**Московский Энергетический Институт (Технический Университет)**

Москва, ‘07

**Отчет по дисциплине «ТИ и ИО»**

**Лабораторная работа №1**

**Подготовил студент группы А-13-03 Трофимов В.С.**

Оглавление

[Постановка задачи - 3 -](#_Toc166844037)

[Общее описание предметной области - 3 -](#_Toc166844038)

[Поведение машин - 4 -](#_Toc166844039)

[Описание табличной модели - 5 -](#_Toc166844040)

[Таблица 1 - 5 -](#_Toc166844041)

[Таблица 2 - 5 -](#_Toc166844042)

[Таблица 3 - 6 -](#_Toc166844043)

[Таблица 4 - 7 -](#_Toc166844044)

[Таблица 5 - 8 -](#_Toc166844045)

[Приложение. Исходный код - 9 -](#_Toc166844046)

[Обработка сообщений - 9 -](#_Toc166844047)

[Генерирование машин - 12 -](#_Toc166844048)

[Движение - 12 -](#_Toc166844049)

[Повороты - 13 -](#_Toc166844050)

[Переключение светофоров - 14 -](#_Toc166844051)

[Определение занята ли дорога, есть ли машина впереди - 14 -](#_Toc166844052)

[Прочие процедуры работы со списком машин - 15 -](#_Toc166844053)

[Примеры работы программы - 16 -](#_Toc166844054)

# Постановка задачи

Сформировать модель принятия решений в виде совокупности таблиц решений и написать модуль-интерпретатор для данной модели. Объект моделирования – автомойка с выездом.

# Общее описание предметной области

Имеется автомойка с двумя въездами/выездами. Перед каждым въездом имеется светофор, который запрещает въезд при занятости помещения. При выезде машины с второстепенной дороги, она должна пропускать другие машины, движущиеся по главной дороги.

У каждой машины имеется:

* статус-позиция в данный момент движения (движется по главной дороги – onRoad, движется перед мойкой – beforeW, внутри мойки – insideW, после мойки - afterW)
* кол-во шагов (но это не реальное кол-во, а просто некоторые «метки», уточняющие позицию машины в данной области)
* необходимость ехать на мойку (isDirty)
* помыта ли машина (isWash). Этот параметр принимает значение true ТОЛЬКО в случае выезда из мойки. Т.е. если машина вообще не заезжает на мойку, то она считается не помытой.
* Изображение машины (черная машина – грязная, красная – чистая, желтая – помытая). Используется компонент TRotateImage для удобства вращения изображения машин на поворотах.
* Направление движения (вверх, вниз, влево, вправо), судя по экрану (вид сверху)
* Дорожная полоса (А, В, main). Поскольку у нас въезд на мойку с двух полос, возникает необходимость хранить номер полосы для каждой машины, чтобы корректно отображать ее движение.
* Все машины хранятся в списке

|  |  |
| --- | --- |
| obz.JPG  Рис. 1 | Рис. 2 |

На рисунке 1 отмечены зоны нахождения машины, а так же шаги. Эти шаги, как уже говорилось, не реальные – другими словами, это просто отметки, с помощью которых можно более точно определить нахождение машины в зоне (нахождение перед перекрестком/светофором).

## Поведение машин

Рассмотрим поведение машин для разных случаев:

1. Машине не нужно на мойку (красная машина)

В этом случае машина просто будет двигать по прямой, никуда не сворачивая

1. Машине нужно на мойку (еще не помыта – черная машина)
   1. Место свободно

Место считается свободным, если кол-во машин в зоне мойки < 6 (2 машины на мойке и 4 в очередях)

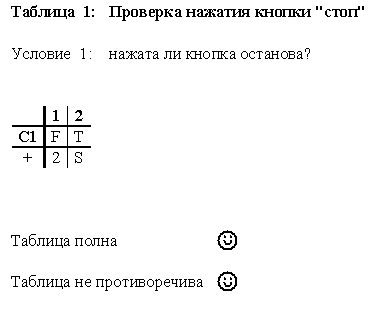
* + 1. В этом случае машина поворачивает на полосу А или В – в зависимости от того, на какой полосе меньше машин
  1. При красном светофоре машина останавливается (отметка 20) и ждет зеленого сигнала
  2. При зеленом светофоре машина заезжает на мойку (светофор переключается на красный)
  3. Место занято
     1. Машина проезжает мимо по прямой

1. Машине нужно было на мойку (уже помыта – желтая машина)
   1. Выезжает из автомойки (светофор переключается на зеленый свет)
   2. При подъезде ко второму перекрестку (отметка 50) машина «смотрит» занята ли главная дорога.
      1. Если есть машина на главной дороге (отметка 30), второстепенная машина останавливается и пропускает первую
      2. В случае отсутствия машины на главной дороге (тут понимается не вся главная дорога, а только часть около второго перекрестка), второстепенная машина (на отметке 50) поворачивает на главную дорогу.

