РЕФЕРАТ

Расчетно-пояснительная записка 18 с., 4 рис., 4 лист., 2 ист.

РАСПИСАНИЕ, СТАНЦИЯ, .NET, C#, WINFORMS

Целью данной работы является разработка программы для составления расписания движения по железнодорожной станции.

В процессе работы были определены стуктуры данных, описывающие объекты в станции. Был введен алгоритм для управления движением. Были введены критерия для распределения поездов. Были выбраны технологии для решения поставленной задачи. Для визуализации работы алгоритма была разработана соответствующая программа.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Железнодорожный транспорт является важнейшим звеном в системе грузовых и пассажирских перевозок. Эффективное управление движением поездов на станциях является важной задачей, особенно в условиях высокой загруженности и необходимости избегать задержек. Правильное распределение поездов по доступным платформам позволяет минимизировать время ожидания и исключить ситуации, когда поезда вынуждены останавливаться перед станцией в ожидании освобождения платформы.

Целью данной работы является разработка программы для составления расписания движения по железнодорожной станции.

Для достижения поставленной цели необходимо решить следующие задачи:

— описать структуры данных, хранящие информации о поезде, пути, платформе, станции;

— определить критерии, по которым будет осуществляться выбор платформы для поезда;

— разработать алгоритм, который будет автоматически распределять поезда по платформам в зависимости от их направления;

— создать программное обеспечение, обеспечивающее демонстрации работы алгоритма;

1 Аналитический раздел

1.1 Постановка задачи

Необходимо разработать систему, которая будет автоматически управлять распределением поездов по платформам на железнодорожной станции, избегая задержек и перегрузок. Система должна оптимизировать движение поездов с учетом направления и доступных ресурсов станции.

На рисунке 1 приведены IDEF0-схемы для поставленной задачи.

