ГБОУ гимназия №1505

Московская городская педагогическая гимназия-лаборатория

**ПРОЕКТ**

**Разработка алгоритма движения поездов в метрополитене**

**и его реализация на примере компьютерной программы в среде ООП Delphi 2010**

*Руководитель проекта*: ученик 8 «Б» класса

Теплухин Рустам

*Консультант*: Мокин Евгений Анатольевич

Москва

2015

**Содержание**

Введение...................................................................................................................2

Глава I. Разработка алгоритма движения поездов...............................................3

Глава II. Реализация алгоритма в среде ООП Delphi 2010..................................8

Заключение.............................................................................................................10

Литература.............................................................................................................11

Приложение............................................................................................................12

**Введение**

Метрополитен является важнейшим транспортным средством, позволяющим эффективно решить проблему перевозок пассажиров, особенно в больших городах. Однако поток людей в метрополитене в час пик огромен, а каждая ошибка машиниста может привести к большему времени ожидания поезда. Чем больше время ожидания, тем больше людей скапливаются на станциях, и это может вызывать множество проблем. Создание нового алгоритма позволит упорядочить движение составов, сократить интервал между поездами и, следовательно, увеличить количество перевезённых пассажиров.

Целью проекта является разработка алгоритма движения поездов с минимально возможным интервалом в метрополитене и реализация данного алгоритма на примере компьютерной программы в среде ООП Delphi 2010.

Перед выполнением работы были поставлены следующие задачи:

* Изучение подобранной литературы по некоторым разделам программирования на языке Delphi в среде ООП Delphi 2010;
* Разработка алгоритма движения поездов с соблюдением минимально возможного интервала;
* Создание компьютерной программы-макета, демонстрирующей реализацию данного алгоритма в среде ООП Delphi 2010;
* Подготовка отчета о проделанной работе;
* Создание презентации результатов проектной деятельности в бумажной и электронной формах.

В ходе работы над проектом использовались теоретические методы: анализ литературы, обобщение информации, ее интерпретация. Путем рассуждений были сформулированы соответствующие выводы.

Результатами работы над проектом являются разработанный алгоритм движения поездов и его иллюстрация на примере компьютерной программы-макета, а также письменный отчет о проделанной работе.

**Глава I. Разработка алгоритма движения поездов.**

Перед созданием алгоритма был создан список требований к модели линии метрополитена:

1). Линия является кольцом;

2). Поезда двигаются только в одном направлении;

3). Участок, преодолеваемый поездом, будем считать итерацией движения поезда, длительность которой в модели будет составлять 1 секунду;

4). Каждая станция является набором из трех итераций (прибытие поезда на станцию, посадка и высадка пассажиров, отправление поезда от станции).

В ходе выполнения работы было решено, что функция вычисления интервала времени движения между поездами будет зависеть от двух входных параметров: количество участков пути, каждый из которых проезжается поездом за одно и то же время, и количество поездов на линии.

Затем был разработан алгоритм вычисления интервалов между поездами. Для этого был рассмотрен общий случай:

По умолчанию на линии точно будет один поезд. Пользователь перед стартом и во время пауз сможет менять количество поездов на линии. В зависимости от количества подвижных составов, эти поезда получают свои начальные координаты таким образом, чтобы число итераций между каждой парой было равным. Выполним следующие действия:

1). Так как количество итераций на станции равно трем, то получаем, что число итераций между двумя поездами должно быть не меньше трех (первый поезд только уехал от станции, а следом тут же на станцию въезжает второй поезд). В итоге получаем, что максимальное число поездов для любой схемы с тремя итерациями на станции равно *N div* 3, где *N* – протяженность линии (число участков линии), *div* – оператор получения целого от деления в Delphi. Используем переменную *count* (количество поездов на линии), в которую запишем число поездов на линии от 1 до *N div* 3.

