**ПРИМЕР №4**

Задача: реализация 3D модели гайки.

**Поэтапное построение эскиза.**

1. Создаем эскиз шестиугольника, описанного около окружности:

void CreateSketch2(RPArray<MbContour>& \_arrContours)

{

// Центры и радиусы окружностей, дуги которых входят в контур

const MbCartPoint centerCircle(0, 0);

const double RAD = 20;

// Количество сторон многоугольника

const int SIDE\_CNT = 6;

// Радиус описывающей окружности многоугольника

const double RAD1 = 45.0;

// Массив для хранения вершин ломаной

SArray<MbCartPoint> arrPnts(SIDE\_CNT);

// Вычисление вершин ломаной равномерным делением окружности

for (int i = 0; i < SIDE\_CNT; i++)

{

// Угловое положение i-й вершины на описывающей окружности.

// Угловое положение начальной вершины - M\_PI/2 (эта вершина

// расположена на вертикальной оси).

double angle = M\_PI / 2 + 2 \* M\_PI / SIDE\_CNT \* i;

MbCartPoint pnt(RAD1 \* cos(angle), RAD1 \* sin(angle));

arrPnts.Add(pnt);

}

// Построение единой ломаной внешнего контура по точкам

MbPolyline\* pPolyline = new MbPolyline(arrPnts, true);

MbContour\* pContour = new MbContour(\*pPolyline, true);

// Построение окружности

MbArc\* pCircle = new MbArc(centerCircle, RAD);

MbContour\* pContourCircle = new MbContour(\*pCircle, true);

\_arrContours.push\_back(pContour);

\_arrContours.push\_back(pContourCircle);

}

1. Построение эскиза резьбы:

void CreateSketch01(RPArray<MbContour>& \_arrContours)

{

// Параметры для эскиза

double R = 18;

double l = 2;

// Массив точек для массив

SArray<MbCartPoint>arrPnts(4);

arrPnts.Add(MbCartPoint(-l \* 0.1 + l, -l + R));

arrPnts.Add(MbCartPoint(l \* 0.1 + l, -l + R));

arrPnts.Add(MbCartPoint(2\*l, l + R));

arrPnts.Add(MbCartPoint(0, l + R));

// Создание единого контура

MbLineSegment\* pS1 = new MbLineSegment(arrPnts[0], arrPnts[1]);

MbContour\* pContour = new MbContour(\*pS1, true);

MbLineSegment\* pS2 = new MbLineSegment(arrPnts[1], arrPnts[2]);

pContour->AddSegment(pS2);

MbLineSegment\* pS3 = new MbLineSegment(arrPnts[2], arrPnts[3]);

pContour->AddSegment(pS3);

MbLineSegment\* pS4 = new MbLineSegment(arrPnts[3], arrPnts[0]);

pContour->AddSegment(pS4);

\_arrContours.push\_back(pContour);

}

1. Переходим к основному коду:

void MakeUserCommand0()

{

// Локальная СК (по умолчанию совпадает с мировой СК)

MbPlacement3D pl;

// СОЗДАНИЕ КОНТУРОВ ДЛЯ ОБРАЗУЮЩЕЙ

RPArray<MbContour> arrContours;

CreateSketch2(arrContours);

// Отображение образующей (в плоскости XY глобальной СК)

for (int i = 0; i < arrContours.size(); i++)

viewManager->AddObject(Style(1, RGB(0, 0, 255)), arrContours[i], &pl);

// ПОСТРОЕНИЕ ТЕЛА ВЫДАВЛИВАНИЯ

// Образующая размещается на плоскости XY глобальной СК.

// Важное замечание: объект-плоскость должен создаваться динамически,

// поскольку он продолжает использоваться в объекте-твердом теле после

// выхода из данной функции.

MbPlane\* pPlaneXY = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(1, 0, 0),

MbCartPoint3D(0, 1, 0));

// Объект, хранящий параметры образующей

MbSweptData sweptData(\*pPlaneXY, arrContours);

// Направляющий вектор для операции выдавливания

MbVector3D dir(0, 0, -1);

// Параметры операции выдавливания, задающие свойства тела для построения:

// расстояние выдавливания в прямом и в обратном направлении вдоль

// направляющего вектора

const double HEIGHT\_FORWARD = 30.0, HEIGHT\_BACKWARD = 0.0;

ExtrusionValues extrusionParam(HEIGHT\_FORWARD, HEIGHT\_BACKWARD);

// Служебный объект для именования элементов модели твердого тела

MbSNameMaker operNames(ct\_CurveExtrusionSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> cNames(0, 1, false);

// Построение твердого тела выдавливания

MbSolid\* pSolid = NULL;

MbResultType res = ::ExtrusionSolid(sweptData, dir, NULL, NULL, false,

extrusionParam, operNames, cNames, pSolid);

// Локальная СК (по умолчанию совпадает с мировой СК)

MbPlacement3D pl1;

