCompDef = new Dictionary<string, PartComponentDefinition>();  
        private Dictionary<string, TransientGeometry> oTransGeom = new Dictionary<string, TransientGeometry>();  
        double D1 = 440, H1 = 30, D2 = 350, H2 = 50, DR = 395, oR = 11; // Основание  
        double L1 = 480, L2 = 440, T = 15, HB = 250, D3 = 250, H3 = 90; // "Коробка"  
        double H4 = 400, D4D = 30, D4d = 20, A = 40; // Трубки  
        double H5 = 205, D51 = 200, L51 = 20, D52 = 245, L52 = 45, D53 = 100, L53 = 22.5, D54 = 122.5, L54 = 35; // Крышка  
        private void Build\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            oPartDoc["КВ"] = (PartDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kPartDocumentObject));  
            oCompDef["КВ"] = oPartDoc["КВ"].ComponentDefinition;  
            oTransGeom["КВ"] = ThisApplication.TransientGeometry;  
            oFileName["КВ"] = null;  
            oPartDoc["КВ"].DisplayName = "Корпус для выгрузки материалов";  
            Parameters oParameters = oCompDef["КВ"].Parameters;  
            SketchPoint[] point = new SketchPoint[10];  
            SketchLine[] line = new SketchLine[10];  
            SketchCircle[] circle = new SketchCircle[2];  
            // Основание  
            PlanarSketch oSketch = oCompDef["КВ"].Sketches.Add(oCompDef["КВ"].WorkPlanes[2]);  
            circle[0] = oSketch.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), D1 / 10 / 2);  
            Profile oProfile = (Profile)oSketch.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, H1 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile);  
            PlanarSketch oSketch1 = oCompDef["КВ"].Sketches.Add(oExtrudeDef.Faces[3], false);  
            circle[0] = oSketch1.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), D2 / 10 / 2);  
            Profile oProfile1 = (Profile)oSketch1.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef1 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile1, H2 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile1);  
            // Кубическая часть корпуса  
            PlanarSketch oSketch2 = oCompDef["КВ"].Sketches.Add(oExtrudeDef1.Faces[2], false);  
            point[0] = oSketch2.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(L2 / 10 / 2, L1 / 10 / 2), false);  
            point[1] = oSketch2.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[2] = oSketch2.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[3] = oSketch2.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, point[0].Geometry.Y), false);  
            line[0] = oSketch2.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch2.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch2.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch2.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile2 = (Profile)oSketch2.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef2 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile2, T / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile2);  
            PlanarSketch oSketch3 = oCompDef["КВ"].Sketches.Add(oExtrudeDef2.Faces[6], false);  
            point[0] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(L2 / 10 / 2, L1 / 10 / 2), false);  
            point[1] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[2] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[3] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, point[0].Geometry.Y), false);  
            line[0] = oSketch3.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch3.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch3.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch3.SketchLines.AddByTwoPoints(point[3], point[0]);  
            point[4] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X - T / 10, point[0].Geometry.Y - T / 10), false);  
            point[5] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X - T / 10, -point[0].Geometry.Y + T / 10), false);  
            point[6] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X + T / 10, -point[0].Geometry.Y + T / 10), false);  
            point[7] = oSketch3.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X + T / 10, point[0].Geometry.Y - T / 10), false);  
            line[4] = oSketch3.SketchLines.AddByTwoPoints(point[4], point[5]);  
            line[5] = oSketch3.SketchLines.AddByTwoPoints(point[5], point[6]);  
            line[6] = oSketch3.SketchLines.AddByTwoPoints(point[6], point[7]);  
            line[7] = oSketch3.SketchLines.AddByTwoPoints(point[7], point[4]);  
