Санкт-Петербургский Государственный Электротехнический

Университет «ЛЭТИ»

Кафедра МО ЭВМ

**ОТЧЕТ**

**Лабораторная работа №3**

**Построение классов с использованием простого**

**и множественного наследования**

Выполнил Монько А.О.

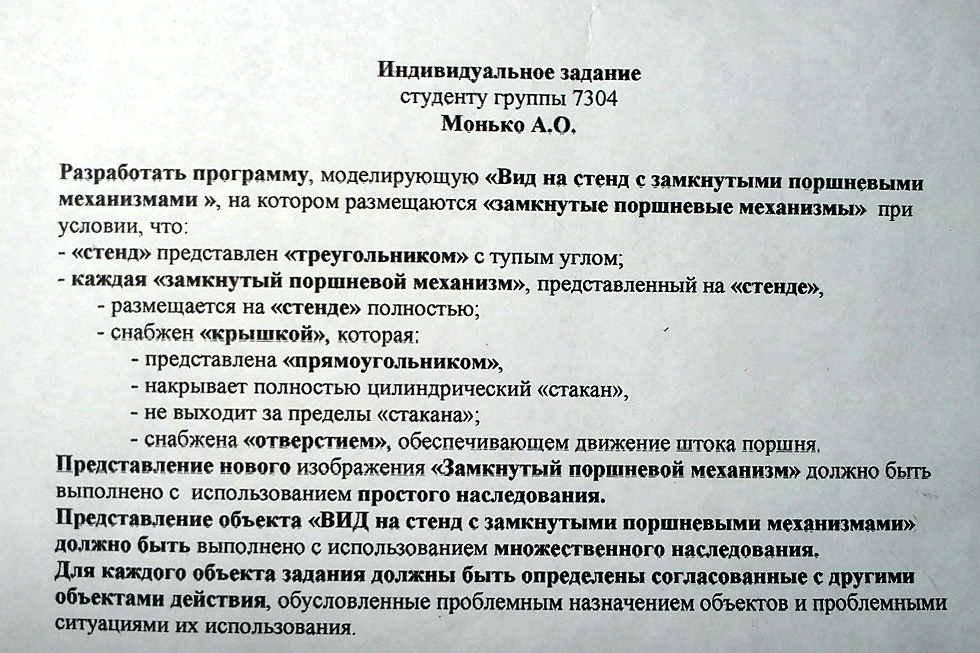
Группа 7304

Преподаватель Смольянинов А.В.

Санкт-Петербург

2010 г.

**Задание**



**Объектно-ориентированный анализ задания**

Исходя из постановка задачи нам требуется разработать новый объект - «Художественная кость»(ArtBone), который будет использовать простое наследование. Базовым классом для ArtBone будет класс Bone, а «Окантовка»(Edge), которая представлена в форме прямоугольника, будет членом данных.

Также по задании требуется реализовать «Вид на столешницу с художественными костями домино», используя множественное наследование. Добавим класс «Столешница»(TableTop), который представляет собой ромб, острые углы которого не меньше 30 градусов. Базовыми классами для «Вида на столешницу» будут «Столешница» и «Набор костей». Причем набор костей теперь может включать в себя как «Художественные» так и «обычные» кости.

## 1.2. Построение системы новых объектов

#### Введем множества:



#### Состав и свойства атрибутов и методов каждого объекта:

**Объект "Крышка" (class CCover)**

Атрибуты:

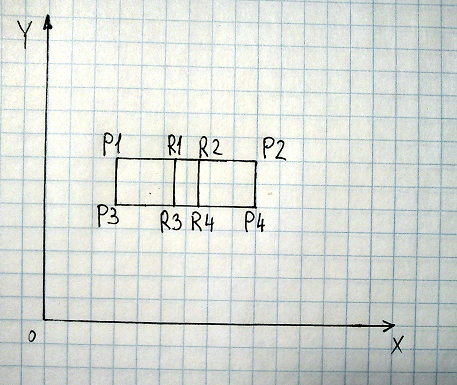
1. Отверстие - Прямоугольник(СRectangle Hole)

2. Крышка - Прямоугольник (CRectangle Cover)

Инвариант класса «Крышка»:

{(Cover.inv() && Hole.inv() && (Cover.GetHeight() == Hole.GetHeight())

&& (Hole.GetWidth() < Cover.GetWidth()) && (Hole.GetWidth() / 2 <= 1))}



Методы:

1. Плоскопараллельное перемещение «крышки» на (dx, dy):

Cover.SetP1(Cover.GetP1().GetX() + dx,Cover.GetP1().GetY() + dy);

Hole.SetP1(Hole.GetP1().GetX() + dx,Hole.GetP1().GetY() + dy);

Предусловие: (dyY) & (dxX) & (CCover.inv())

Постусловие: CCover.inv()

2. Помещение «точки» P1 в некоторую заданную точку P0(x0,y0):

d =Расстояние от точки P1 до точки R1;

Cover.SetP1(P0);

Hole.SetP1(Cover.GetP1().GetX() + d, Cover.GetP1().GetY());

Предусловие: (y0Y) & (x0 X) & (CCover.inv())

Постусловие: (CCover.inv())

**Объект "Тупоугольный треугольник" (class CTriangle)**

Атрибуты

1. P1,P2,P3 - Точки (CPoint);

2. P1P2,P2P3,P3P1(Длины соответствующих сторон) – Вещественный;

Инвариант класса«Тупоугольный треугольник»:

((P1>= -1000 && P1 <= 1000) && (P2>= -1000 && P2 <= 1000) &&

(P3>= -1000 && P3 <= 1000) && (P1.GetY() == P2.GetY()) && (fabs((P1.GetY() - P2.GetY())\*(P1.GetY() - P3.GetY()) + (P2.GetX() -P1.GetX())\*(P3.GetX() - P1.GetX())) != 0))

Y

X

P1

P3

P2

P1P2 || OX

Методы:

1. Плоскопараллельное перемещение «прямоугольника» на (dx,dy):

P1.SetX(P1.GetX() + dx);

P1.SetY(P1.GetY() + dy);

P2.SetX(P2.GetX() + dx);

P2.SetY(P2.GetY() + dy);

P3.SetX(P3.GetX() + dx);

P3.SetY(P3.GetY() + dy);

Предусловие: {(dyY) & (dxX) CTriangle.inv()}

Постусловие: { CTriangle.inv()}

2.Помещение точки P1 в некоторую заданную точку P0(x0,y0), при этом остальные точки пересчитываются :

P1 = P0;

P2.x = P0.x + P1P2

P2.y = P0.y

P3.x = 

P3.y = ****

Предусловие: {(y0Y) & (x0X) & CTriangle.inv()}

Постусловие: {CTriangle.inv()}

**Объект "Стенд" (class CStand)**

Атрибуты:

1. Стенд - Тупоугольный треугольник (CTriangle Stand); 2. Список характерных точек;

Инвариант класса «Стенд»:

{ (Stand.inv() && MainPoints.GetElemByNum(0)->GetValue() == Stand.GetP1()

&& MainPoints.GetElemByNum(1)->GetValue() == Stand.GetP2()

&& MainPoints.GetElemByNum(2)->GetValue() == Stand.GetP3())}

Методы:

1. Плоскопараллельное перемещение «прямоугольника» на (dx,dy):

P1.SetX(P1.GetX() + dx);

P1.SetY(P1.GetY() + dy);

P2.SetX(P2.GetX() + dx);

P2.SetY(P2.GetY() + dy);

P3.SetX(P3.GetX() + dx);

P3.SetY(P3.GetY() + dy);

Предусловие: {(dyY) & (dxX) CTriangle.inv()}

Постусловие: { CTriangle.inv()}

2.Помещение точки P1 в некоторую заданную точку P0(x0,y0), при этом остальные точки пересчитываются :

P1 = P0;

P2.x = P0.x + P1P2

P2.y = P0.y

P3.x = 

P3.y = ****

Предусловие: {(y0Y) & (x0X) & CTriangle.inv()}

Постусловие: {CTriangle.inv()}

**Объект "Замкнутый поршневой механизм" (class ClosedPistonMechanism)**

Атрибуты

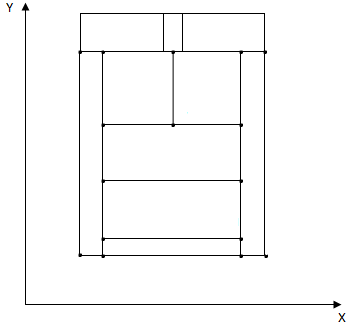
1. Крышка - CCover Cover;

Инвариант класса: (PistonMechanism.inv() && Cover.inv())

Поршневой механизм

Замкнутый поршневой механизм

Класс "Поршневой механизм" описан в отчете к лабораторной работе №2



Крышка

Поршень

Основание стакана

Левая часть стакана

Правая часть стакана

Шток

Методы:

1. Плоскопараллельное перемещение «Закрытого поршневого механизма» на (dx,dy):

Предусловие: {(dyY) & (dxX) & (CClosedPistonMechanism.Inv)}

Постусловие: {(CClosedPistonMechanism.Inv)}

2. Помещение «поршневого механизма» в некоторую заданную точку A0(x0,y0), при этом остальные точки вычисляются.