2). Определим целое число итераций между поездами и запишем результат в массив интервалов поездов:

for i := 0 to count - 2 do

trainArrayDistance[i] := N div count;

где *i* – счетчик цикла, *N* – число участков линии, *count* – количество поездов на линии, *trainArrayDistance[]* – массив интервалов между поездами, причем элемент *trainArrayDistance[0]* соответствует числу итераций между первым и вторым составами (от главного вагона первого состава до главного вагона второго состава), а последний элемент массива *trainArrayDistance[count - 2]* соответствует дистанции между предпоследним и последним составами;

3). Определим остаток итераций *p*, который оказался между последним и первым составами в дополнение к основному интервалу:

p := N mod count;

4). Увеличим первые *p* элементов массива интервалов *trainArrayDistance[]* на единицу, тем самым распределим остаток из дистанции между последним и первым поездом равномерно в интервалах других поездов:

for i:= 0 to p - 1 do

trainArrayDistance[i] := trainArrayDistance[i] + 1;

5). Сформируем массив стартовых координат для поездов на линии, в соответствии с которыми расстояния (в итерациях) между ними будут равными (примерно равными в случае *p>*0). Для этого присвоим первому поезду координату начала линии, а для последующих поездов просуммируем координату предыдущего поезда и соответствующий интервал из массива *trainArrayDistance[]*:

trainArray[0] := 1;

for i := 1 to count - 1 do

trainArray[i] :=

trainArray[i - 1] + trainArrayDistance[i - 1];

Таким образом, получим конечные координаты поездов в количестве *count* штук на линии с протяженностью в *N* участков. Среднее время интервала движения поездов будет равняться , а количество поездов на линии будет варьироваться от 1 до *N div* 3.

Рассмотрим частный случай: *N* = 48:

1). Диапазон количества поездов на линии: от 1 до 16;

2). Получим массивы интервалов *trainArrayDistance[]* для всех случаев *count:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| 0 |  | **24** | **16** | **12** | **9** | **8** | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 1 |  |  | **16** | **12** | **9** | **8** | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 2 |  |  |  | **12** | **9** | **8** | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 3 |  |  |  |  | **9** | **8** | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 4 |  |  |  |  |  | **8** | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 5 |  |  |  |  |  |  | **6** | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 6 |  |  |  |  |  |  |  | **6** | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 7 |  |  |  |  |  |  |  |  | **5** | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 8 |  |  |  |  |  |  |  |  |  | **4** | **4** | **4** | **3** | **3** | **3** | **3** |
| 9 |  |  |  |  |  |  |  |  |  |  | **4** | **4** | **3** | **3** | **3** | **3** |
| 10 |  |  |  |  |  |  |  |  |  |  |  | **4** | **3** | **3** | **3** | **3** |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** | **3** | **3** |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** | **3** |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** |

3). Вычислим остатки для всех вариантов значений *count*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *count* | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| *p* |  | **0** | **0** | **0** | **3** | **0** | **6** | **0** | **3** | **8** | **4** | **0** | **9** | **6** | **3** | **0** |

4). Обновим таблицу массивов *trainArrayDistance[]* – добавим единицы в элементы соответствующих массивов (в каждом массиве их *p–*1):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| 0 |  | **24** | **16** | **12** | **10** | **8** | **7** | **6** | **6** | **5** | **5** | **4** | **4** | **4** | **4** | **3** |
| 1 |  |  | **16** | **12** | **10** | **8** | **7** | **6** | **6** | **5** | **5** | **4** | **4** | **4** | **4** | **3** |
| 2 |  |  |  | **12** | **10** | **8** | **7** | **6** | **6** | **5** | **5** | **4** | **4** | **4** | **4** | **3** |
| 3 |  |  |  |  | **9** | **8** | **7** | **6** | **5** | **5** | **5** | **4** | **4** | **4** | **3** | **3** |
| 4 |  |  |  |  |  | **8** | **7** | **6** | **5** | **5** | **4** | **4** | **4** | **4** | **3** | **3** |
| 5 |  |  |  |  |  |  | **7** | **6** | **5** | **5** | **4** | **4** | **4** | **4** | **3** | **3** |
| 6 |  |  |  |  |  |  |  | **6** | **5** | **5** | **4** | **4** | **4** | **3** | **3** | **3** |
| 7 |  |  |  |  |  |  |  |  | **5** | **5** | **4** | **4** | **4** | **3** | **3** | **3** |
| 8 |  |  |  |  |  |  |  |  |  | **4** | **4** | **4** | **4** | **3** | **3** | **3** |
| 9 |  |  |  |  |  |  |  |  |  |  | **4** | **4** | **3** | **3** | **3** | **3** |
| 10 |  |  |  |  |  |  |  |  |  |  |  | **4** | **3** | **3** | **3** | **3** |
| 11 |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** | **3** | **3** |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** | **3** |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** | **3** |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **3** |