// Построение направляющей кривой в виде незамкнутого NURBS-сплайна

// 4-го порядка по контрольным точкам

MbConeSpiral\* pSpiral = new MbConeSpiral(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, 0, -26), MbCartPoint3D(0, 26, 0), 4, false);

// Описание образующей кривой в виде плоского контура на плоскости XY мировой СК

MbPlane\* pPlaneXY1 = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, 0, -50),

MbCartPoint3D(0, 50, 0));

// Построение образующей кривой с помощью вспомогательной функции CreateSketch

RPArray<MbContour> arrContours1;

CreateSketch01(arrContours1);

MbSweptData sweptData1(\*pPlaneXY1, arrContours1);

// Объект с параметрами операции построения тела заметания

EvolutionValues params;

// Вариант плоскопараллельного движения образующей вдоль направляющей

params.parallel = 1;

// Служебные объекты-именователи для вызова геометрической операции

MbSNameMaker operNames1(ct\_CurveEvolutionSolid, MbSNameMaker::i\_SideNone, 0);

MbSNameMaker cNames1(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> contourNames(1, 0, false);

contourNames.Add(&cNames1);

MbSNameMaker splineNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

// Вызов операции построения тела заметания

MbSolid\* pSolid1 = NULL;

MbResultType res1 = ::EvolutionSolid(sweptData1, \*pSpiral, params, operNames1,

contourNames, splineNames, pSolid1);

// Отображение построенного тела

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid1);

if (res1 == rt\_Success)

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid1);

// Отображение построенного тела

if (res == rt\_Success)

{

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid);

}

// Уменьшение счетчиков ссылок динамически созданных объектов ядра

::DeleteItem(pSolid);

::DeleteItem(pPlaneXY);

for (int i = 0; i < arrContours.size(); i++)

::DeleteItem(arrContours[i]);

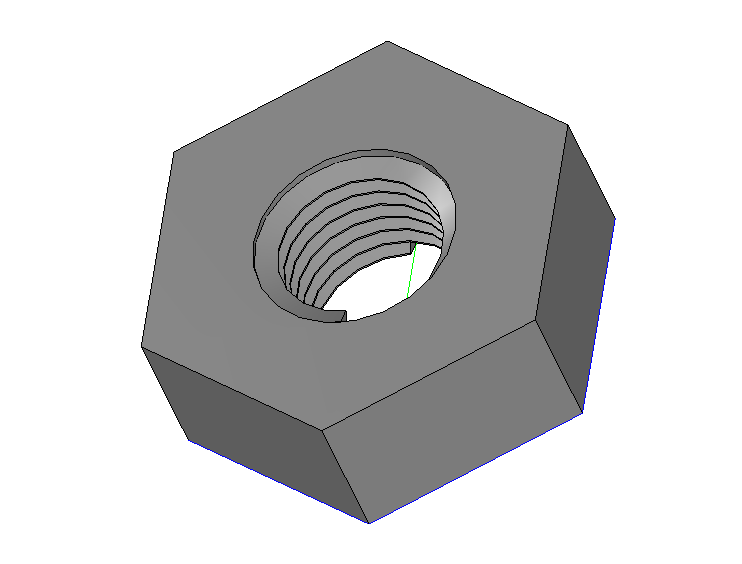
::DeleteItem(pSolid1);

::DeleteItem(pPlaneXY1);

::DeleteItem(pSpiral);

}

**Результат построения.**



**Код программы.**

void CreateSketch2(RPArray<MbContour>& \_arrContours)

{

// Центры и радиусы окружностей, дуги которых входят в контур

const MbCartPoint centerCircle(0, 0);

const double RAD = 20;

// Количество сторон многоугольника

const int SIDE\_CNT = 6;

// Радиус описывающей окружности многоугольника

const double RAD1 = 45.0;

// Массив для хранения вершин ломаной

SArray<MbCartPoint> arrPnts(SIDE\_CNT);

// Вычисление вершин ломаной равномерным делением окружности

for (int i = 0; i < SIDE\_CNT; i++)

{

// Угловое положение i-й вершины на описывающей окружности.

// Угловое положение начальной вершины - M\_PI/2 (эта вершина

// расположена на вертикальной оси).

double angle = M\_PI / 2 + 2 \* M\_PI / SIDE\_CNT \* i;

MbCartPoint pnt(RAD1 \* cos(angle), RAD1 \* sin(angle));

arrPnts.Add(pnt);

}

// Построение единой ломаной внешнего контура по точкам

MbPolyline\* pPolyline = new MbPolyline(arrPnts, true);

MbContour\* pContour = new MbContour(\*pPolyline, true);

MbArc\* pCircle = new MbArc(centerCircle, RAD);

MbContour\* pContourCircle = new MbContour(\*pCircle, true);

\_arrContours.push\_back(pContour);

\_arrContours.push\_back(pContourCircle);

}

void CreateSketch01(RPArray<MbContour>& \_arrContours)

{

double R = 18;

double l = 2;