Предусловие: {(y0Y) & (x0X) & (CClosedPistonMechanism.Inv)}

Постусловие: {(CClosedPistonMechanism.Inv)}

3. Перемещение «поршня» в «поршневом стакане» на dy.

Предусловие: {(dyY) & (CClosedPistonMechanism.Inv)}

Постусловие: {(CClosedPistonMechanism.Inv)}

**Объект "Вид на стенд с закрытыми поршневыми механизмами" (class CView)**

Атрибуты

Набор поршневых механизмов

Стенд

Вид на стенд с механизмами

Методы:

1. Плоскопараллельное перемещение «набора» и «стенда» на (dx,dy):

moveStandAndColl(double dx, double dy)

Перемещаем весь набор и стенд на (dx,dy),

2. Переместить «набор» и «окно» в точку P0(x0, y0):

SetStandAndCollP1(CPint P0)

Перемещаем весь набор и окно в (x0,y0)

Количественные характеристики программы на C++:

Общее количество классов в программе: 14

Количество новых классов: 5

Количество измененных классов: 1

Количество классов без изменения: 8

Количество файлов на классы: 28

Общее количество файлов: 29

Общее количество строк: 2381

Разработанные классы на C++:

**Класс «Тупоугольный треугольник»**

class CTriangle {

private:

CPoint P1, P2, P3;

double P1P2, P2P3, P3P1;

static unsigned int debug;

static unsigned int total;

unsigned int id;

static unsigned int current;

public:

CTriangle(CPoint \_P1,double \_P1P2,CPoint \_P3); // P1 length(p1,p2) P3

CTriangle(); // Constructor without parametrs

~CTriangle();

int inv (); // Инвариант

void SetP1(double,double);

void SetP1(CPoint); // Задание точки P1

void Move (double,double); // Плоскопараллельное перемещение на dx dy

CPoint GetP1() const;

CPoint GetP2() const;

CPoint GetP3() const;

double GetP1P2() const;

double GetP2P3() const;

double GetP3P1() const;

int isIn (CPoint P) const; // Проверка принадлежности триугольнику

void Print ();

void SetDebug (unsigned int);

unsigned int GetID ();

unsigned int GetCurrent();

unsigned int GetTotal();

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CTria.h"

unsigned int CTriangle :: debug = 0;

unsigned int CTriangle :: total = 0;

unsigned int CTriangle :: current = 0;

CTriangle :: CTriangle (CPoint \_P1, double \_P1P2, CPoint \_P3) {

P1 = \_P1;

P3 = \_P3;

if (P3.nCheck(P1, CPoint(P1.GetX(), P1.GetY() + 1)) > 0) {

P2.SetX(P1.GetX() + \_P1P2);

P2.SetY(P1.GetY());

}

else {

P2.SetX(P1.GetX() - \_P1P2);

P2.SetY(P1.GetY());

}

P1P2 = \_P1P2;

P2P3 = P2.d(P3);

P3P1 = P3.d(P1);

id = ++total;

++current;

if (debug) cout << "Triangle " << P1 << "," << P2 << "," << P3 << " created" << endl;

}

CTriangle :: CTriangle () {

P1 = CPoint(0,0);

P3 = CPoint(-5,-5);

double \_P1P2 = 6;

if (P3.nCheck(P1, CPoint(P1.GetX(), P1.GetY() + 1)) > 0) {

P2.SetX(P1.GetX() + \_P1P2);

P2.SetY(P1.GetY());

}

else {

P2.SetX(P1.GetX() - \_P1P2);

P2.SetY(P1.GetY());

}

P1P2 = \_P1P2;

P2P3 = P2.d(P3);

P3P1 = P3.d(P1);

id = ++total;

++current;

if (debug) cout << "Triangle " << P1 << "," << P2 << "," << P3 << " created" << endl;

}

CTriangle :: ~CTriangle () {

if (debug) cout << "Triangle " << P1 << "," << P2 << "," << P3 << " deleted" << endl;

--current;

}

int CTriangle :: inv () {

return ((P1>= -1000 && P1 <= 1000) && (P2>= -1000 && P2 <= 1000) &&

(P3>= -1000 && P3 <= 1000) && (P1.GetY() == P2.GetY()) &&

(fabs((P1.GetY() - P2.GetY())\*(P1.GetY() - P3.GetY()) + (P2.GetX() - P1.GetX())\*(P3.GetX() - P1.GetX())) != 0));

}

void CTriangle :: SetP1(double x, double y) {

if (P1.GetY() >= P3.GetY()) {

P1 = CPoint(x,y);

P2 = CPoint(P1.GetX() + P1P2, P1.GetY());

P3.SetX(-0.5\*((pow(P3P1,2) - pow(P2P3,2) - pow(P1.GetX(),2) + pow(P2.GetX(),2)) / (P1.GetX() - P2.GetX())));

P3.SetY(( (P1.GetY() \* P1.GetX()) - (P1.GetY() \* P2.GetX()) +

sqrt( (2 \* P3P1 \* P3P1 \* P1.GetX() \* P1.GetX() - 4 \* P2.GetX() \* P1.GetX() \* P3P1 \* P3P1 - 4 \* P2.GetX() \* P1.GetX() \* P2P3 \* P2P3 + 4 \* pow(P2.GetX(),3) \* P1.GetX() + 4 \* P2.GetX() \* pow(P1.GetX(), 3)

- pow(P2.GetX(), 4) - 6 \* P2.GetX() \* P2.GetX() \* P1.GetX() \* P1.GetX() + 2 \* P3P1 \* P3P1 \* P2P3 \* P2P3 - pow(P2P3, 4) +

2 \* P2.GetX() \* P2.GetX() \* P3P1 \* P3P1 + 2 \* P2.GetX() \* P2.GetX() \* P2P3 \* P2P3 + 2 \* P2P3 \* P2P3 \* P1.GetX() \* P1.GetX() - pow(P3P1, 4) - pow(P1.GetX(), 4))) / 0.2e1) / (P1.GetX() - P2.GetX()));

}

else {

P1 = CPoint(x,y);

P2 = CPoint(P1.GetX() + P1P2, P1.GetY());

P3.SetX(-0.5\*((pow(P3P1,2) - pow(P2P3,2) - pow(P1.GetX(),2) + pow(P2.GetX(),2)) / (P1.GetX() - P2.GetX())));

P3.SetY(( (P1.GetY() \* P1.GetX()) - (P1.GetY() \* P2.GetX()) -

sqrt( (2 \* P3P1 \* P3P1 \* P1.GetX() \* P1.GetX() - 4 \* P2.GetX() \* P1.GetX() \* P3P1 \* P3P1 - 4 \* P2.GetX() \* P1.GetX() \* P2P3 \* P2P3 + 4 \* pow(P2.GetX(),3) \* P1.GetX() + 4 \* P2.GetX() \* pow(P1.GetX(), 3)

- pow(P2.GetX(), 4) - 6 \* P2.GetX() \* P2.GetX() \* P1.GetX() \* P1.GetX() + 2 \* P3P1 \* P3P1 \* P2P3 \* P2P3 - pow(P2P3, 4) +