5). Сформируем таблицу массивов стартовых координат поездов на линии *trainArray[]*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| 0 | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| 1 |  | **25** | **17** | **13** | **11** | **9** | **8** | **7** | **7** | **6** | **6** | **5** | **5** | **5** | **5** | **4** |
| 2 |  |  | **33** | **25** | **21** | **17** | **15** | **13** | **13** | **11** | **11** | **9** | **9** | **9** | **9** | **7** |
| 3 |  |  |  | **37** | **31** | **25** | **22** | **19** | **19** | **16** | **16** | **13** | **13** | **13** | **13** | **10** |
| 4 |  |  |  |  | **40** | **33** | **29** | **25** | **24** | **21** | **21** | **17** | **17** | **17** | **16** | **13** |
| 5 |  |  |  |  |  | **41** | **36** | **31** | **29** | **26** | **25** | **21** | **21** | **21** | **19** | **16** |
| i | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| 6 |  |  |  |  |  |  | **43** | **37** | **34** | **31** | **29** | **25** | **25** | **25** | **22** | **19** |
| 7 |  |  |  |  |  |  |  | **43** | **39** | **36** | **33** | **29** | **29** | **28** | **25** | **22** |
| 8 |  |  |  |  |  |  |  |  | **44** | **41** | **37** | **33** | **33** | **31** | **28** | **25** |
| 9 |  |  |  |  |  |  |  |  |  | **45** | **41** | **37** | **37** | **34** | **31** | **28** |
| 10 |  |  |  |  |  |  |  |  |  |  | **45** | **41** | **40** | **37** | **34** | **31** |
| 11 |  |  |  |  |  |  |  |  |  |  |  | **45** | **43** | **40** | **37** | **34** |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  | **46** | **43** | **40** | **37** |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  | **46** | **43** | **40** |
| 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **46** | **43** |
| 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **46** |

В итоге, получены все возможные варианты расположения поездов на схеме при количестве участков *N* = 48 и изменении количества поездов на линии от 1 до 16.

Стоит отметить, что у данного алгоритма есть свои плюсы и минусы. К плюсам можно отнести полезность данного алгоритма, т.к. в соответствии с таблицей можно понять, что в час пик необязательно выпускать максимальное количество составов, ибо разница в ожидании поезда может оказаться не очень существенной. К минусам можно отнести отсутствие учета внештатных ситуаций (поломка поезда) или переходных процессов (вывод на линию или снятие с линии поезда). Однако первый минус незначителен, так как в Московском метрополитене поезда перед входом на пути тщательно осматриваются и проверяются, и, следовательно, риск поломки сводится к минимуму.

**Глава II. Реализация алгоритма в среде ООП Delphi 2010.**

Переходя к практической части проекта, была поставлена цель – создать компьютерную программу-макет и продемонстрировать разработанный алгоритм на примере частного случая, рассмотренного в главе I.

Для создания компьютерной программы была использована студия разработки Embarcadero RAD Studio 2010 со средой разработки Delphi 2010. На мой взгляд, именно эта среда ООП позволяет создавать и использовать удобный графический интерфейс для реализации интерактивного диалога пользователя с работающим приложением, а приложение, в свою, очередь, помогает посредством визуализации процессов облегчить понимание разработанного алгоритма.