# Описание табличной модели

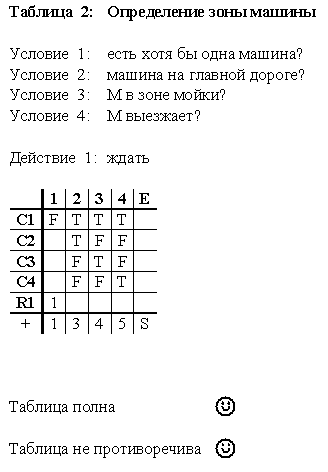
## Таблица 1

Предназначена для отслеживания нажатия кнопки останова.



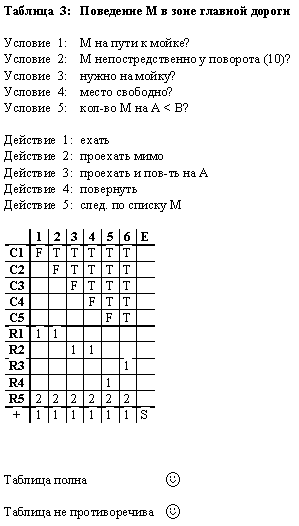
## Таблица 2

Проверяет, в какой зоне находится машина и вызывает нужную таблицу.



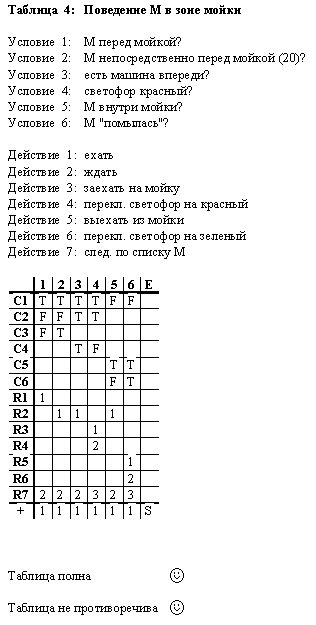
## Таблица 3

Моделирует поведение машины на главной дороге (onRoad)



## Таблица 4

Моделирует поведение машины перед самой мойкой и в ней самой (beforeW, insideW)



## Таблица 5

Моделируется поведение машин после мойки и выезд на главную дорогу (afterW)



# Приложение. Исходный код

## Обработка сообщений

procedure TfrmCarWashing.WndProc(var Message: TMessage);

var

res: boolean;

i: byte;

begin

// если получили сообщение

if message.Msg = msgSIMPR then begin

res := false;

if message.WParamHi = 0 then begin

// УСЛОВИЕ

case message.WParamLo of

// таблица 1

1: begin

case message.LParam of

1: begin // условие 1

res := isProgramStop;

if res then memo1.Lines.Add('условие 11 YES')

else memo1.Lines.Add('условие 11 NO');

end;

end; {case}

end;

// таблица 2

2: begin

case message.LParam of

1: begin // условие 1

if mLengthCar > 0 then res := true;

if res then memo1.Lines.Add('условие 21 YES')

else memo1.Lines.Add('условие 21 NO');

end;

2: begin // условие 2

if (mCarCurr <> nil) and

(mCarCurr.status = onRoad) then res := true;

if res then memo1.Lines.Add('условие 22 YES')

else memo1.Lines.Add('условие 22 NO');

end;

3: begin // условие 3

if (mCarCurr <> nil) and

((mCarCurr.status = beforeW) or

(mCarCurr.status = insideW)) then res := true;

if res then memo1.Lines.Add('условие 23 YES')

else memo1.Lines.Add('условие 23 NO');

end;

4: begin // условие 4

if (mCarCurr <> nil) and

(mCarCurr.status = afterW) then res := true;

if res then memo1.Lines.Add('условие 24 YES')

else memo1.Lines.Add('условие 24 NO');

end;

end; {case}

end;