2 \* P2.GetX() \* P2.GetX() \* P3P1 \* P3P1 + 2 \* P2.GetX() \* P2.GetX() \* P2P3 \* P2P3 + 2 \* P2P3 \* P2P3 \* P1.GetX() \* P1.GetX() - pow(P3P1, 4) - pow(P1.GetX(), 4))) / 0.2e1) / (P1.GetX() - P2.GetX()));

}

}

void CTriangle :: SetP1(CPoint P0) {

if (P1.GetY() >= P3.GetY()) {

P1 = P0;

P2 = CPoint(P1.GetX() + P1P2, P1.GetY());

P3.SetX(-0.5\*((pow(P3P1,2) - pow(P2P3,2) - pow(P1.GetX(),2) + pow(P2.GetX(),2)) / (P1.GetX() - P2.GetX())));

P3.SetY(( (P1.GetY() \* P1.GetX()) - (P1.GetY() \* P2.GetX()) +

sqrt( (2 \* P3P1 \* P3P1 \* P1.GetX() \* P1.GetX() - 4 \* P2.GetX() \* P1.GetX() \* P3P1 \* P3P1 - 4 \* P2.GetX() \* P1.GetX() \* P2P3 \* P2P3 + 4 \* pow(P2.GetX(),3) \* P1.GetX() + 4 \* P2.GetX() \* pow(P1.GetX(), 3)

- pow(P2.GetX(), 4) - 6 \* P2.GetX() \* P2.GetX() \* P1.GetX() \* P1.GetX() + 2 \* P3P1 \* P3P1 \* P2P3 \* P2P3 - pow(P2P3, 4) +

2 \* P2.GetX() \* P2.GetX() \* P3P1 \* P3P1 + 2 \* P2.GetX() \* P2.GetX() \* P2P3 \* P2P3 + 2 \* P2P3 \* P2P3 \* P1.GetX() \* P1.GetX() - pow(P3P1, 4) - pow(P1.GetX(), 4))) / 0.2e1) / (P1.GetX() - P2.GetX()));

}

else {

P1 = P0;

P2 = CPoint(P1.GetX() + P1P2, P1.GetY());

P3.SetX(-0.5\*((pow(P3P1,2) - pow(P2P3,2) - pow(P1.GetX(),2) + pow(P2.GetX(),2)) / (P1.GetX() - P2.GetX())));

P3.SetY(( (P1.GetY() \* P1.GetX()) - (P1.GetY() \* P2.GetX()) -

sqrt( (2 \* P3P1 \* P3P1 \* P1.GetX() \* P1.GetX() - 4 \* P2.GetX() \* P1.GetX() \* P3P1 \* P3P1 - 4 \* P2.GetX() \* P1.GetX() \* P2P3 \* P2P3 + 4 \* pow(P2.GetX(),3) \* P1.GetX() + 4 \* P2.GetX() \* pow(P1.GetX(), 3)

- pow(P2.GetX(), 4) - 6 \* P2.GetX() \* P2.GetX() \* P1.GetX() \* P1.GetX() + 2 \* P3P1 \* P3P1 \* P2P3 \* P2P3 - pow(P2P3, 4) +

2 \* P2.GetX() \* P2.GetX() \* P3P1 \* P3P1 + 2 \* P2.GetX() \* P2.GetX() \* P2P3 \* P2P3 + 2 \* P2P3 \* P2P3 \* P1.GetX() \* P1.GetX() - pow(P3P1, 4) - pow(P1.GetX(), 4))) / 0.2e1) / (P1.GetX() - P2.GetX()));

}

}

int CTriangle :: isIn (CPoint P) const{

if (P == P1 || P == P2 || P == P3) return 1;

else {

int ab = P.nCheck(P1, P2);

int bc = P.nCheck(P2, P3);

int cd = P.nCheck(P3, P1);

// Если точка за пределами фигуры...

if( ab > 0 || bc > 0 || cd > 0) return 0;

}

return 1;

}

void CTriangle :: Move (double dx, double dy) {

P1.SetX(P1.GetX() + dx);

P1.SetY(P1.GetY() + dy);

P2.SetX(P2.GetX() + dx);

P2.SetY(P2.GetY() + dy);

P3.SetX(P3.GetX() + dx);

P3.SetY(P3.GetY() + dy);

}

CPoint CTriangle :: GetP1() const{

return P1;

}

CPoint CTriangle :: GetP2() const{

return P2;

}

CPoint CTriangle :: GetP3() const{

return P3;

}

void CTriangle :: Print () {

cout << "Triangle " << P1 << "," << P2 << "," << P3 << endl;

cout << "P1P2 = "<< P1P2 << ", P2P3 = " << P2P3<< ", P3P1 = " <<P3P1<< endl;

}

void CTriangle :: SetDebug (unsigned int d) {

debug = d;

}

unsigned int CTriangle :: GetID () {

return id;

}

unsigned int CTriangle :: GetCurrent () {

return current;

}

unsigned int CTriangle :: GetTotal () {

return total;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Класс «Стенд»**

class CStand {

protected:

CircleList<CPoint> MainPoints;

CTriangle Stand;

private:

static unsigned int debug;

static unsigned int total;

unsigned int id;

static unsigned int current;

virtual void AddPoints ();

public:

CStand(CPoint \_P1,double \_P1P2,CPoint \_P3);

CStand(); // Constructor without parametrs

virtual ~CStand();

virtual int invStand (); // Инвариант

virtual void RefreshPoints ();

virtual void SetP(double,double);

virtual void SetP(CPoint); // Задание точки P1

virtual void MoveStand (double,double); // Плоскопараллельное перемещение на dx dy

const CircleList<CPoint>& CStand :: GetPoints () const;

const CTriangle& CStand :: GetStand() const;

int isIn (const CircleList<CPoint>& Points) const;

virtual void Print ();

virtual void SetDebug (unsigned int);

virtual unsigned int GetID ();

virtual unsigned int GetCurrent();

virtual unsigned int GetTotal();

};

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include "CStand.h"

unsigned int CStand :: debug = 0;

unsigned int CStand :: total = 0;

unsigned int CStand :: current = 0;

void CStand :: AddPoints () {

MainPoints.Append(new CLNode<CPoint>(Stand.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Stand.GetP2()));

MainPoints.Append(new CLNode<CPoint>(Stand.GetP3()));

}

CStand :: CStand (CPoint \_P1, double \_P1P2, CPoint \_P3)

: Stand(\_P1, \_P1P2, \_P3) {

AddPoints();

id = ++total;

++current;

if (debug) cout << "Stand " << Stand.GetP1() << "," << Stand.GetP2() << "," << Stand.GetP3() << " created" << endl;

}

CStand :: CStand () : Stand() {

AddPoints();

id = ++total;

++current;

if (debug) cout << "Stand " << Stand.GetP1() << "," << Stand.GetP2() << "," << Stand.GetP3() << " created" << endl;

}

CStand :: ~CStand () {

if (debug) cout << "Stand " << Stand.GetP1() << "," << Stand.GetP2() << "," << Stand.GetP3() << " deleted" << endl;

--current;

}

int CStand :: invStand () {

return (Stand.inv() && MainPoints.GetElemByNum(0)->GetValue() == Stand.GetP1()

&& MainPoints.GetElemByNum(1)->GetValue() == Stand.GetP2()

&& MainPoints.GetElemByNum(2)->GetValue() == Stand.GetP3());

}

void CStand :: RefreshPoints () {

CLNode<CPoint>\* tN = MainPoints.GetFirst();

tN->SetValue(Stand.GetP1());

tN = tN->GetNext();

tN->SetValue(Stand.GetP2());

tN = tN->GetNext();

tN->SetValue(Stand.GetP3());

}

void CStand:: SetP(double x, double y) {

Stand.SetP1(x,y);

RefreshPoints();

}

void CStand :: SetP(CPoint P0) {

Stand.SetP1(P0);

RefreshPoints();

}

void CStand :: MoveStand (double dx, double dy) {

Stand.Move(dx,dy);

RefreshPoints();