            Profile oProfile3 = (Profile)oSketch3.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef3 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile3, (HB - T \* 2) / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile3);  
            WorkPlane oWorkPlane4 = oCompDef["КВ"].WorkPlanes.AddByPlaneAndOffset(oCompDef["КВ"].WorkPlanes[2], (H1 + H2 + HB - T) / 10, false);  
            oWorkPlane4.Visible = false;  
            PlanarSketch oSketch4 = oCompDef["КВ"].Sketches.Add(oWorkPlane4);  
            point[0] = oSketch4.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(L2 / 10 / 2, L1 / 10 / 2), false);  
            point[1] = oSketch4.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[2] = oSketch4.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, -point[0].Geometry.Y), false);  
            point[3] = oSketch4.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(-point[0].Geometry.X, point[0].Geometry.Y), false);  
            line[0] = oSketch4.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch4.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch4.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch4.SketchLines.AddByTwoPoints(point[3], point[0]);  
            circle[0] = oSketch4.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), D3 / 2 / 10);  
            Profile oProfile4 = (Profile)oSketch4.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef4 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile4, T / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile4);  
            // Переход к коническому корпусу  
            PlanarSketch oSketch5 = oCompDef["КВ"].Sketches.Add(oExtrudeDef4.Faces[3], false);  
            circle[0] = oSketch5.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), D3 / 2 / 10);  
            circle[1] = oSketch5.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), (D3 / 2 + T) / 10);  
            Profile oProfile5 = (Profile)oSketch5.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef5 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile5, H3 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile5);  
            // Отверстия у основания  
            PlanarSketch oSketch6 = oCompDef["КВ"].Sketches.Add(oCompDef["КВ"].WorkPlanes[2]);  
            point[0] = oSketch6.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch6.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(0, DR / 2 / 10), false);  
            point[2] = oSketch6.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(DR / 2 / 10, 0), false);  
            line[0] = oSketch6.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch6.SketchLines.AddByTwoPoints(point[0], point[2]);  
            oSketch6.DimensionConstraints.AddTwoLineAngle(line[0], line[1], oTransGeom["КВ"].CreatePoint2d(1, 1));  
            oSketch6.GeometricConstraints.AddHorizontal((SketchEntity)line[1]);  
            circle[0] = oSketch6.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(DR / 10, -1), oR / 10);  
            oParameters["d13"].Expression = "105 degree";  
            oSketch6.GeometricConstraints.AddCoincident((SketchEntity)circle[0].CenterSketchPoint, (SketchEntity)point[1]);  
            Profile oProfile6 = (Profile)oSketch6.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef6 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile6, H1 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile6);  
            ObjectCollection objCollection6 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection6.Add(oExtrudeDef6);  
            CircularPatternFeature CircularPatternFeature6 = oCompDef["КВ"].Features.CircularPatternFeatures.Add(objCollection6, oCompDef["КВ"].WorkAxes[2], true, 12, "360 degree", true, PatternComputeTypeEnum.kIdenticalCompute);  
            // Трубки  
            PlanarSketch oSketch7 = oCompDef["КВ"].Sketches.Add(oCompDef["КВ"].WorkPlanes[1]);  
            circle[0] = oSketch7.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(H4 / 10, 0), D4D / 2 / 10);  
            Profile oProfile7 = (Profile)oSketch7.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef7 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile7, D3 / 10 + T \* 2 / 10 + A \* 2 / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile7);  
            PlanarSketch oSketch8 = oCompDef["КВ"].Sketches.Add(oCompDef["КВ"].WorkPlanes[1]);  
            circle[0] = oSketch8.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(H4 / 10, 0), D4d / 2 / 10);  
            Profile oProfile8 = (Profile)oSketch8.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef8 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile8, D3 / 10 + T \* 2 / 10 + A \* 2 / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile8);  