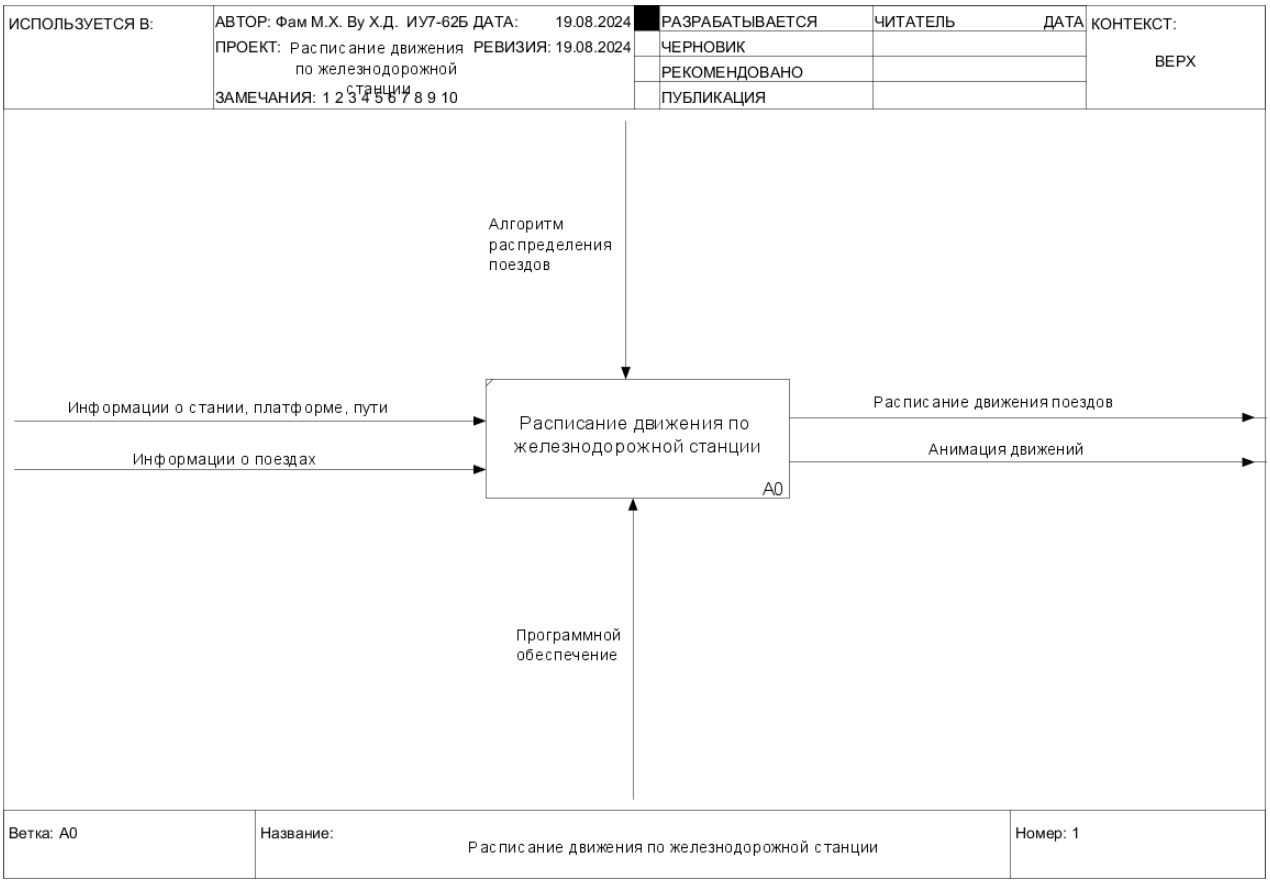

Рисунок 1 – Контекстная диаграмма (A-0)

1.2 Формализация данных

Выделяются следующие сущности, связанные с данной задачей:

— станция;

— платформа;

— путь;

— поезд;

Информация о каждой сущности проводится в таблице 1.

Таблица 1 – Сущности и их описания

Сущность	**Описание**
Станция	Название, список платформ, расписание
Платформа	ID, список путей
Путь	ID, направление, список текущих поездов
Поезд	ID, рисунок, время отправления, время прибытия, тип, ID платформы, ID пути

Вывод

В данном разделе была введена постановка задачи и проведена формализация данных.

2 Конструкторский раздел

2.1 Критерия для распределения поездов

Все поезда должны соответствовать следующим критериям:

— на одной пути интервал между двумя поездами — 10 минут;

— если все нет свободных платформ, поезд задерживается.

2.2 Разработка алгоритма

На рисунке 2 приведена схема алгоритма распределения поездов, сделанная студентом Фам Минь Хиеу.