Первым шагом в разработке программы-макета было создание формы программы. Для этого при создании формы программы использовались следующие компоненты:

* TButton – кнопка. В программе реализованы кнопки «Старт» – начало движения поездов, «Пауза» – остановка движения поездов, и «Справка» – вывода модального окна с информацией о схеме линии, правилах пользования программой и ее создателе;
* TLabel – текстовое поле. В программе с помощью этого компонента создавались подсказки для пользователей;
* TShape – объект «фигура». С помощью данного компонента были созданы схема пути движения поездов (линии и станции) и контур таблицы;
* TTimer – таймер, который вводит необходимую задержку между выполнением тех или иных действий. В данной программе таймер создает задержку между итерациями движения поездов;
* TTrackBar – визуальный элемент управления в виде ползунка, который можно перемещать клавишами или курсором мыши во время выполнения. С помощью данного компонента в программе можно менять количество поездов на линии (доступно во время паузы работы программы);
* TMemo – размещение многострочного текста. В данной программе использовался для вывода справки о программе.

В итоге программа обрела следующий вид (Прил. 2).

Вторым шагом являлось написание программного кода. Документированный код программы находится в Приложении (Прил. 1).

В результате была создана компьютерная программа-макет, демонстрирующая частный случай работы алгоритма по изменяемому параметру *count* (количество поездов на линии) и заданному по умолчанию параметру *N* = 48 (протяженность пути).

**Заключение**

В ходе работы над проектом была достигнута основная цель – разработка алгоритма движения поездов с минимально возможным интервалом в метрополитене и реализация данного алгоритма на примере компьютерной программы в среде ООП Delphi 2010, и были выполнены поставленные перед началом работы задачи:

* Изучение подобранной литературы по некоторым разделам программирования на языке Delphi в среде ООП Delphi 2010;
* Разработка алгоритма движения поездов с соблюдением минимально возможного интервала;
* Создание компьютерной программы-макета, демонстрирующей реализацию данного алгоритма в среде ООП Delphi 2010;
* Подготовка отчета о проделанной работе;
* Создание презентации результатов проектной деятельности в бумажной и электронной формах.

Перспективы данного проекта заключаются в учете таких факторов, как введение нового подвижного состава на линию или снятие его с линии (въезд в оборотный тупик или отправка в электродепо) и возникновение внештатных ситуаций. Также стоит отметить и возможность улучшения компьютерной программы до уровня полнофункциональной программы: наличие возможности построения конкретного количества станций, установки типа линии и ее конкретной протяженности, добавления съездов с линии в тупики или электродепо.

**Литература**

1. Рапаков Г.Г. **Turbo Pascal для студентов и школьников** / Г.Г. Рапаков, С.Ю. Ржеуцкая. – СПб.: БХВ-Петербург, 2011. - 352 с.: ил.
2. Интернет-портал. **Уроки Delphi начинающим с нуля.** http://www.delphi-manual.ru/. Ссылка действительна на 24.12.2015.
3. Интернет-портал. **Описание компонентов Delphi.** http://helpdelphi.ru/. Ссылка действительна на 24.12.2015.
4. Интернет-форум. **Delphi Sources.** http://www.delphisources.ru/forum/. Ссылка действительна на 24.12.2015.

**Приложение**

*Прил. 1.* Код компьютерной программы-макета.