}

int CStand :: isIn (const CircleList<CPoint>& Points) const {

for (int i = 0;i < Points.GetSize();i++) {

if (Stand.isIn(Points.GetElemByNum(i)->GetValue()) == 0) return 0;

}

return 1;

}

const CTriangle& CStand :: GetStand() const {

return Stand;

}

const CircleList<CPoint>& CStand :: GetPoints () const{

return MainPoints;

}

void CStand :: Print () {

cout << "Stand id = " << id <<endl;

Stand.Print();

cout << "Main Points : "<<endl;

cout<<"-------------------"<<endl;

MainPoints.Print();

cout <<"-------------------"<<endl;

}

void CStand :: SetDebug (unsigned int d) {

debug = d;

}

unsigned int CStand :: GetID () {

return id;

}

unsigned int CStand :: GetCurrent () {

return current;

}

unsigned int CStand :: GetTotal () {

return total;

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Класс «Крышка»**

class CCover {

private:

CircleList<CPoint> MainPoints;

CRectangle Hole;

CRectangle Cover;

static unsigned int debug;

static unsigned int total;

unsigned int id;

static unsigned int current;

public:

CCover(CPoint,double cover\_width,double cover\_height, double hole\_width); // P1 width heigth length

CCover(); // Constructor without parametrs

~CCover();

int inv (); // Инвариант

void RefreshPoints ();

void SetP1(double,double);

void SetP1(CPoint P0); // Задание точки P1

void Move (double dx, double dy); // Плоскопараллельное перемещение на dx dy

const CircleList<CPoint>& GetPoints () const;

CRectangle GetHole() const;

CRectangle GetCover() const;

void Print ();

void SetDebug (unsigned int);

unsigned int GetID ();

unsigned int GetCurrent();

unsigned int GetTotal();

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CCover.h"

unsigned int CCover :: debug = 0;

unsigned int CCover :: total = 0;

unsigned int CCover :: current = 0;

CCover :: CCover (CPoint P0, double cover\_width, double cover\_height, double hole\_width) : Cover(P0.GetX(),P0.GetY(),cover\_width,cover\_height),

Hole(P0.GetX() + cover\_width/2 - hole\_width/2,P0.GetY(),hole\_width,cover\_height) {

MainPoints.Append(new CLNode<CPoint>(Cover.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP4()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP2()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP2()));

id = ++total;

++current;

if (debug) cout << "Cover " << Cover.GetP1() << "," << Cover.GetP2() <<

"," << Cover.GetP3() << "," << Cover.GetP4() <<

" Width = "<< Cover.GetWidth() << ", Height = " << Cover.GetHeight()<< endl <<

"Hole " << Hole.GetP1() << "," << Hole.GetP2() <<

"," << Hole.GetP3() << "," << Hole.GetP4() <<

" Width = "<< Cover.GetWidth() << " created"<<endl;

}

CCover :: CCover (): Cover(0,1,1,1),

Hole(0.4,1,0.1,1) {

MainPoints.Append(new CLNode<CPoint>(Cover.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetP4()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP2()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP1()));

MainPoints.Append(new CLNode<CPoint>(Hole.GetP2()));

id = ++total;

++current;

if (debug) cout << "Cover " << Cover.GetP1() << "," << Cover.GetP2() <<

"," << Cover.GetP3() << "," << Cover.GetP4() <<

" Width = "<< Cover.GetWidth() << ", Height = " << Cover.GetHeight()<< endl <<

"Hole " << Hole.GetP1() << "," << Hole.GetP2() <<

"," << Hole.GetP3() << "," << Hole.GetP4() <<

" Width = "<< Cover.GetWidth() << " created"<<endl;

}

CCover :: ~CCover () {

if (debug) cout << "Cover " << Cover.GetP1() << "," << Cover.GetP2() <<

"," << Cover.GetP3() << "," << Cover.GetP4() <<

" Width = "<< Cover.GetWidth() << ", Height = " << Cover.GetHeight()<< endl <<

"Hole " << Hole.GetP1() << "," << Hole.GetP2() <<

"," << Hole.GetP3() << "," << Hole.GetP4() <<

" Width = "<< Cover.GetWidth() << " deleted"<<endl;

--current;

}

int CCover :: inv () {

return (Cover.inv() && Hole.inv() && (Cover.GetHeight() == Hole.GetHeight())

&& (Hole.GetWidth() < Cover.GetWidth()) && (Hole.GetWidth() / 2 <= 1));

}

void CCover :: RefreshPoints () {

CLNode<CPoint>\* tN = MainPoints.GetFirst();

tN->SetValue(Cover.GetP1());

tN = tN->GetNext();

tN->SetValue(Cover.GetP2());

tN = tN->GetNext();

tN->SetValue(Cover.GetP3());

tN = tN->GetNext();

tN->SetValue(Cover.GetP4());

tN = tN->GetNext();

tN->SetValue(Hole.GetP1());

tN = tN->GetNext();

tN->SetValue(Hole.GetP2());

tN = tN->GetNext();

tN->SetValue(Hole.GetP3());

tN = tN->GetNext();

tN->SetValue(Hole.GetP4());

}

void CCover :: SetP1(double x, double y) {

double d = Cover.GetP1().d(Hole.GetP1());

Cover.SetP1(x,y);

Hole.SetP1(Cover.GetP1().GetX() + d, Cover.GetP1().GetY());

RefreshPoints();

}

void CCover :: SetP1(CPoint P0) {

double d = Cover.GetP1().d(Hole.GetP1());

Cover.SetP1(P0);

Hole.SetP1(Cover.GetP1().GetX() + d, Cover.GetP1().GetY());

RefreshPoints();

}

void CCover :: Move (double dx, double dy) {

Cover.SetP1(Cover.GetP1().GetX() + dx,Cover.GetP1().GetY() + dy);

Hole.SetP1(Hole.GetP1().GetX() + dx,Hole.GetP1().GetY() + dy);

RefreshPoints();

}

CRectangle CCover :: GetHole() const {

return Hole;

}

CRectangle CCover :: GetCover() const {

return Cover;

}

const CircleList<CPoint>& CCover :: GetPoints () const{

return MainPoints;

}

void CCover :: Print () {

cout << "Cover "<< Cover.GetP1() << "," << Cover.GetP2() <<

"," << Cover.GetP3() << "," << Cover.GetP4() <<

" Width = "<< Cover.GetWidth() << ", Height = " << Cover.GetHeight()<< endl <<

"Hole " << Hole.GetP1() << "," << Hole.GetP2() <<

"," << Hole.GetP3() << "," << Hole.GetP4() <<

" Width = "<< Cover.GetWidth() <<endl;

}

void CCover :: SetDebug (unsigned int d) {

debug = d;

}

unsigned int CCover :: GetID () {

return id;

}

unsigned int CCover :: GetCurrent () {

return current;

}

unsigned int CCover :: GetTotal () {

return total;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Класс «Закрытый поршневой механизм»**

#include "CPMec.h"

class CClosedPistonMechanism : public CPistonMechanism {

private:

CircleList<CPoint> MainPoints;

CCover Cover;

void AddPoints ();

public:

/\*Конструкторы\*/

//-------------------------------------------------

CClosedPistonMechanism(CPoint,double,double,double,double,double,double,double,double,double,double);

CClosedPistonMechanism(double,double,double,double,double,double,double,double,double,double,double,double);

CClosedPistonMechanism(); // Constructor without parametrs

//-------------------------------------------------

/\*Деструктор\*/

//-------------------------------------------------

virtual ~CClosedPistonMechanism();

//-------------------------------------------------

/\*\*/

int inv (); // Инвариант

void RefreshPoints ();

//---------------------------------------------------

void SetP1(double,double);

void SetP1(CPoint); // Задание точки P1

void Move (double,double); // Плоскопараллельное перемещение на dx dy

const CircleList<CPoint>& GetPoints () const;

const CCover& GetCover() const;

//---------------------------------------------------

void Print ();

friend ostream& operator<<( ostream& output,const CClosedPistonMechanism&);

};