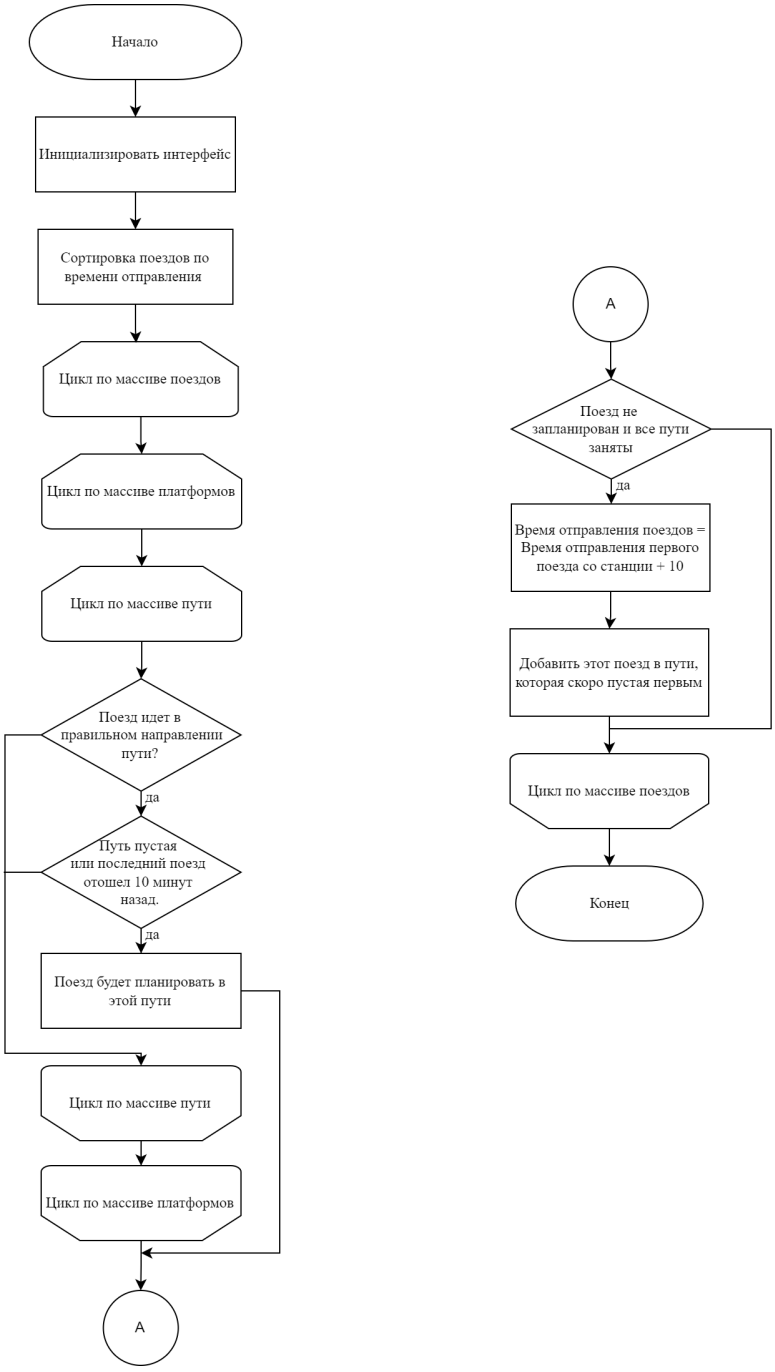

Рисунок 2 – Алгоритм распределения поездов

3 Технологический раздел

3.1 Средства реализации

Для написания данной работы был выбран язык C# [1]. Этот выбор обусловлен следующими причинами:

— C# — объектно-ориентированный язык, что соответствует методологии, выбранной для разработки программы;

— C# предоставляется обширный набор библиотек и шаблонов, что позволяет использовать готовые конструкции и значительно экономит время разработки.

В качестве среды разработки был использован Visual Studio 2022 [2]. Данный выбор обусловлен следующими факторами:

— в Visual Studio есть возможность быстрого создания интерфейса с помощью WinForms;

— Visual Studio предоставляет шаблоны, которые облегчают процесс написания и отладки проекта.

3.2 Сведения о модулях программы

Программы состоит из следующих модулей:

— Program.cs — точка входа в программу;

— Form.cs — интерфейс программы;

— Station.cs — описывает станцию;

— Platform.cs — описывает платформу;

— Track.cs — описывает путь;

— Train.cs — описывает поезд.

3.3 Реализация программы

Ниже, на листингах 1–4, приведены реализации классов, сделанны студентом By Хай Данг.

Листинг 1 – Класс Station

```
public class Station
{
    private string name;
    private List<Platform> platforms;
    private List<Train> trains;

    public string Name { get => name; set => name = value; }
    public List<Platform> Platforms { get => platforms; set => platforms
        = value; }
    public List<Train> Trains { get => trains; set => trains = value; }

    public Station(string name, List<Platform> platforms)
    {
        this.name = name;
        this.platforms = platforms;
        this.trains = new List<Train>();
    }
    public void AddTrain(Train train)
    {
        this.trains.Add(train);
    }

    public void Schedule()
    {
        foreach (var train in trains.OrderBy(t => t.DepartureTime))
        {
            TimeSpan minTime = new TimeSpan(23, 59, 59);
            Platform platformTmp = null;
            Track trackTmp = null;
            foreach (var platform in platforms)
            {
                foreach (var track in platform.Tracks)
                {
```

```
33                  if ((train.Direction.Split(' ')[0] == "Moscow" &&
                        track.Direction == "From Moscow") ||
34                  (train.Direction.Split(' ')[0] != "Moscow" &&
                        track.Direction == "To Moscow"))
35                  {
36                      if (track.CurrTrains.Count == 0 ||
37                      (train.DepartureTime.TotalMinutes -
                            track.CurrTrains.Last().
38                      DepartureTime.TotalMinutes) >= 9)
39                      {
40                          track.CurrTrains.Add(train);
41                          train.PlatformAssigned = platform.Id;
42                          train.TrackAssigned = track.Id;
43                          train.WasPlaned = true;
44                          train.setSpeed();
45                          break;
46                      }
47                      else
48                      {
49                          if (track.CurrTrains.Last().DepartureTime <=
                                minTime)
50                          {
51                              minTime =
                                    track.CurrTrains.Last().DepartureTime;
52                              platformTmp = platform;
53                              trackTmp = track;
54                          }
55                      }
56                  }
57              }
58              if (train.WasPlaned)
59              break;
60          }
61          if (!train.WasPlaned)
62          {
63              train.DepartureTime =
                    TimeSpan.FromMinutes(minTime.TotalMinutes + 10);
64              trackTmp.CurrTrains.Add(train);
65              train.PlatformAssigned = platformTmp.Id;
66              train.TrackAssigned = trackTmp.Id;
```

```
67                  train.WasPlaned = true;
68                  train.setSpeed();
69              }
70
71          }
72      }
73
74      public void UpdatePlatforms(int width)
75      {
76          foreach (var platform in platforms)
77          {
78              foreach (var track in platform.Tracks)
79              {
80                  track.CurrTrains.RemoveAll(t => t.hasArrived(width) ==
                        true);
81              }
82          }
83      }
84  }
```