            PlanarSketch oSketch9 = oCompDef["КВ"].Sketches.Add(oExtrudeDef4.Faces[3], false);  
            circle[0] = oSketch9.SketchCircles.AddByCenterRadius(oTransGeom["КВ"].CreatePoint2d(0, 0), D3 / 2 / 10);  
            Profile oProfile9 = (Profile)oSketch9.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef9 = oCompDef["КВ"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile9, H3 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile9);  
            // Крышка  
            PlanarSketch oSketch10 = oCompDef["КВ"].Sketches.Add(oCompDef["КВ"].WorkPlanes[3]);  
            point[0] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(L2 / 2 / 10, H5 / 10), false);  
            point[1] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[0].Geometry.X, point[0].Geometry.Y + D51 / 2 / 10), false);  
            point[2] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[1].Geometry.X + L51 / 10, point[1].Geometry.Y), false);  
            point[3] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[2].Geometry.X, point[0].Geometry.Y + D52 / 2 / 10), false);  
            point[4] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[3].Geometry.X + L52 / 10, point[3].Geometry.Y), false);  
            point[5] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[4].Geometry.X, point[0].Geometry.Y + D53 / 2 / 10), false);  
            point[6] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[5].Geometry.X + L53 / 10, point[5].Geometry.Y), false);  
            point[7] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[6].Geometry.X, point[0].Geometry.Y + D54 / 2 / 10), false);  
            point[8] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[7].Geometry.X + L54 / 10, point[7].Geometry.Y), false);  
            point[9] = oSketch10.SketchPoints.Add(oTransGeom["КВ"].CreatePoint2d(point[8].Geometry.X, point[0].Geometry.Y), false);  
            line[0] = oSketch10.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch10.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch10.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch10.SketchLines.AddByTwoPoints(point[3], point[4]);  
            line[4] = oSketch10.SketchLines.AddByTwoPoints(point[4], point[5]);  
            line[5] = oSketch10.SketchLines.AddByTwoPoints(point[5], point[6]);  
            line[6] = oSketch10.SketchLines.AddByTwoPoints(point[6], point[7]);  
            line[7] = oSketch10.SketchLines.AddByTwoPoints(point[7], point[8]);  
            line[8] = oSketch10.SketchLines.AddByTwoPoints(point[8], point[9]);  
            line[9] = oSketch10.SketchLines.AddByTwoPoints(point[9], point[0]);  
            Profile oProfile10 = (Profile)oSketch10.Profiles.AddForSolid();  
            RevolveFeature revolvefeature10 = oCompDef["КВ"].Features.RevolveFeatures.AddFull(oProfile10, line[9], PartFeatureOperationEnum.kJoinOperation);  
            MessageBox.Show("Создание корпуса для выгрузки материалов завершено.", "Построение корпуса для выгрузки материалов");  
        }  
        private void Save\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            try  
            {  
                saveFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                saveFileDialog1.Title = "Сохранение корпуса для выгрузки материалов";  
                saveFileDialog1.FileName = oPartDoc["КВ"].DisplayName;  
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
                    if (!string.IsNullOrWhiteSpace(saveFileDialog1.FileName))  
                    {  
                        oPartDoc["КВ"].SaveAs(saveFileDialog1.FileName, false);  
                        oFileName["КВ"] = saveFileDialog1.FileName;  
                    }  
                Close();  
            }  
            catch  
            {  
                MessageBox.Show("Деталь ещё не создана, либо программа не видит созданную деталь.", "Построение корпуса для выгрузки материалов");  
            }  
        }  
        private void error\_check(System.Windows.Forms.TextBox textBox, double Number) // Проверка ошибки в textBox  
        {  
            try  
            {  
                if (Convert.ToDouble(textBox.Text) <= 0)  
                {  
                    MessageBox.Show("Введите значение больше нуля!", "Построение корпуса для выгрузки материалов");  
                    textBox.Text = Convert.ToString(Number);  
                }  
            }  
            catch  
            {  
                textBox.Text = Convert.ToString(Number);  