Файл *Unit1.pas*:

**unit** Unit1; *// Файл кода программы для первой формы.*

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes, Graphics,

Controls, Forms, Dialogs, ExtCtrls, StdCtrls, ComCtrls;

**type**

TForm1 = **class**(TForm)

Timer1: TTimer;

TrackBar1: TTrackBar;

Button1, Button2, Button3: TButton;

Shape1, Shape2, Shape3, Shape4, Shape5, Shape6, Shape7,

Shape8, Shape9, Shape10, Shape11, Shape12, Shape13,

Shape14, Shape15, Shape16, Shape17, Shape18, Shape19,

Shape20, Shape21, Shape22, Shape23, Shape24, Shape25,

Shape26, Shape27, Shape28, Shape29, Shape30, Shape31,

Shape32, Shape33, Shape34, Shape35, Shape36, Shape37,

Shape38, Shape39, Shape40, Shape41, Shape42, Shape43,

Shape44, Shape45, Shape46, Shape47, Shape48, Shape49,

Shape50, Shape51, Shape52, Shape53, Shape54, Shape55,

Shape56, Shape57, Shape58, Shape59, Shape60, Shape61,

Shape62, Shape63, Shape64, Shape65, Shape66, Shape67,

Shape68, Shape69, Shape70: TShape;

Label1, Label2, Label3, Label4, Label5, Label6, Label7,

Label8, Label9, Label10, Label11, Label12, Label13,

Label14, Label15, Label16, Label17, Label18, Label19,

Label20, Label21, Label22, Label23, Label24, Label25,

Label26, Label27, Label28, Label29, Label30, Label31,

Label32, Label33, Label34, Label35, Label36, Label37,

Label38, Label39, Label40, Label41: TLabel;

**procedure** FormCreate(Sender: TObject);

**procedure** FormClose(Sender: TObject; **var** Action:

TCloseAction);

**procedure** Button1Click(Sender: TObject);

**procedure** Button2Click(Sender: TObject);

**procedure** TrackBar1Change(Sender: TObject);

**procedure** Button3Click(Sender: TObject);

**private**

**public**

**procedure** Timer1Event(Sender: TObject);

**procedure** TrainGenerator();

**end**;

**var**

Form1: TForm1;

count: Integer; *//* *Переменная, отвечающая за количество*

*// поездов на линии.*

flagNoTrains: Boolean; *//* *Переменная-флажок, требуется для*

*// отделения процесса перерисовки*

*// поездов при их движении (false,*

*// по умолчанию) от процесса*

*// перерисовки поездов*

*// при перемещении ползунка.*

trainArray, trainArrayDistance *//* *Динамические массивы*

: **Array of** Integer; *//* *координат поездов и*

*//* *интервалов между поездами.*

**implementation**

**uses** Unit2; *// Установка связи со второй формой.*

{$R \*.dfm}

*// Кнопка «Старт».*

**procedure** TForm1.Button1Click(Sender: TObject);

**begin**

Timer1.Enabled := true; *//* *Включение таймера.*

Button1.Enabled := false; *// Кнопка «Старт» недоступна.*

Button1.Visible := false; *// Кнопка «Старт» невидима.*

Button2.Visible := true; *// Кнопка «Пауза» видима.*

Button2.Enabled := true; *// Кнопка «Пауза» доступна.*

TrackBar1.Enabled := false; *// Ползунок недоступен.*

**end**;

*// Кнопка «Пауза».*

**procedure** TForm1.Button2Click(Sender: TObject);

**begin**

Timer1.Enabled := false; *//* *Отключение таймера.*

Button2.Enabled := false; *// Кнопка «Пауза» недоступна.*

Button2.Visible := false; *// Кнопка «Пауза» невидима.*

Button1.Visible := true; *// Кнопка «Старт» видима.*

Button1.Enabled := true; *// Кнопка «Старт» доступна.*

TrackBar1.Enabled := true; *// Ползунок доступен.*

**end**;

*// Кнопка «Справка».*

**procedure** TForm1.Button3Click(Sender: TObject);

**begin**

Form2.ShowModal(); *// Вызов окна справки в модальном окне.*

**end**;

*// Процедура, вызываемая при закрытии окна.*

**procedure** TForm1.FormClose(Sender: TObject; **var** Action:

TCloseAction);