Листинг 2 – Класс Platform

```
1  public class Platform
2  {
3      private int id;
4      private List<Track> tracks;
5      public int Id { get => id; set => id = value; }
6      internal List<Track> Tracks { get => tracks; set => tracks = value; }
7
8      public Platform(int id, List<Track> tracks)
9      {
10         this.id = id;
11         this.tracks = tracks;
12     }
13 }
```

Листинг 3 – Класс Track

```
1  public class Track
2  {
3      private int id;
4      private string direction;
```

```
5       private List<Train> currTrains;
6       public int Id { get => id; set => id = value; }
7       public List<Train> CurrTrains { get => currTrains; set => currTrains
            = value; }
8       public string Direction { get => direction; set => direction = value;
            }
9
10      public Track(int id, string direction)
11      {
12          this.id = id;
13          this.direction = direction;
14          this.CurrTrains = new List<Train>();
15      }
16  }
```

Листинг 4 – Класс Train

```
1   public class Train
2   {
3       private int id;
4       private PictureBox pic = new PictureBox();
5       private TimeSpan arrivalTime;
6       private TimeSpan departureTime;
7       private string direction;
8       private string type;
9       private int platformAssigned;
10      private int trackAssigned;
11      private bool hasDrawn = false;
12      private bool wasPlaned = false;
13      private int speed = 7;
14
15      private int timeStoped = 50;
16      static public int speedGlobal = 7;
17      static public Dictionary<int, Point> positions = new Dictionary<int,
            Point>
18      {
19          {1, new Point(0, 71 + 177) },
20          {2, new Point(1127, 115 + 177) },
21          {3, new Point(0, 242 + 177) },
22          {4, new Point(1127, 286 + 177) },
23          {5, new Point(0, 428 + 177) },
```

```
24          {6, new Point(1127, 472 + 177) }
25      };
26
27      public int Id { get => id; set => id = value; }
28      public TimeSpan ArrivalTime { get => arrivalTime; set => arrivalTime
            = value; }
29      public TimeSpan DepartureTime { get => departureTime; set =>
            departureTime = value; }
30      public string Direction { get => direction; set => direction = value;
            }
31      public string Type { get => type; set => type = value; }
32      public PictureBox Pic { get => pic; set => pic = value; }
33      public int PlatformAssigned { get => platformAssigned; set =>
            platformAssigned = value; }
34      public int TrackAssigned { get => trackAssigned; set => trackAssigned
            = value; }
35      public bool HasDrawn { get => hasDrawn; set => hasDrawn = value; }
36      public int Speed { get => speed; set => speed = value; }
37      public int TimeStoped { get => timeStoped; set => timeStoped = value;
            }
38      public bool WasPlaned { get => wasPlaned; set => wasPlaned = value; }
39
40      public Train(int id, TimeSpan departureTime, TimeSpan arrivalTime,
            string direction, string type)
41      {
42          this.id = id;
43          pic.BackColor = Color.Red;
44          pic.Location = new Point();
45          pic.Size = new Size(100, 38);
46          this.arrivalTime = arrivalTime;
47          this.departureTime = departureTime;
48          this.direction = direction;
49          this.type = type;
50      }
51
52      public void MoveTrain()
53      {
54          var pos = pic.Location;
55          pos.X += Speed;
56          pic.Location = pos;
```

```
57      }
58
59      public void setSpeed()
60      {
61          if (trackAssigned % 2 == 0)
62          Speed = -Speed;
63      }
64      public bool hasArrived(int width)
65      {
66          if (trackAssigned % 2 == 0)
67          return pic.Location.X <= 0;
68          else
69          return pic.Location.X >= width;
70      }
71  }
72  }
```

3.4 Демонстрация работы программы

На рисунках 3–4 приведен интерфейс программы, сделанны студентом Фам Минь Хиеу.