SArray<MbCartPoint>arrPnts(4);

arrPnts.Add(MbCartPoint(-l \* 0.1 + l, -l + R));

arrPnts.Add(MbCartPoint(l \* 0.1 + l, -l + R));

arrPnts.Add(MbCartPoint(2\*l, l + R));

arrPnts.Add(MbCartPoint(0, l + R));

MbLineSegment\* pS1 = new MbLineSegment(arrPnts[0], arrPnts[1]);

MbContour\* pContour = new MbContour(\*pS1, true);

MbLineSegment\* pS2 = new MbLineSegment(arrPnts[1], arrPnts[2]);

pContour->AddSegment(pS2);

MbLineSegment\* pS3 = new MbLineSegment(arrPnts[2], arrPnts[3]);

pContour->AddSegment(pS3);

MbLineSegment\* pS4 = new MbLineSegment(arrPnts[3], arrPnts[0]);

pContour->AddSegment(pS4);

\_arrContours.push\_back(pContour);

}

void MakeUserCommand0()

{

MbPlacement3D pl; // Локальная СК (по умолчанию совпадает с мировой СК)

// СОЗДАНИЕ КОНТУРОВ ДЛЯ ОБРАЗУЮЩЕЙ

RPArray<MbContour> arrContours;

CreateSketch2(arrContours);

// Отображение образующей (в плоскости XY глобальной СК)

for (int i = 0; i < arrContours.size(); i++)

viewManager->AddObject(Style(1, RGB(0, 0, 255)), arrContours[i], &pl);

// ПОСТРОЕНИЕ ТЕЛА ВЫДАВЛИВАНИЯ

// Образующая размещается на плоскости XY глобальной СК.

// Важное замечание: объект-плоскость должен создаваться динамически,

// поскольку он продолжает использоваться в объекте-твердом теле после

// выхода из данной функции.

MbPlane\* pPlaneXY = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(1, 0, 0),

MbCartPoint3D(0, 1, 0));

// Объект, хранящий параметры образующей

MbSweptData sweptData(\*pPlaneXY, arrContours);

// Направляющий вектор для операции выдавливания

MbVector3D dir(0, 0, -1);

// Параметры операции выдавливания, задающие свойства тела для построения:

// расстояние выдавливания в прямом и в обратном направлении вдоль

// направляющего вектора

const double HEIGHT\_FORWARD = 30.0, HEIGHT\_BACKWARD = 0.0;

ExtrusionValues extrusionParam(HEIGHT\_FORWARD, HEIGHT\_BACKWARD);

// Служебный объект для именования элементов модели твердого тела

MbSNameMaker operNames(ct\_CurveExtrusionSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> cNames(0, 1, false);

// Построение твердого тела выдавливания

MbSolid\* pSolid = NULL;

MbResultType res = ::ExtrusionSolid(sweptData, dir, NULL, NULL, false,

extrusionParam, operNames, cNames, pSolid);

// Локальная СК (по умолчанию совпадает с мировой СК)

MbPlacement3D pl1;

// Построение направляющей кривой в виде незамкнутого NURBS-сплайна

// 4-го порядка по контрольным точкам

MbConeSpiral\* pSpiral = new MbConeSpiral(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, 0, -26),

MbCartPoint3D(0, 26, 0), 4, false);

// Описание образующей кривой в виде плоского контура на плоскости XY мировой СК

MbPlane\* pPlaneXY1 = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, 0, -50),

MbCartPoint3D(0, 50, 0));

// Построение образующей кривой с помощью вспомогательной функции CreateSketch

RPArray<MbContour> arrContours1;

CreateSketch01(arrContours1);

MbSweptData sweptData1(\*pPlaneXY1, arrContours1);

// Объект с параметрами операции построения тела заметания

EvolutionValues params;

// Вариант плоскопараллельного движения образующей вдоль направляющей

params.parallel = 1;

// Служебные объекты-именователи для вызова геометрической операции

MbSNameMaker operNames1(ct\_CurveEvolutionSolid, MbSNameMaker::i\_SideNone, 0);

MbSNameMaker cNames1(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> contourNames(1, 0, false);

contourNames.Add(&cNames1);

MbSNameMaker splineNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

// Вызов операции построения тела заметания

MbSolid\* pSolid1 = NULL;

MbResultType res1 = ::EvolutionSolid(sweptData1, \*pSpiral, params, operNames1,

contourNames, splineNames, pSolid1);

// Отображение построенного тела

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid1);

if (res1 == rt\_Success)

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid1);

// Отображение построенного тела

if (res == rt\_Success)

{

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid);

}

// Уменьшение счетчиков ссылок динамически созданных объектов ядра

::DeleteItem(pSolid);

::DeleteItem(pPlaneXY);

for (int i = 0; i < arrContours.size(); i++)

::DeleteItem(arrContours[i]);

::DeleteItem(pSolid1);

::DeleteItem(pPlaneXY1);

::DeleteItem(pSpiral);

}