// таблица 3

3: begin

case message.LParam of

1: begin // условие 1

if (mCarCurr <> nil) and

(mCarCurr.steps <= 10) then res := true;

if res then memo1.Lines.Add('условие 31 YES')

else memo1.Lines.Add('условие 31 NO');

end;

2: begin // условие 2

if (mCarCurr <> nil) and

(mCarCurr.steps = 10) then res := true;

if res then memo1.Lines.Add('условие 32 YES')

else memo1.Lines.Add('условие 32 NO');

end;

3: begin // условие 3

if (mCarCurr <> nil) and

(mCarCurr.isDirty) then res := true;

if res then memo1.Lines.Add('условие 33 YES')

else memo1.Lines.Add('условие 33 NO');

end;

4: begin // условие 4

if ((lineA + lineB) < 6) then res := true;

if res then memo1.Lines.Add('условие 34 YES')

else memo1.Lines.Add('условие 34 NO');

end;

5: begin // условие 5

if (lineA < lineB) then res := true;

if res then memo1.Lines.Add('условие 35 YES')

else memo1.Lines.Add('условие 35 NO');

end;

end; {case}

end;

// таблица 4

4: begin

case message.LParam of

1: begin // условие 1

if (mCarCurr <> nil) and

(mCarCurr.status = beforeW) then res := true;

if res then memo1.Lines.Add('условие 41 YES')

else memo1.Lines.Add('условие 41 NO');

end;

2: begin // условие 2

if (mCarCurr <> nil) and

(mCarCurr.steps = 20) then res := true;

if res then memo1.Lines.Add('условие 42 YES')

else memo1.Lines.Add('условие 42 NO');

end;

3: begin // условие 3

res := isCarBefore();

if res then memo1.Lines.Add('условие 43 YES')

else memo1.Lines.Add('условие 43 NO');

end;

4: begin // условие 4

if (mCarCurr <> nil) then

if ((mCarCurr.road = A) and (not signalA)) or

((mCarCurr.road = B) and (not signalB)) then res := true;

if res then memo1.Lines.Add('условие 44 YES')

else memo1.Lines.Add('условие 44 NO');

end;

5: begin // условие 5

if (mCarCurr <> nil) and

(mCarCurr.status = insideW) then res := true;

if res then memo1.Lines.Add('условие 45 YES')

else memo1.Lines.Add('условие 45 NO');

end;

6: begin // условие 6

if (mCarCurr <> nil) and

(mCarCurr.isWash) then res := true;

if res then memo1.Lines.Add('условие 46 YES')

else memo1.Lines.Add('условие 46 NO');

end;

end; {case}

end;

// таблица 5

5: begin

case message.LParam of

1: begin // условие 1

if (mCarCurr <> nil) and

(mCarCurr.steps = 50) then res := true;

if res then memo1.Lines.Add('условие 51 YES')

else memo1.Lines.Add('условие 51 NO');

end;

2: begin

res := isCarBefore();

if res then memo1.Lines.Add('условие 52 YES')

else memo1.Lines.Add('условие 52 NO');

end;

3: begin // условие 3

if not isRoadBusy() then res := true;

if res then memo1.Lines.Add('условие 53 YES')

else memo1.Lines.Add('условие 53 NO');

end;

end; {case}

end;

end; {case}

end;

if message.WParamHi = 1 then begin

// ДЕЙСТВИЕ

case message.WParamLo of

// таблица 2

2: begin

case message.LParam of

1: begin

wait();

memo1.Lines.Add('действие 21 ждать YES');

end;

end; {Case}

end;

// таблица 3

3: begin

case message.LParam of

1: begin

moving();

memo1.Lines.Add('действие 31 ехать YES');

end;

2: begin

//movingPass();

moving();

if mCarCurr <> nil then mCarCurr.steps := 30;

memo1.Lines.Add('действие 32 проехать мимо YES');

end;

3: begin

for i := 1 to 4 do moving();

carTurnToRight();

if (mCarCurr <> nil) then begin

mCarCurr.status := beforeW;

mCarCurr.road := A;

end;

inc(lineA);

memo1.Lines.Add('действие 33 повернуть на А YES');

end;

4: begin

carTurnToRight();

if (mCarCurr <> nil) then begin

mCarCurr.status := beforeW;

mCarCurr.road := B;

end;

inc(lineB);

memo1.Lines.Add('действие 34 повернуть на В YES');

end;