            }  
        }  
        private void textBox2\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox2, 440);  
            D1 = Convert.ToDouble(textBox2.Text);  
        }  
        private void textBox3\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox3, 30);  
            H1 = Convert.ToDouble(textBox3.Text);  
            textBox1.Text = Convert.ToString(H1 + H2 + HB + H3);  
        }  
        private void textBox4\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox4, 395);  
            DR = Convert.ToDouble(textBox4.Text);  
        }  
        private void textBox5\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox5, 11);  
            oR = Convert.ToDouble(textBox5.Text);  
        }  
        private void textBox6\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox6, 350);  
            D2 = Convert.ToDouble(textBox6.Text);  
        }  
        private void textBox7\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox7, 50);  
            H2 = Convert.ToDouble(textBox7.Text);  
            textBox1.Text = Convert.ToString(H1 + H2 + HB + H3);  
        }  
        private void textBox8\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox8, 480);  
            L1 = Convert.ToDouble(textBox8.Text);  
        }  
        private void textBox9\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox9, 440);  
            L2 = Convert.ToDouble(textBox9.Text);  
        }  
        private void textBox10\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox10, 250);  
            HB = Convert.ToDouble(textBox10.Text);  
            textBox1.Text = Convert.ToString(H1 + H2 + HB + H3);  
        }  
        private void textBox11\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox11, 15);  
            T = Convert.ToDouble(textBox11.Text);  
        }  
        private void textBox12\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox12, 250);  
            D3 = Convert.ToDouble(textBox12.Text);  
        }  
        private void textBox13\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox13, 90);  
            H3 = Convert.ToDouble(textBox13.Text);  
            textBox1.Text = Convert.ToString(H1 + H2 + HB + H3);  
        }  
        private void textBox14\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox14, 400);  
            H4 = Convert.ToDouble(textBox14.Text);  
        }  
        private void textBox15\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox15, 30);  
            D4D = Convert.ToDouble(textBox15.Text);  
        }  
        private void textBox16\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox16, 20);  
            D4d = Convert.ToDouble(textBox16.Text);  
        }  
        private void textBox17\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox17, 40);  
            A = Convert.ToDouble(textBox17.Text);  
        }  
        private void textBox18\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox18, 205);  
            H5 = Convert.ToDouble(textBox18.Text);  
        }  
        private void textBox19\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox19, 200);  
            D51 = Convert.ToDouble(textBox19.Text);  
        }  
        private void textBox20\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox20, 20);  
            L51 = Convert.ToDouble(textBox20.Text);  
        }  
        private void textBox21\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox21, 245);  
            D52 = Convert.ToDouble(textBox21.Text);  
        }  
        private void textBox22\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox22, 45);  
            L52 = Convert.ToDouble(textBox22.Text);  
        }  
        private void textBox23\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox23, 100);  
            D53 = Convert.ToDouble(textBox23.Text);  
        }  
        private void textBox24\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox24, 22.5);  
            L53 = Convert.ToDouble(textBox24.Text);  
        }  
        private void textBox25\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox25, 122.5);  
            D54 = Convert.ToDouble(textBox25.Text);  
        }  
        private void textBox26\_TextChanged(object sender, EventArgs e)  
        {  
            error\_check(textBox26, 35);  
            L54 = Convert.ToDouble(textBox26.Text);  
        }  
    }

public partial class Hold : Form  
    {  
        public Hold()  
        {  
            InitializeComponent();  
            this.comboBox1.Items.AddRange(new object[] { 100, 160, 255 });  