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include "CPCMec.h"

void CClosedPistonMechanism :: AddPoints () {

MainPoints.Append(new CLNode<CPoint>(Cup.GetLeftSide().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetLeftSide().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetLeftSide().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetLeftSide().GetP4()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetBottomSide().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetBottomSide().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetBottomSide().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetBottomSide().GetP4()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetRightSide().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetRightSide().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetRightSide().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cup.GetRightSide().GetP4()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetPiston().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetPiston().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetPiston().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetPiston().GetP4()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetStock().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Piston.GetStock().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetCover().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetCover().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetCover().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetCover().GetP4()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetHole().GetP1()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetHole().GetP2()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetHole().GetP3()));

MainPoints.Append(new CLNode<CPoint>(Cover.GetHole().GetP4()));

}

CClosedPistonMechanism :: CClosedPistonMechanism (double x1, double y1, double leftwidth, double leftheight, double bottomwidth, double bottomheight, double rightwidth, double y2, double pistonheight, double stocklength,double cover\_height,double hole\_width)

: CPistonMechanism(x1,y1,leftwidth,leftheight,bottomwidth,bottomheight,rightwidth,y2,pistonheight,stocklength),

Cover(CPoint(x1,y1 + cover\_height),leftwidth + bottomwidth + rightwidth,cover\_height,hole\_width)

{

AddPoints();

id = ++total;

++current;

if (debug){ cout << "LeftCupSide : "; Cup.GetLeftSide().Print();

cout << "BottomCupSide : "; Cup.GetBottomSide().Print();

cout << "RightCupSide : "; Cup.GetRightSide().Print();

cout << "Piston : "; Piston.GetPiston().Print();

cout << "Stock : "; Piston.GetStock().Print(); cout <<endl<<

cout << "Cover : "; Cover.GetCover().Print(); cout << endl;

cout << "Hole : "; Cover.GetHole().Print(); cout <<" created";

}

}

CClosedPistonMechanism :: CClosedPistonMechanism (CPoint P1, double leftwidth, double leftheight, double bottomwidth, double bottomheight, double rightwidth, double y2, double pistonheight, double stocklength, double cover\_height, double hole\_width)

: CPistonMechanism(P1,leftwidth,leftheight,bottomwidth,bottomheight,rightwidth,y2,pistonheight,stocklength),

Cover(CPoint(P1.GetX(),P1.GetY() + cover\_height),leftwidth + bottomwidth + rightwidth,cover\_height,hole\_width)

{

AddPoints();

id = ++total;

++current;

if (debug){ cout << "LeftCupSide : "; Cup.GetLeftSide().Print();

cout << "BottomCupSide : "; Cup.GetBottomSide().Print();

cout << "RightCupSide : "; Cup.GetRightSide().Print();

cout << "Piston : "; Piston.GetPiston().Print();

cout << "Stock : "; Piston.GetStock().Print(); cout <<endl<<

cout << "Cover : "; Cover.GetCover().Print(); cout << endl;

cout << "Hole : "; Cover.GetHole().Print(); cout <<" created";

}

}

CClosedPistonMechanism :: CClosedPistonMechanism ()

: Cover()

{

AddPoints();

id = ++total;

++current;

if (debug){ cout << "LeftCupSide : "; Cup.GetLeftSide().Print();

cout << "BottomCupSide : "; Cup.GetBottomSide().Print();

cout << "RightCupSide : "; Cup.GetRightSide().Print();

cout << "Piston : "; Piston.GetPiston().Print();

cout << "Stock : "; Piston.GetStock().Print(); cout <<endl<<

cout << "Cover : "; Cover.GetCover().Print(); cout << endl;

cout << "Hole : "; Cover.GetHole().Print(); cout <<" created";

}

}

CClosedPistonMechanism :: ~CClosedPistonMechanism () {

if (debug){ cout << "LeftCupSide : "; Cup.GetLeftSide().Print();

cout << "BottomCupSide : "; Cup.GetBottomSide().Print();

cout << "RightCupSide : "; Cup.GetRightSide().Print();

cout << "Piston : "; Piston.GetPiston().Print();

cout << "Stock : "; Piston.GetStock().Print(); cout <<endl<<

cout << "Cover : "; Cover.GetCover().Print(); cout << endl;

cout << "Hole : "; Cover.GetHole().Print(); cout <<" deleted";

}

--current;

}

int CClosedPistonMechanism :: inv () {

return (Cup.inv() && Piston.inv() && (Piston.GetPiston().GetWidth() == Cup.GetBottomSide().GetP1().d(Cup.GetBottomSide().GetP2())) &&

(Cup.GetBottomSide().GetP1().d(Piston.GetPiston().GetP3()) >= 0) && (Piston.GetPiston().GetP3().GetY() >= Cup.GetBottomSide().GetP1().GetY()) &&

(Piston.GetPiston().GetP1().GetY() <= Cup.GetLeftSide().GetP2().GetY()) &&

(Piston.GetStock().GetL() >= Cup.GetLeftSide().GetHeight() - Cup.GetBottomSide().GetHeight() - Piston.GetPiston().GetHeight()) &&

(Cup.GetLeftSide().GetP1() == MainPoints.GetElemByNum(0)->GetValue() &&

Cup.GetLeftSide().GetP2() == MainPoints.GetElemByNum(1)->GetValue() &&

Cup.GetLeftSide().GetP3() == MainPoints.GetElemByNum(2)->GetValue() &&

Cup.GetLeftSide().GetP4() == MainPoints.GetElemByNum(3)->GetValue() &&

Cup.GetBottomSide().GetP1() == MainPoints.GetElemByNum(4)->GetValue() &&

Cup.GetBottomSide().GetP2() == MainPoints.GetElemByNum(5)->GetValue() &&

Cup.GetBottomSide().GetP3() == MainPoints.GetElemByNum(6)->GetValue() &&

Cup.GetBottomSide().GetP4() == MainPoints.GetElemByNum(7)->GetValue() &&

Cup.GetRightSide().GetP1() == MainPoints.GetElemByNum(8)->GetValue() &&

Cup.GetRightSide().GetP2() == MainPoints.GetElemByNum(9)->GetValue() &&

Cup.GetRightSide().GetP3() == MainPoints.GetElemByNum(10)->GetValue() &&

Cup.GetRightSide().GetP4() == MainPoints.GetElemByNum(11)->GetValue() &&

Piston.GetPiston().GetP1() == MainPoints.GetElemByNum(12)->GetValue() &&

Piston.GetPiston().GetP2() == MainPoints.GetElemByNum(13)->GetValue() &&

Piston.GetPiston().GetP3() == MainPoints.GetElemByNum(14)->GetValue() &&

Piston.GetPiston().GetP4() == MainPoints.GetElemByNum(15)->GetValue() &&

Piston.GetStock().GetP1() == MainPoints.GetElemByNum(16)->GetValue() &&

Piston.GetStock().GetP2() == MainPoints.GetElemByNum(17)->GetValue() &&

Cover.GetCover().GetP1() == MainPoints.GetElemByNum(18)->GetValue() &&

Cover.GetCover().GetP2() == MainPoints.GetElemByNum(19)->GetValue() &&

Cover.GetCover().GetP3() == MainPoints.GetElemByNum(20)->GetValue() &&

Cover.GetCover().GetP4() == MainPoints.GetElemByNum(21)->GetValue() &&

Cover.GetHole().GetP1() == MainPoints.GetElemByNum(22)->GetValue() &&

Cover.GetHole().GetP2() == MainPoints.GetElemByNum(23)->GetValue() &&

Cover.GetHole().GetP3() == MainPoints.GetElemByNum(24)->GetValue() &&

Cover.GetHole().GetP4() == MainPoints.GetElemByNum(25)->GetValue()) && Cover.inv());