5: begin

nextCar();

memo1.Lines.Add('действие 35 след. по списку машина YES');

end;

end; {case}

end;

// таблица 4

4: begin

case message.LParam of

1: begin

moving();

memo1.Lines.Add('действие 41 ехать YES');

end;

2: begin

wait();

memo1.Lines.Add('действие 42 ждать YES');

end;

3: begin

movingIntoW();

memo1.Lines.Add('действие 43 заехать на мойку YES');

end;

4: begin

signalOnRed();

memo1.Lines.Add('действие 43 перекл. св. на кр. YES');

end;

5: begin

if (mCarCurr <> nil) then begin

mCarCurr.status := afterW;

mCarCurr.isDirty := false;

mCarCurr.isWash := true;

for i := 0 to 5 do moving();

if (mCarCurr.road = A) then dec(lineA)

else if (mCarCurr.road = B) then dec(lineB);

end;

memo1.Lines.Add('действие 43 выехать из мойки YES');

end;

6: begin

signalOnGreen();

memo1.Lines.Add('действие 43 перекл. св. на зел. YES');

end;

7: begin

nextCar();

memo1.Lines.Add('действие 43 след. по списку М YES');

end;

end; {case}

end;

// таблица 5

5: begin

case message.LParam of

1: begin

moving();

memo1.Lines.Add('действие 51 ехать YES');

end;

2: begin

wait();

memo1.Lines.Add('действие 52 ждать YES');

end;

3: begin

if (mCarCurr <> nil) then begin

if (mCarCurr.road = B) then

for i :=1 to 5 do moving()

else if (mCarCurr.road = A) then

for i := 1 to 8 do moving();

carTurnToRight();

mCarCurr.status := onRoad;

end;

memo1.Lines.Add('действие 53 повернуть на гл. дорогу YES');

end;

4: begin

nextCar();

memo1.Lines.Add('действие 54 след. по списку М YES');

end;

end; {case}

end;

end; {case}

res := true;

end;

if res then

message.Result := Yes

else

message.Result := No;

end

else

inherited;

end;

## Генерирование машин

procedure TfrmCarWashing.TimerComeTimer(Sender: TObject);

var

car: TCar;

begin

TimerCome.Enabled := False;

if (mCarEnd <> nil) then begin

if (mCarEnd.image.Left + mCarEnd.image.Height < 410) and

(mLengthCar < 8) then

begin

// создаем машину

car.status := onRoad;

car.steps := 0;

car.road := main;

randomize();

if (trunc(random(2)) = 0) then car.isDirty := false

else car.isDirty := true;

car.isWash := false; // этот параметр будет true только после выезда из мойки

car.moving := toLeft;

car.next := Nil;

car.image := TRotateImage.Create(self);

car.image.Parent := frmCarWashing;

car.image.Left := 460;

car.image.Top := 375;

car.image.Angle := 0;

// делаем разные картинки для машин с учетом загрязненности :)

if not car.isDirty then begin

car.image.Picture.LoadFromFile(GetCurrentDir + REDCAR\_LEFT);

end

else begin

car.image.Picture.LoadFromFile(GetCurrentDir + BLACKCAR\_LEFT);

end;

car.image.Transparent := true;

car.image.AutoSize := true;

car.image.UniqueSize := false;

addCar(car);

car.image.Visible := true;

randomize;

TimerCome.Interval := Round(random(2000)) + 1000;

end;

end

else begin

// создаем машину

car.status := onRoad;

car.steps := 0;

car.road := main;

randomize();

if (trunc(random(2)) = 0) then car.isDirty := false

else car.isDirty := true;

car.isWash := false; // этот параметр будет true только после выезда из мойки

car.moving := toLeft;

car.next := Nil;

car.image := TRotateImage.Create(self);

car.image.Parent := frmCarWashing;

car.image.Left := 460;

car.image.Top := 375;

car.image.Angle := 0;

// делаем разные картинки для машин с учетом загрязненности :)

if not car.isDirty then begin

car.image.Picture.LoadFromFile(GetCurrentDir + REDCAR\_LEFT);

end

else begin

car.image.Picture.LoadFromFile(GetCurrentDir + BLACKCAR\_LEFT);

end;

car.image.Transparent := true;

car.image.AutoSize := true;

car.image.UniqueSize := false;

addCar(car);

car.image.Visible := true;

randomize;