**begin**

Timer1.Enabled := false; *//* *Отключение таймера.*

SetLength(trainArray, 0); *//* *Очистка массива*

*// координат поездов.*

SetLength(trainArrayDistance, 0); *//* *Очистка массива*

*// интервалов.*

**end**;

*// Процедура создания окна, вызываемая при запуске программы.*

**procedure** TForm1.FormCreate(Sender: TObject);

**begin**

TrackBar1.Min := 1; *// Мин. знач. полосы ползунка.*

TrackBar1.Max := 12; *// Макс. знач. полосы ползунка.*

count := TrackBar1.Position; *// Получение позиции ползунка.*

TrainGenerator(); *// Вызов процедуры генерирования*

*// координат поездов.*

Timer1.OnTimer := Timer1Event; *//* *Добавление обработчика*

*// событий для таймера –*

*// «Процедура закрашивания*

*// фигур».*

Timer1.Interval := 1000; *// Установки задержки длиной*

*// в 1 секунду.*

Timer1.Enabled := false; *//* *Отключение таймера.*

Button2.Enabled := false; *// Кнопка «Пауза» недоступна.*

Button2.Visible := false; *// Кнопка «Пауза» невидима.*

Timer1Event(Sender); *// Вызов процедуры закрашивания фигур;*

*// будет закрашена одна фигура, так как*

*// при запуске программы ползунок*

*// установлен в положении «1 поезд».*

**end**;

*// Процедура закрашивания фигур.*

**procedure** TForm1.Timer1Event(Sender: TObject);

**var**

i: Integer; *// Локальная переменная для цикла со счетчиком.*

MyComponent: TComponent; *// Локальная переменная для*

*// неопределенного компонента*

*// программы, используется*

*// для работы с фигурами схемы.*

**begin**

**for** i := 0 **to** count - 1 **do begin** *// Цикл обхода всех поездов.*

MyComponent := *// Записываем найденную по координате фигуру*

*// для редактирования ее свойств.*

Form1.FindComponent('Shape'+IntToStr(trainArray[i]));

*// Закрашиваем фигуру в черный цвет.*

TShape(MyComponent).Brush.Color := RGB(0, 0, 0);

*// Если поезд находился в конце пути, то меняем его*

*// координату на нулевую, чтобы при добавлении единицы*

*// он вернулся в начало пути.*

**if** (trainArray[i] = 48) **then**

trainArray[i] := 0;

*// Увеличиваем координату поезда на единицу.*

trainArray[i] := trainArray[i] + 1;

*// Если функция вызвана при движении поездов по таймеру,*

*// то закрашиваем прямоугольник следующей итерации в цвет*

*// поезда, в противном случае пропускаем этот условный*

*// оператор.*

**if** (flagNoTrains = false) **then begin**

MyComponent :=

Form1.FindComponent('Shape'+IntToStr(trainArray[i]));

TShape(MyComponent).Brush.Color := RGB(0, 230, 0);

**end**;

**end**;

**end**;

*// Процедура, вызываемая при изменении положения ползунка.*

**procedure** TForm1.TrackBar1Change(Sender: TObject);

**begin**

flagNoTrains := true; *// Изменение значения флажка.*

Timer1Event(Sender); *// Вызов процедуры закрашивания фигур;*

*// все фигуры станут черными, так как*

*// происходит движение ползунка,*

*// следовательно, поезда каждый раз*

*// будут рисоваться в других фигурах.*

count := TrackBar1.Position; *// Получение позиции ползунка и*

*// ее запись в переменную*

*// количества поездов.*

TrainGenerator(); *// Вызов процедуры генерирования*

*// координат поездов.*

flagNoTrains := false; *// Изменение значения флажка.*

Timer1Event(Sender); *// Вызов процедуры закрашивания фигур;*

*// происходит закраска тех фигур,*

*// координаты которых были указаны*

*// в массиве координат – это и есть*

*// начальное расположение поездов.*

**end**;

*// Процедура, реализующая алгоритм вычисления интервалов.*

**procedure** TForm1.TrainGenerator();

**var**

i: Integer; *// Локальная переменная для цикла со счетчиком.*

**begin**

SetLength(trainArray, count); *// Устанавливаем новую длину*

*// динамического массива*

*// координат поездов на линии.*

SetLength(trainArrayDistance, count - 1); *// Устанавливаем*

*// новую длину*

*// динамического*

*// массива*

*// интервалов*

*// между поездами.*

**for** i := 0 **to** count - 2 **do** *// Заполнение массива интервалов.*

trainArrayDistance[i] := 48 **div** count;