}

void CClosedPistonMechanism :: RefreshPoints () {

CLNode<CPoint>\* tN = MainPoints.GetFirst();

tN->SetValue(Cup.GetLeftSide().GetP1());

tN = tN->GetNext();

tN->SetValue(Cup.GetLeftSide().GetP2());

tN = tN->GetNext();

tN->SetValue(Cup.GetLeftSide().GetP3());

tN = tN->GetNext();

tN->SetValue(Cup.GetLeftSide().GetP4());

tN = tN->GetNext();

tN->SetValue(Cup.GetBottomSide().GetP1());

tN = tN->GetNext();

tN->SetValue(Cup.GetBottomSide().GetP2());

tN = tN->GetNext();

tN->SetValue(Cup.GetBottomSide().GetP3());

tN = tN->GetNext();

tN->SetValue(Cup.GetBottomSide().GetP4());

tN = tN->GetNext();

tN->SetValue(Cup.GetRightSide().GetP1());

tN = tN->GetNext();

tN->SetValue(Cup.GetRightSide().GetP2());

tN = tN->GetNext();

tN->SetValue(Cup.GetRightSide().GetP3());

tN = tN->GetNext();

tN->SetValue(Cup.GetRightSide().GetP4());

tN = tN->GetNext();

tN->SetValue(Piston.GetPiston().GetP1());

tN = tN->GetNext();

tN->SetValue(Piston.GetPiston().GetP2());

tN = tN->GetNext();

tN->SetValue(Piston.GetPiston().GetP3());

tN = tN->GetNext();

tN->SetValue(Piston.GetPiston().GetP4());

tN = tN->GetNext();

tN->SetValue(Piston.GetStock().GetP1());

tN = tN->GetNext();

tN->SetValue(Piston.GetStock().GetP2());

tN = tN->GetNext();

tN->SetValue(Cover.GetCover().GetP1());

tN = tN->GetNext();

tN->SetValue(Cover.GetCover().GetP2());

tN = tN->GetNext();

tN->SetValue(Cover.GetCover().GetP3());

tN = tN->GetNext();

tN->SetValue(Cover.GetCover().GetP4());

tN = tN->GetNext();

tN->SetValue(Cover.GetHole().GetP1());

tN = tN->GetNext();

tN->SetValue(Cover.GetHole().GetP2());

tN = tN->GetNext();

tN->SetValue(Cover.GetHole().GetP3());

tN = tN->GetNext();

tN->SetValue(Cover.GetHole().GetP4());

}

void CClosedPistonMechanism:: SetP1(double x, double y) {

double tL = 0;

tL = Cup.GetBottomSide().GetP1().d(Piston.GetPiston().GetP3());

Cup.SetP1(x,y);

Piston.SetP1(Cup.GetBottomSide().GetP1().GetX(),Cup.GetBottomSide().GetP1().GetY() + tL + Piston.GetPiston().GetHeight());

Cover.SetP1(CPoint(x,y+Cover.GetCover().GetHeight()));

RefreshPoints();

}

void CClosedPistonMechanism :: SetP1(CPoint P0) {

double tL = 0;

tL = CPoint(Cup.GetBottomSide().GetP1()).d(Piston.GetPiston().GetP3());

Cup.SetP1(P0);

Piston.SetP1(Cup.GetBottomSide().GetP1().GetX(),Cup.GetBottomSide().GetP1().GetY() + tL + Piston.GetPiston().GetHeight());

Cover.SetP1(CPoint(P0.GetX(),P0.GetY()+Cover.GetCover().GetHeight()));

RefreshPoints();

}

void CClosedPistonMechanism :: Move (double dx, double dy) {

Cup.Move(dx,dy);

Piston.Move(dx,dy);

Cover.Move(dx,dy);

RefreshPoints();

}

const CCover& CClosedPistonMechanism :: GetCover() const {

return Cover;

}

const CircleList<CPoint>& CClosedPistonMechanism :: GetPoints () const{

return MainPoints;

}

void CClosedPistonMechanism :: Print () {

cout << "ClosedPistonMechanism id = "<< id <<endl <<

"LeftCupSide : "; Cup.GetLeftSide().Print();

cout << "BottomCupSide : "; Cup.GetBottomSide().Print();

cout << "RightCupSide : "; Cup.GetRightSide().Print();

getch();

cout << "Piston : "; Piston.GetPiston().Print();

cout << "Stock : "; Piston.GetStock().Print();

cout << "Cover : "; Cover.GetCover().Print(); cout << endl;

cout << "Hole : "; Cover.GetHole().Print(); cout << endl;

getch();

cout << "Main Points : "<<endl;

cout<<"-------------------"<<endl;

MainPoints.Print();

cout <<"-------------------"<<endl;

getch();

}

ostream& operator<<(ostream& output,const CClosedPistonMechanism& P) {

output << "ClosedPistonMechanism id = " << P.GetID() << endl;

output << "LeftCupSide : " << P.GetCup().GetLeftSide();

output << "BottomCupSide : " << P.GetCup().GetBottomSide();

output << "RightCupSide : " << P.GetCup().GetRightSide();

output << "Piston : " << P.GetPiston().GetPiston();

output<< "Stock : P1"<<P.GetPiston().GetStock().GetP1() << ", P2" << P.GetPiston().GetStock().GetP2()<<", L = " << P.GetPiston().GetStock().GetL()<<endl;

cout << "Cover : " << P.GetCover().GetCover() << endl;

cout << "Hole : " << P.GetCover().GetHole() << endl;

output << "Main Points : "<<endl;

output<<"-------------------"<<endl;

if(P.GetPoints().GetSize() == 0){

cout<<endl<<"The List is empty"<<endl;

return output;

}

short i = 0;

CLNode <CPoint>\* temp = P.GetPoints().GetFirst();

while (i < P.GetPoints().GetSize() && temp) {

output << "Element "<<i<<": "<< temp->GetValue()<<endl;

i++;

temp = temp->GetNext();

}

output <<endl<<endl;

output <<"-------------------"<<endl;

return output;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Класс «Вид на столешницу»**

#include "CPColl.h"

#include "CStand.h"

class CView : public CPistonCollection, public CStand {

private:

public:

/\*Конструкторы\*/

//-------------------------------------------------

CView(CPoint \_P1,double \_P1P2,CPoint \_P3); // P1 width heigth length

CView(); // Constructor without parametrs

//-------------------------------------------------

/\*Деструктор\*/

//-------------------------------------------------

~CView();

//-------------------------------------------------

int inv (); // Инвариант

//---------------------------------------------------

void SetCollectionP1(CPoint P0); // Задание точки P1

void SetStandP1 (CPoint P0);

void MoveCollection (double dx, double dy); // Плоскопараллельное перемещение на dx dy

void MoveStand (double dx, double dy); // Плоскопараллельное перемещение на dx dy

void MoveCollectionAndStand (double dx, double dy);

void SetCollectionAndStandP1(CPoint P0); // Задание точки P1

void Add(CPistonMechanism\*) const;

void Add(const CPistonCollection& C) const;

int isOnStand (CPistonMechanism\*) const;

//---------------------------------------------------

void Print ();

void PrintView ();

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include "CView.h"

CView :: CView (CPoint \_P1,double \_P1P2,CPoint \_P3)

: CStand(\_P1,\_P1P2,\_P3),

CPistonCollection() {}

CView :: CView ()

: CStand(),

CPistonCollection() {}

CView :: ~CView () {}

int CView :: inv () {

return (invColl() && invStand());

}

void CView :: SetCollectionP1(CPoint P0) {

CPistonCollection :: SetP1(P0);

}

void CView :: SetStandP1(CPoint P0) {

CStand :: SetP(P0);

}

void CView :: MoveCollection (double dx, double dy) {

CPistonCollection :: Move(dx,dy);

}

void CView :: MoveStand (double dx, double dy) {

CStand :: MoveStand(dx,dy);

}

void CView :: MoveCollectionAndStand (double dx, double dy) {

MoveStand(dx,dy);

MoveCollection(dx,dy);

}

void CView :: SetCollectionAndStandP1(CPoint P0){

SetCollectionP1(P0);

SetStandP1(P0);

}

void CView :: Add(CPistonMechanism\* P) const{

CPistonCollection :: Add(P);

}

void CView :: Add(const CPistonCollection& C) const{

for (int i =0;i < C.GetCollection().GetSize();i++)

CPistonCollection :: Add(C.GetCollection().GetElemByNum(i)->GetValue());

}

int CView :: isOnStand (CPistonMechanism\* P) const {

return isIn(P->GetPoints());

}

void CView :: PrintView () {

CStand :: Print ();