TimerCome.Interval := Round(random(2000)) + 1000;

end;

TimerCome.Enabled := True;

end;

## Движение

procedure TfrmCarWashing.moving();

begin

if (mCarCurr <> nil) then begin

// в зависимости от направления движения, передвигаем машину

case mCarCurr.moving of

toLeft: begin

mCarCurr.image.Left := mCarCurr.image.Left - step;

if (mCarCurr.image.Left >= 340) and

(mCarCurr.image.Left <= 360) and

(mCarCurr.image.Top > 200) then mCarCurr.steps := 10

else if (mCarCurr.image.Left <= 150) and

(mCarCurr.image.Top < 200) and

(mCarCurr.road = B) then carTurnToLeft()

else if (mCarCurr.image.Left <= 200) and

(mCarCurr.image.Top < 200) and

(mCarCurr.road = A) then carTurnToLeft()

else if (mCarCurr.image.Left <= 0) then delCar();

end;

toUp: begin

mCarCurr.image.Top := mCarCurr.image.Top - step;

if (mCarCurr.image.Top <= 50) and

(mCarCurr.status = afterW) and

(mCarCurr.road = B) then carTurnToLeft()

else if (mCarCurr.image.Top <= 90) and

(mCarCurr.status = afterW) and

(mCarCurr.road = A) then carTurnToLeft()

else if (mCarCurr.status = beforeW) and

(mCarCurr.image.Top >= 210) and

(mCarCurr.image.Top <= 230) then mCarCurr.steps := 20;

end;

toDown: begin

mCarCurr.image.Top := mCarCurr.image.Top + step;

if (mCarCurr.image.Top >= 280) then mCarCurr.steps := 50;

end;

end; {Case}

mCarCurr.image.Repaint;

sleep(20);

end;

end;

## Повороты

{

Поворот машины направо

}

procedure TfrmCarWashing.carTurnToRight();

var

n: integer;

begin

n := 5;

Case mCarCurr.moving of

toLeft: begin

while mCarCurr.image.Angle > -90 do begin

if mCarCurr.image.Angle > -25 then

mCarCurr.image.Left := mCarCurr.image.Left - n;

mCarCurr.image.Top := mCarCurr.image.Top - round(n/2);

mCarCurr.image.Angle := mCarCurr.image.Angle - n;

mCarCurr.image.Repaint;

sleep(10);

end;

mCarCurr.moving := toUp;

end;

toDown: begin

while mCarCurr.image.Angle > 0 do begin

if mCarCurr.image.Angle > 45 then

mCarCurr.image.Left := mCarCurr.image.Left - n;

mCarCurr.image.Top := mCarCurr.image.Top + round(n/2);

mCarCurr.image.Angle := mCarCurr.image.Angle - n;

mCarCurr.image.Repaint;

sleep(10);

end;

mCarCurr.moving := toLeft;

end;

end; {Case}

end;

{

Поворот машины налево

}

procedure TfrmCarWashing.carTurnToLeft();

var

n: integer;

begin

n := 5;

Case mCarCurr.moving of

toUp: begin

while mCarCurr.image.Angle < 0 do begin

if mCarCurr.image.Angle < -45 then

mCarCurr.image.Left := mCarCurr.image.Left - n;

mCarCurr.image.Top := mCarCurr.image.Top - round(n/2);

mCarCurr.image.Angle := mCarCurr.image.Angle + n;

mCarCurr.image.Repaint;

sleep(10);

end;

mCarCurr.moving := toLeft;

end;

toLeft: begin;

while mCarCurr.image.Angle < 90 do begin

if mCarCurr.image.Angle < 45 then

mCarCurr.image.Left := mCarCurr.image.Left - n;

mCarCurr.image.Top := mCarCurr.image.Top + round(n/2);

mCarCurr.image.Angle := mCarCurr.image.Angle + n;

mCarCurr.image.Repaint;

sleep(10);

end;

mCarCurr.moving := toDown;

end;

end; {Case}

end;

## Переключение светофоров

{

Переключает светофор на зеленый

}

procedure TfrmCarWashing.signalOnGreen();

begin

if (mCarCurr <> nil) and

(mCarCurr.road = A) then

begin

signalA := true;

svLeft.Picture.LoadFromFile(GetCurrentDir + GREENSV);

end

else if (mCarCurr <> nil) and

(mCarCurr.road = B) then

begin

signalB := true;

svRight.Picture.LoadFromFile(GetCurrentDir + GREENSV);

end;

end;