            comboBox1.SelectedIndex = 0;  
        }  
        private Inventor.Application ThisApplication = null;  
        private Dictionary<string, PartDocument> oPartDoc = new Dictionary<string, PartDocument>();  
        private Dictionary<string, string> oFileName = new Dictionary<string, string>();  
        private Dictionary<string, PartComponentDefinition> oCompDef = new Dictionary<string, PartComponentDefinition>();  
        private Dictionary<string, TransientGeometry> oTransGeom = new Dictionary<string, TransientGeometry>();  
        double b = 100, a1 = 65, h = 120, S1 = 4, a = 45, K = 10, K1 = 25, b1 = 50, b2 = 45, c = 40, d6 = 12, a2 = 40, h1 = 8;  
        private void Build\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            oPartDoc["БО"] = (PartDocument)ThisApplication.Documents.Add(DocumentTypeEnum.kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(DocumentTypeEnum.kPartDocumentObject));  
            oCompDef["БО"] = oPartDoc["БО"].ComponentDefinition;  
            oTransGeom["БО"] = ThisApplication.TransientGeometry;  
            oFileName["БО"] = null;  
            oPartDoc["БО"].DisplayName = "Боковые опоры";  
            Parameters oParameters = oCompDef["БО"].Parameters; // Хранит параметры детали  
            SketchPoint[] point = new SketchPoint[5];  
            SketchLine[] line = new SketchLine[5];  
            SketchCircle[] circle = new SketchCircle[1];  
            // Нижняя основа  
            PlanarSketch oSketch = oCompDef["БО"].Sketches.Add(oCompDef["БО"].WorkPlanes[3]);  
            point[0] = oSketch.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(b / 10, 0), false);  
            point[2] = oSketch.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(b / 10, -S1 / 10), false);  
            point[3] = oSketch.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(0, -S1 / 10), false);  
            line[0] = oSketch.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile = (Profile)oSketch.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef = oCompDef["БО"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, a1 / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile);  
            // Боковые крепления  
            WorkPlane oWorkPlane1 = oCompDef["БО"].WorkPlanes.AddByPlaneAndOffset(oCompDef["БО"].WorkPlanes[3], a / 2 / 10, false);  
            oWorkPlane1.Visible = false;  
            PlanarSketch oSketch1 = oCompDef["БО"].Sketches.Add(oWorkPlane1);  
            point[0] = oSketch1.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(0, 0), false);  
            point[1] = oSketch1.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d((b \* Math.Sin(MainBody.Degree / 180 \* Math.PI)) / 10, h \* Math.Cos(MainBody.Degree / 180 \* Math.PI) / 10 - S1 / 10), false);  
            point[2] = oSketch1.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(point[1].Geometry.X + K / 10, point[1].Geometry.Y), false);  
            point[3] = oSketch1.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(b / 10, K1 / 10), false);  
            point[4] = oSketch1.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(b / 10, 0), false);  
            line[0] = oSketch1.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch1.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch1.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch1.SketchLines.AddByTwoPoints(point[3], point[4]);  
            line[4] = oSketch1.SketchLines.AddByTwoPoints(point[4], point[0]);  
            Profile oProfile1 = (Profile)oSketch1.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef1 = oCompDef["БО"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile1, S1 / 10, PartFeatureExtentDirectionEnum.kPositiveExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile1);  
            ObjectCollection objCollection1 = ThisApplication.TransientObjects.CreateObjectCollection();  
            objCollection1.Add(oExtrudeDef1);  
            oCompDef["БО"].Features.MirrorFeatures.AddByDefinition(oCompDef["БО"].Features.MirrorFeatures.CreateDefinition(objCollection1, oCompDef["БО"].WorkPlanes[3], PatternComputeTypeEnum.kIdenticalCompute));  
            // Болтовое отверстие  
            PlanarSketch oSketch2 = oCompDef["БО"].Sketches.Add(oCompDef["БО"].WorkPlanes[3]);  
            point[0] = oSketch2.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(b / 10 - b1 / 10, -S1 / 10), false);  
            point[1] = oSketch2.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(point[0].Geometry.X + b2 / 10, point[0].Geometry.Y), false);  