Рисунок 3 – Пример работы программы

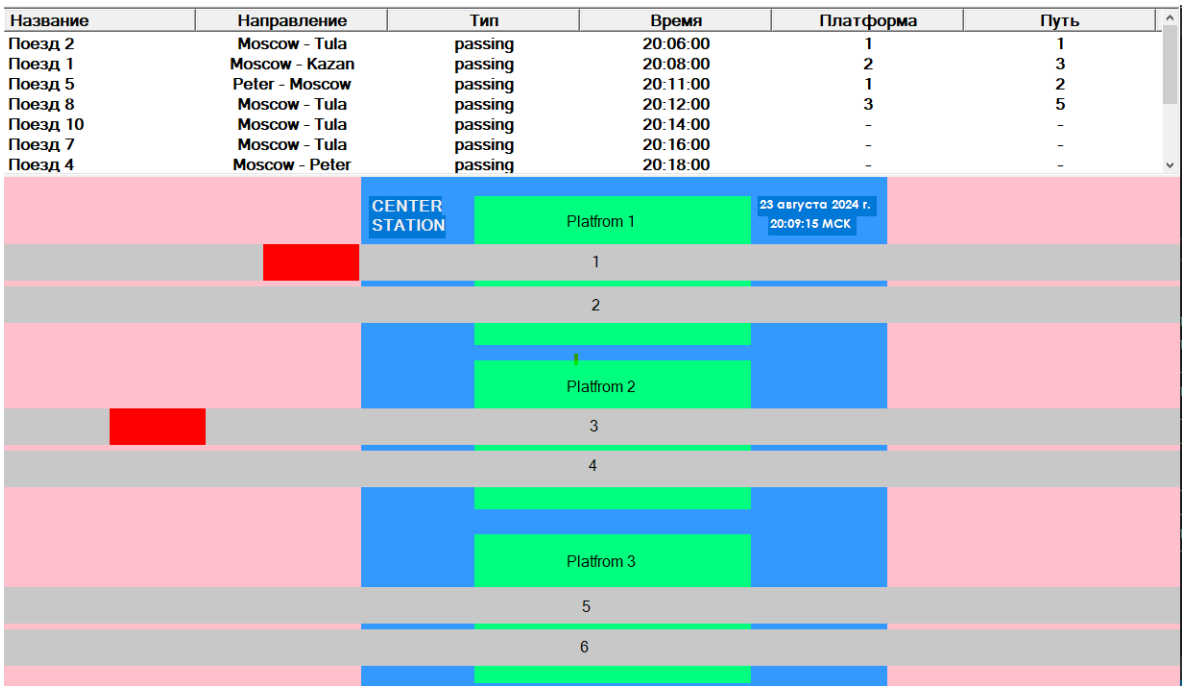

Рисунок 4 – Пример работы программы (продолжение)

ЗАКЛЮЧЕНИЕ

Цель данной работы была достигнута, то есть была разработана программа для составления расписания движения по железнодорожной станции.

Для достижение цели были решены все задачи:

— описаны структуры данных, храняющие информации о поезде, пути, платформе, станции;

— определены критерии, по которым будет осуществляться выбор платформы для поезда;

— разработан алгоритм, который будет автоматически распределять поезда по платформам в зависимости от их направления;

— создано программное обеспечение, обеспечивающее демонстрации работы алгоритма;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Документация по языку C-Sharp [Электронный ресурс]. — Режим доступа: `https://learn.microsoft.com/en-us/dotnet/csharp/` (дата обращения: 20.07.2024).

2 Visual Studio Community [Электронный ресурс]. — Режим доступа: `https://visualstudio.microsoft.com/vs/community/` (дата обращения: 27.07.2024).