Collection.GetFirst();

for (int i = 0; i < Collection.GetSize();i++,Collection.NextCur())

if (isOnStand(Collection.GetCur()->GetValue())){

cout << "Element #"<<i<<": ";

Collection.GetCur()->GetValue()->Print();

}

}

void CView :: Print () {

CStand :: Print ();

CPistonCollection :: Print();

}

Тестирующая программа

#include <iostream.h>

#include <stdlib.h>

#include <conio.h>

#include <stdio.h>

#include <dos.h>

#include "CSection.h"

#include "CPoint.h"

#include "CRect.h"

#include "CList.h"

#include "CListN.h"

#include "CPiston.h"

#include "CCup.h"

#include "CPMec.h"

#include "CPColl.h"

#include "CCover.h"

#include "CTria.h"

#include "CStand.h"

#include "CPCMec.h"

#include "CView.h"

void main () {

{

cout << endl<<endl<<"-----------------------------------------------"<<endl;

cout << "Programm is creating the view on collection of schematic images of"<<endl

<< "closed piston mechanisms which can be moved or placed in point"<<endl

<< "Programmed by : Artyom Mon'ko"<<endl<<endl

<< "----------------Press any key to continue or ESC to exit----------------"<<endl;

while(1) {

int key = getch();

if (key == 27) return; // Waiting for some action

else break;

}

int flag = 1;

int ViewCreated = 0;

char\* buffer = new char[80];

int menu = 0;

CPistonCollection c;

CPistonMechanism\* p;

CView \*v;

while (flag) {

cout << "1.Create View"<<endl

<< "2.Add Mechanism to Collection"<<endl

<< "3.Delete Mechanism from Collection"<<endl

<< "4.Move Collection and Stand"<<endl

<< "5.Place Collection and Stand into Point"<<endl

<< "6.Place Collection into Point"<<endl

<< "7.Place Stand into Point"<<endl

<< "8.Move Collection"<<endl

<< "9.Move Stand"<<endl

<< "10.Move Piston of element"<<endl

<< "11.Move every Piston"<<endl

<< "12.Print View"<<endl

<< "13.Print Collection"<<endl

<< "14.Exit"<<endl;

cin >> buffer;

if ((atoi(buffer) == 0) || (atoi(buffer) < 1 || atoi(buffer) > 14)) {

cout << "Invalid input, try again : "<<endl;

continue;

}

else {

menu = (int)atoi(buffer);

switch (menu) {

case 1: {

double x1 = 0,y1 = 0, \_P1P2,x3,y3;

cout << "Enter x value of point P1 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter lenght of P1P2 side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

\_P1P2 = atof(buffer);

cout << "Enter x value of point P3 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x3 = atof(buffer);

cout << "Enter y value of point P3 : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y3 = atof(buffer);

v = new CView(CPoint(x1,y1),\_P1P2,CPoint(x3,y3));

if (v->inv()) {ViewCreated = 1;cout << "View is created"<<endl;}

else cout << "View do not created due to some input errors"<<endl;

break;

}

case 2: {

double x1 = 0,y1 = 0, lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width;

cout << "Enter x value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x1 = atof(buffer);

cout << "Enter y value of point P1(A) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y1 = atof(buffer);

cout << "Enter width of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lwidth = atof(buffer);

cout << "Enter height of left side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

lheight = atof(buffer);

cout << "Enter width of bottom side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bwidth = atof(buffer);

cout << "Enter height of bottom : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

bheight = atof(buffer);

cout << "Enter width of right side : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

rwidth = atof(buffer);

cout << "Enter y value of piston point P1(I) : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y2 = atof(buffer);

cout << "Enter piston height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

pheight = atof(buffer);

cout << "Enter stock length : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

slen = atof(buffer);

cout << "Enter cover height : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

cover\_height = atof(buffer);

cout << "Enter hole's width : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

hole\_width = atof(buffer);

p = new CClosedPistonMechanism(x1,y1,lwidth,lheight,bwidth,bheight,rwidth,y2,pheight,slen,cover\_height,hole\_width);

if (p->inv()){

c.Add(p);

if (ViewCreated) v->Add(p);

cout << "Closed Piston Mechanism Added"<<endl;}

else cout << "Closed Piston Mechanism not added due to some input errors"<<endl;

break;

}

case 3: {

cout << "Enter number of element : ";

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

c.Delete(atoi(buffer));

if (ViewCreated) v->Delete(atoi(buffer));

if (c.invColl()) cout << "Element deleted"<<endl;

break;

}

case 4: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveCollectionAndStand(dx,dy);

if (v->inv()) cout << "Collection and stand are moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else {

c.Move(dx,dy);

if (c.invColl()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

break;

}

case 5: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetCollectionAndStandP1(CPoint(x,y));

if (v->inv()) cout << "Collection and stand are placed to Point("<<x<<";"<<y<<")"<<endl;

}

else {

c.SetP1(CPoint(x,y));

if (c.invColl()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

break;

}

case 6: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetCollectionP1(CPoint(x,y));

if (v->inv()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

else {

c.SetP1(CPoint(x,y));

if (c.invColl()) cout << "Collection is placed to Point("<<x<<";"<<y<<")"<<endl;

}

break;

}

case 7: {

double x,y;

cout << "Enter P1 "<<endl;

cout << "Enter x : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

x = atof(buffer);

cout << "Enter y : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

y = atof(buffer);

if (ViewCreated) {

v->SetStandP1(CPoint(x,y));

if (v->inv()) cout << "Stand is placed to Point("<<x<<";"<<y<<")"<<endl;

}

else cout << "View is not created"<<endl;

break;

}

case 8: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveCollection(dx,dy);

if (v->inv()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else {

c.Move(dx,dy);

if (c.invColl()) cout << "Collection is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

break;

}

case 9: {

double dx,dy;

cout << "Enter dx : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dx = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MoveStand(dx,dy);

if (v->inv()) cout << "Stand is moved by dx = "<<dx<<", dy = "<<dy<<endl;

}

else

cout << "View is not created"<<dy<<endl;

break;

}

case 10: {

double dy, index = 0;

cout << "Index of element : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

index = atof(buffer);

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->GetCollection().GetElemByNum(index)->GetValue()->MovePiston(dy);

if (v->inv()) cout << "Piston of "<< index<< "'th element is moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

else {

c.GetCollection().GetElemByNum(index)->GetValue()->MovePiston(dy);

if (c.invColl()) cout << "Piston of "<< index<< "'th element is moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

break;

}

case 11: {

double dy;

cout << "Enter dy : "<<endl;

while (1) {

cin >> buffer;

if (atoi(buffer) == 0 && buffer[0] != '0') {

cout << "Invalind input, try again : ";

continue;

}

else break;

}

dy = atof(buffer);

if (ViewCreated) {

v->MovePiston(dy);

if (v->inv()) cout << "Pistons moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

else {

c.MovePiston(dy);

if (c.invColl()) cout << "Pistons moved by dy = "<<dy<<endl;

else cout << "Piston didn't move"<<endl;

}

break;

}

case 12: {

if(ViewCreated) v->PrintView();

else cout << "View is not created"<<endl;

break;

}

case 13: {

//CPistonMechanism\* pm = new CClosedPistonMechanism(3,10,1,7,4,2,1,8,2,6,2,1);

if(ViewCreated) v->Print();

else c.Print();

break;

}

case 14: {

flag = 0;

break;