            point[2] = oSketch2.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(point[1].Geometry.X, point[0].Geometry.Y - h1 / 10), false);  
            point[3] = oSketch2.SketchPoints.Add(oTransGeom["БО"].CreatePoint2d(point[0].Geometry.X, point[2].Geometry.Y), false);  
            line[0] = oSketch2.SketchLines.AddByTwoPoints(point[0], point[1]);  
            line[1] = oSketch2.SketchLines.AddByTwoPoints(point[1], point[2]);  
            line[2] = oSketch2.SketchLines.AddByTwoPoints(point[2], point[3]);  
            line[3] = oSketch2.SketchLines.AddByTwoPoints(point[3], point[0]);  
            Profile oProfile2 = (Profile)oSketch2.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef2 = oCompDef["БО"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile2, a2 / 10, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kJoinOperation, oProfile2);  
            PlanarSketch oSketch3 = oCompDef["БО"].Sketches.Add(oCompDef["БО"].WorkPlanes[2]);  
            circle[0] = oSketch3.SketchCircles.AddByCenterRadius(oTransGeom["БО"].CreatePoint2d(-(b / 10 - c / 10), 0), d6 / 2 / 10);  
            Profile oProfile3 = (Profile)oSketch3.Profiles.AddForSolid();  
            ExtrudeFeature oExtrudeDef3 = oCompDef["БО"].Features.ExtrudeFeatures.AddByDistanceExtent(oProfile3, h, PartFeatureExtentDirectionEnum.kSymmetricExtentDirection, PartFeatureOperationEnum.kCutOperation, oProfile3);  
            // Резьба  
            EdgeCollection EdgeCollection1 = ThisApplication.TransientObjects.CreateEdgeCollection();  
            EdgeCollection1.Add(oExtrudeDef3.SideFaces[1].Edges[1]);  
            ThreadFeatures ThreadFeatures1 = oCompDef["БО"].Features.ThreadFeatures;  
            StandardThreadInfo stInfo1 = ThreadFeatures1.CreateStandardThreadInfo(false, true, "ISO Metric profile", "M" + d6 + "x1.5", "6g");  
            ThreadFeatures1.Add(oExtrudeDef3.SideFaces[1], oExtrudeDef3.SideFaces[1].Edges[2], (ThreadInfo)stInfo1, false, true, 0);  
            MessageBox.Show("Создание боковых опор завершено.", "Построение боковых опор");  
        }  
        private void Save\_Click(object sender, EventArgs e)  
        {  
            try  
            {  
                ThisApplication = (Inventor.Application)System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application");  
            }  
            catch  
            {  
                InventorControl IC = new InventorControl();  
                IC.ShowDialog();  
                return;  
            }  
            try  
            {  
                saveFileDialog1.Filter = "Inventor Part Document|\*.ipt";  
                saveFileDialog1.Title = "Сохранение боковых опор";  
                saveFileDialog1.FileName = oPartDoc["БО"].DisplayName;  
                if (saveFileDialog1.ShowDialog() == DialogResult.OK)  
                    if (!string.IsNullOrWhiteSpace(saveFileDialog1.FileName))  
                    {  
                        oPartDoc["БО"].SaveAs(saveFileDialog1.FileName, false);  
                        oFileName["БО"] = saveFileDialog1.FileName;  
                    }  
                Close();  
            }  
            catch  
            {  
                MessageBox.Show("Деталь ещё не создана, либо программа не видит созданную деталь.", "Построение боковых опор");  
            }  
        }  
        private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)  
        {  
            switch (comboBox1.SelectedIndex)  
            {  
                case 0: b = 100; a1 = 65; h = 120; S1 = 4; a = 45; K = 10; K1 = 25; b1 = 50; b2 = 45; c = 40; d6 = 12; a2 = 40; h1 = 8; break;  
                case 1: b = 160; a1 = 95; h = 190; S1 = 5; a = 75; K = 15; K1 = 40; b1 = 70; b2 = 65; c = 50; d6 = 12; a2 = 60; h1 = 10; break;  
                case 2: b = 255; a1 = 155; h = 310; S1 = 8; a = 125; K = 25; K1 = 65; b1 = 120; b2 = 115; c = 90; d6 = 20; a2 = 100; h1 = 16; break;  
            }  
            textBox1.Text = Convert.ToString(a1);  
            textBox2.Text = Convert.ToString(h);  
            textBox3.Text = Convert.ToString(S1);  
            textBox4.Text = Convert.ToString(a);  
            textBox5.Text = Convert.ToString(K);  
            textBox6.Text = Convert.ToString(K1);  
            textBox7.Text = Convert.ToString(b1);  
            textBox8.Text = Convert.ToString(b2);  
            textBox9.Text = Convert.ToString(c);  
            textBox10.Text = Convert.ToString(d6);  
            textBox11.Text = Convert.ToString(a2);  
            textBox12.Text = Convert.ToString(h1);  
        }  
    }  
}