{

Переключает светофор на красный

}

procedure TfrmCarWashing.signalOnRed();

begin

if (mCarCurr <> nil) and

(mCarCurr.road = A) then

begin

signalA := false;

svLeft.Picture.LoadFromFile(GetCurrentDir + REDSV);

end

else if (mCarCurr <> nil) and

(mCarCurr.road = B) then

begin

signalB := false;

svRight.Picture.LoadFromFile(GetCurrentDir + REDSV);

end;

end;

## Определение занята ли дорога, есть ли машина впереди

{

возвращает результат - занята ли дорога

}

function isRoadBusy(): boolean;

var

carC: PCar;

r: boolean;

begin

r := false;

with frmCarWashing do begin

if (mCars <> nil) then begin

carC := mCarCurr;

while carC <> mCarEnd do begin

if (carC.steps = 30) and

(carC.image.Left <= 250) then

begin

r := true;

break;

end;

carC := carC.next;

end;

if (mCarEnd.steps = 30) and

(carC.image.Left <= 250) then r := true;

end;

end;

isRoadBusy := r;

end;

{

смотрит есть ли машина впереди

}

function isCarBefore(): boolean;

var

carC: PCar;

topPos: integer;

r: boolean;

begin

r := false;

with frmCarWashing do begin

if (mCarCurr <> nil) and (mLengthCar > 1) then begin

topPos := mCarCurr.image.Top;

// смотрим куда движется машинка

case mCarCurr.moving of

toUp: begin

carC := mCars;

while carC <> mCarCurr do begin

if (carC.moving = toUp) and

(carC.road = mCarCurr.road) and

(carC.image.Top + carC.image.Height >= topPos - 10) then

r := true;

carC := carC.next;

end;

end;

toDown: begin

carC := mCars;

while carC <> mCarCurr do begin

if (carC.moving = toDown) and

(carC.road = mCarCurr.road) and

(carC.image.Top - 15 <= topPos + mCarCurr.image.Height) then

r := true;

carC := carC.next;

end;

end;

end; {case}

end; {if "not nil"}

end; {with}

isCarBefore := r;

end;

## Прочие процедуры работы со списком машин

{

Добавление машины в список

}

procedure TfrmCarWashing.addCar(aCar: TCar);

var car: PCar;

begin

New(car);

car^ := aCar;

car^.next := Nil;

if mCars <> Nil then begin

mCarEnd.next := car;

mCarEnd := car;

inc(mLengthCar);

end

else begin

mCars := car;

mCarEnd := car;

mCarCurr := car;

mCurrentCar := 1;

mLengthCar := 1;

end;

end;

{

Удаляет текущую машину

}

procedure TfrmCarWashing.delCar(); // удалить и перейти к следующему

var APrev, ANext: PCar;

begin

if mCars <> Nil then

begin

if mCars = mCarCurr then // первый

begin

mCars := mCars.next;

dec(mLengthCar);

mCarCurr.image.Free;

Dispose(mCarCurr);

mCarCurr := mCars;

if mLengthCar = 0 then

begin

mCurrentCar := 0;

mCarEnd := mCars;

end;

end

else // не первый

begin

APrev := mCars;

while APrev.next <> mCarCurr do APrev := APrev.Next;

if mCarCurr = mCarEnd then // последний

begin

dec(mLengthCar);

mCurrentCar := 1;

APrev.next := Nil;

mCarCurr.image.Free;

Dispose(mCarCurr);

mCarCurr := mCars;

mCarEnd := APrev;

end

else // в середине

begin

ANext := mCarCurr.next;

APrev.next := ANext;

mCarCurr.image.Free;

Dispose(mCarCurr);

mCarCurr := ANext;

dec(mLengthCar);

end;

end;

end;

end;

{

Переход к следующей машине по списку

}

procedure TfrmCarWashing.nextCar();

begin

if mCars <> nil then

if mCarCurr.next <> nil then

begin

mCarCurr := mCarCurr.next;

inc(mCurrentCar);

end

else

begin

mCarCurr := mCars;

mCurrentCar := 1;

end;

end;

# Примеры работы программы

|  |  |
| --- | --- |
| Screenshot-45.jpg | Screenshot-46.jpg |
| Screenshot-51.jpg | Screenshot-49.jpg |