**for** i:= 0 **to** 48 **mod** count - 1 **do** *// Увеличение некоторых*

*// элементов массива*

*// интервалов на единицу*

*// при наличии остатка.*

trainArrayDistance[i] := trainArrayDistance[i] + 1;

trainArray[0] := 1; *// Установка первого поезда в начало.*

**for** i := 1 **to** count - 1 **do** *// Создание координат*

*// для остальных поездов.*

trainArray[i] := trainArray[i-1] + trainArrayDistance[i-1];

**end**;

**end**.

Файл *Unit2.pas*:

**unit** Unit2; *// Файл кода программы для второй формы.*

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes,

Graphics, Controls, Forms, Dialogs, StdCtrls;

**type**

TForm2 = **class**(TForm)

Memo1: TMemo;

**private**

**public**

**end**;

**var**

Form2: TForm2;

**implementation**

**uses** Unit1; *// Установка связи с первой формой.*

{$R \*.dfm}

**end**.

*Прил. 2.* Форма программы-макета.

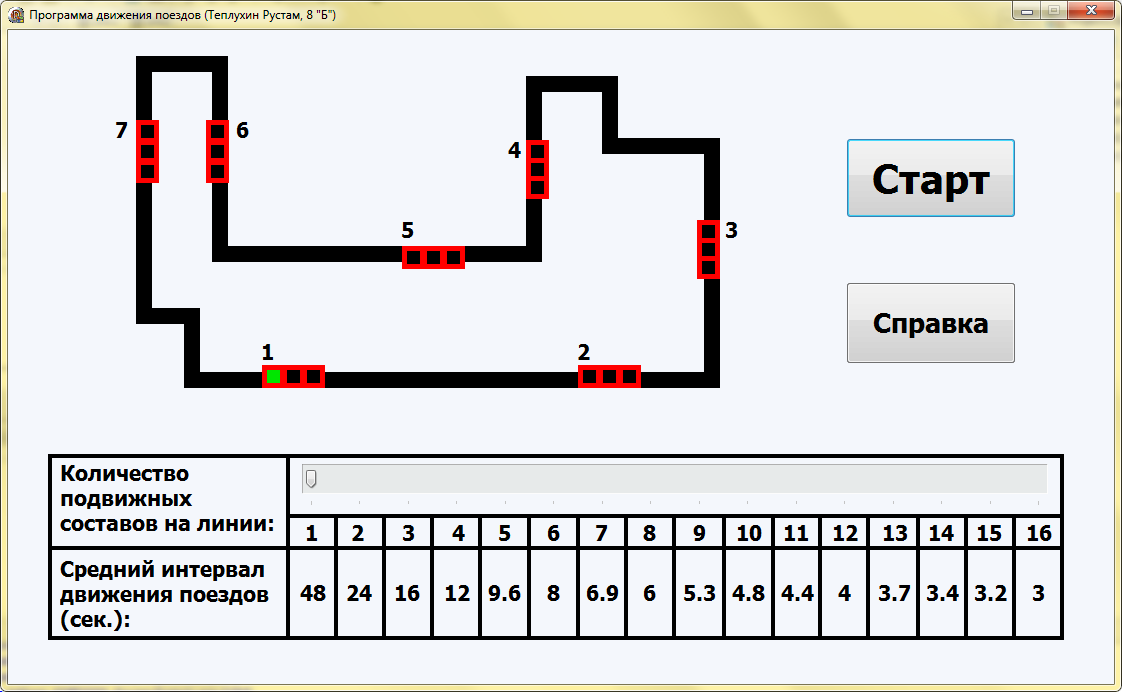


Рис. 1. Программа в режиме ожидания.