}

}

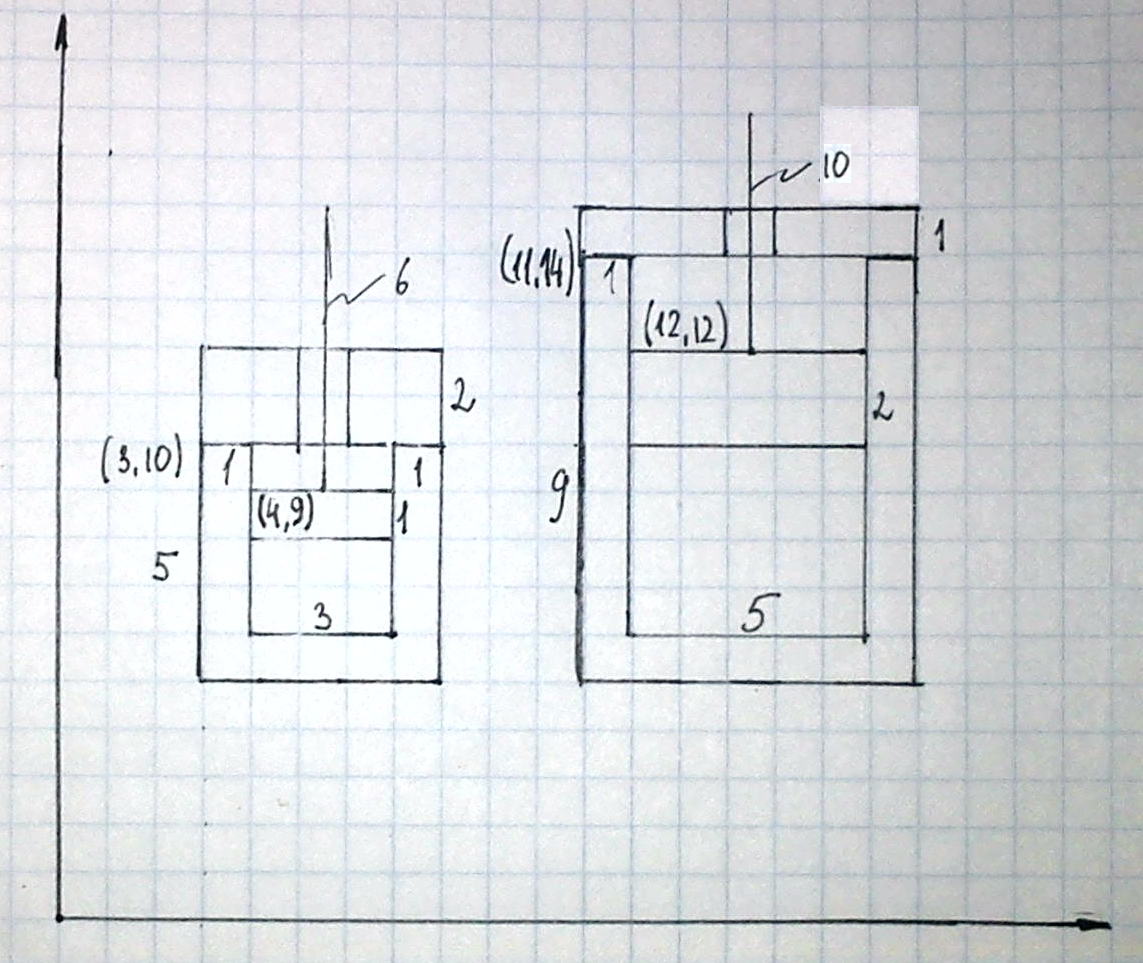
}

}

}

}

Результаты работы программы



1.Create View

2.Add Mechanism to Collection

3.Delete Mechanism from Collection

4.Move Collection and Stand

5.Place Collection and Stand into Point

6.Place Collection into Point

7.Place Stand into Point

8.Move Collection

9.Move Stand

10.Move Piston of element

11.Move every Piston

12.Print View

13.Print Collection

14.Exit

5

Enter P1

Enter x :

3

Enter y :

10

Collection and stand are placed to Point(3;10)

1.Create View

2.Add Mechanism to Collection

3.Delete Mechanism from Collection

4.Move Collection and Stand

5.Place Collection and Stand into Point

6.Place Collection into Point

7.Place Stand into Point

8.Move Collection

9.Move Stand

10.Move Piston of element

11.Move every Piston

12.Print View

13.Print Collection

14.Exit

12

Stand id = 1

Triangle (3;10),(123;10),(1;-190)

P1P2 = 120, P2P3 = 234.273, P3P1 = 200.01

Main Points :

-------------------

Element 0: (3;10)

Element 1: (123;10)

Element 2: (1;-190)

-------------------

Press any key to see more

1.Create View

2.Add Mechanism to Collection

3.Delete Mechanism from Collection

4.Move Collection and Stand

5.Place Collection and Stand into Point

6.Place Collection into Point

7.Place Stand into Point

8.Move Collection

9.Move Stand

10.Move Piston of element

11.Move every Piston

12.Print View

13.Print Collection

14.Exit

13

Stand id = 1

Triangle (3;10),(123;10),(1;-190)

P1P2 = 120, P2P3 = 234.273, P3P1 = 200.01

Main Points :

-------------------

Element 0: (3;10)

Element 1: (123;10)

Element 2: (1;-190)

-------------------

Press any key to see more

Element 0: ClosedPistonMechanism id = 2

LeftCupSide : Rectangle : id = 4 (3;10),(4;10),(3;5),(4;5) Width = 1, Height = 5

BottomCupSide : Rectangle : id = 3 (4;6),(7;6),(4;5),(7;5) Width = 3, Height = 1

RightCupSide : Rectangle : id = 2 (7;10),(8;10),(7;5),(8;5) Width = 1, Height =

5

Piston : Rectangle : id = 1 (4;9),(7;9),(4;8),(7;8) Width = 3, Height = 1

Stock : P1(5.5;9) -- P2(5.5;15), L = 6

Cover : Rectangle : id = 6 (3;12),(8;12),(3;10),(8;10) Width = 5, Height = 2

Hole : Rectangle : id = 5 (5;12),(6;12),(5;10),(6;10) Width = 1, Height = 2

Main Points :

-------------------

Element 0: (3;10)

Element 1: (4;10)

Element 2: (3;5)

Element 3: (4;5)

Element 4: (4;6)

Element 5: (7;6)

Element 6: (4;5)

Element 7: (7;5)

Element 8: (7;10)

Element 9: (8;10)

Element 10: (7;5)

Element 11: (8;5)

Element 12: (4;9)

Element 13: (7;9)

Element 14: (4;8)

Element 15: (7;8)

Element 16: (5.5;9)

Element 17: (5.5;15)

Element 18: (3;12)

Element 19: (8;12)

Element 20: (3;10)

Element 21: (8;10)

Element 22: (5;12)

Element 23: (6;12)

Element 24: (5;10)

Element 25: (6;10)

-------------------

Element 1: ClosedPistonMechanism id = 6

LeftCupSide : Rectangle : id = 16 (3;10),(4;10),(3;1),(4;1) Width = 1, Height =

9

BottomCupSide : Rectangle : id = 15 (4;2),(9;2),(4;1),(9;1) Width = 5, Height =

1

RightCupSide : Rectangle : id = 14 (9;10),(10;10),(9;1),(10;1) Width = 1, Height

= 9

Piston : Rectangle : id = 13 (4;8),(9;8),(4;6),(9;6) Width = 5, Height = 2

Stock : P1(6.5;8) -- P2(6.5;18), L = 10

Cover : Rectangle : id = 18 (3;11),(10;11),(3;10),(10;10) Width = 7, Height = 1

Hole : Rectangle : id = 17 (6;11),(7;11),(6;10),(7;10) Width = 1, Height = 1

Main Points :

-------------------

Element 0: (3;10)

Element 1: (4;10)

Element 2: (3;1)

Element 3: (4;1)

Element 4: (4;2)

Element 5: (9;2)

Element 6: (4;1)

Element 7: (9;1)

Element 8: (9;10)

Element 9: (10;10)

Element 10: (9;1)

Element 11: (10;1)

Element 12: (4;8)

Element 13: (9;8)

Element 14: (4;6)

Element 15: (9;6)

Element 16: (6.5;8)

Element 17: (6.5;18)

Element 18: (3;11)

Element 19: (10;11)

Element 20: (3;10)

Element 21: (10;10)

Element 22: (6;11)

Element 23: (7;11)

Element 24: (6;10)

Element 25: (7;10)

-------------------

1.Create View

2.Add Mechanism to Collection

3.Delete Mechanism from Collection

4.Move Collection and Stand

5.Place Collection and Stand into Point

6.Place Collection into Point

7.Place Stand into Point

8.Move Collection

9.Move Stand

10.Move Piston of element

11.Move every Piston

12.Print View

13.Print Collection

14.Exit