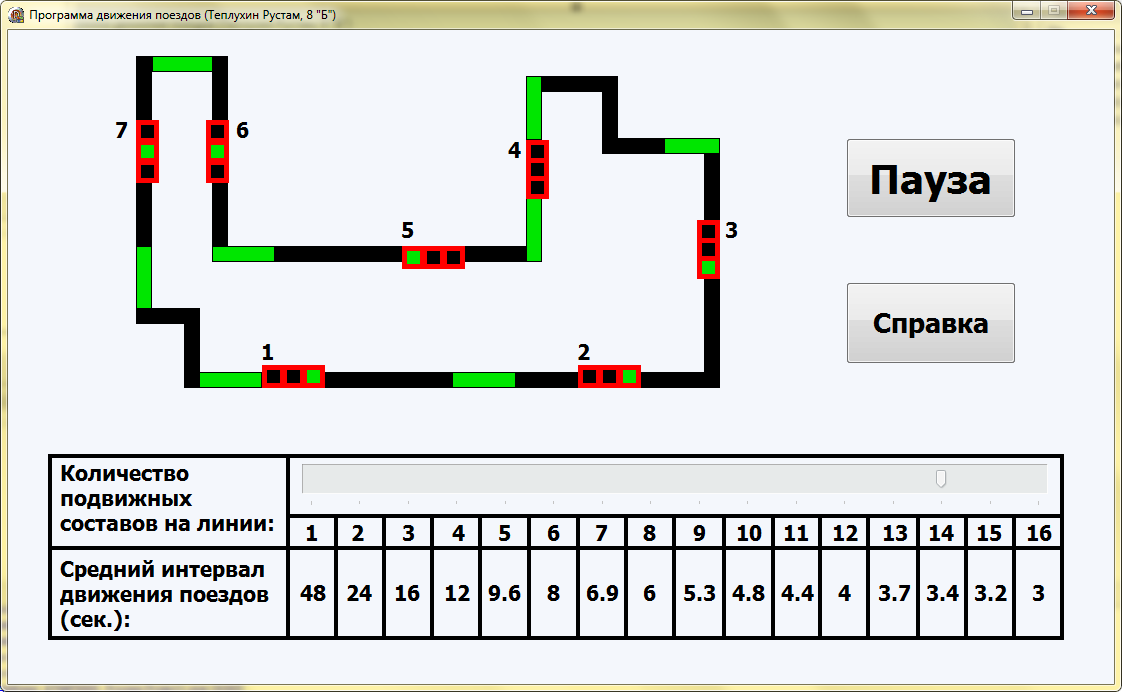


Рис. 2. Демонстрация движения поездов (при определенном количестве поездов соблюдается заданный в таблице интервал).

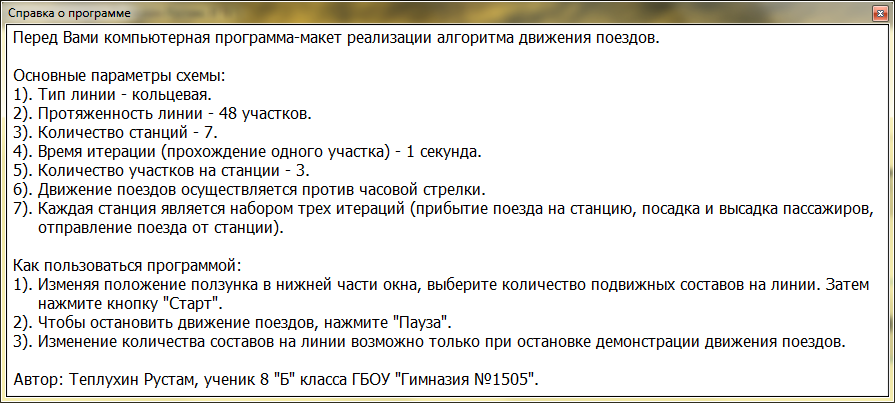


Рис. 3. Справка о программе.