УДК 004

М. В. Головенко, О. В. Толстель, Д. В. Михеенко

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПОСТРОЕНИЯ ОПТИМАЛЬНЫХ МАРШРУТОВ НА КАРТЕ С ПРЕПЯТСТВИЯМИ

Поступила в редакцию 04.05.2021 г.
Рецензия от 22.05.2021 г.

В настоящее время во всех отраслях производства заметна тенденция автоматизирования всех процессов, в частности внутренней логистики. Для оптимального использования автоматизированных или автоматических транспортных средств требуется решать задачу поиска оптимальных маршрутов на карте с препятствиями. В статье рассмотрено написание программы для поиска таких маршрутов в среде программирования Microsoft Visual Studio с использованием языка C#.

The tendency to automate all processes, in particular internal logistics, is now evident in all branches of production. The optimal use of automated or automated vehicles requires solving the problem of finding optimal routes on a map with obstacles. The article deals with writing a program for finding such routes in the Microsoft Visual Studio programming environment using the C# language.

Ключевые слова: оптимальный путь, построение маршрута, карта с препятствиями, волновой алгоритм, компьютерная программа, C#

Key words: optimal way, route construction, map with obstacles, wave algorithm, computer software, C#

Маршрут (англ. *route*) — путь следования объекта, учитывающий направление движения относительно ориентиров или координат, с указанием начальной, конечной и промежуточных точек (в случае их наличия). Маршрут из точки А в точку Б называется *оптимальным*, если он удовлетворяет выбранному критерию оптимальности (например, длина, время в пути, количество поворотов и др.) [1].

ООО «Автотор» — один из крупнейших автопроизводителей России, первым в стране начавший выпуск автомобилей иностранных марок [2]. Для автоматизации доставок в цехах «Автотора» используются автоматически управляемые тележки, которые двигаются по заданным маршрутам.

Для контроля за всем автоматически управляемым транспортом (Automatic Guided Vehicle — AGV) в цеху существует логистическая система — программа, управляющая логистикой цеха в реальном времени. Она имеет доступ к карте цеха, передвижных стоек с инструментами, особых зон и др. Автоматизированные тележки постоянно передают свои данные, такие как положение, скорость, состояние загрузки, в эту систему.

При необходимости доставить груз из одной точки в другую система назначает на это действие свободную тележку, формирует файл на создание маршрута перемещения, составляет маршрут доставки и передает его на выбранное транспортное средство, контролирует выполнение доставки. Каждую из функций выполняет отдельная подпрограмма, и результат каждой подпрограммы важен.

Однако самое большое влияние на всю перевозку оказывает маршрут [3]. Даже самая быстрая тележка при неправильном построении пути доставит груз позднее, чем медленный AGV, движущийся по самому оптимальному плану.

Для лучшего использования ресурсов предприятия и минимизации простоев оборудования требуется найти и использовать самые оптимальные маршруты доставки [1]. Решение данной задачи зависит от выбранного алгоритма поиска пути.

Существует несколько факторов, оценка которых дает ответ на вопрос, минимален ли полученный путь. Каждый фактор по-разному влияет на минимизацию расходов при производстве. Выбор приоритетного фактора также должен быть рассчитан.

В случае с доставкой деталей для сборки с использованием АТС критерием оптимальности будем считать длину маршрута, которая в конечном итоге сказывается на времени выполнения операции.

Рассмотрим задачу нахождения пути на карте с препятствиями с использованием языка программирования C#. Целью является написание программы на языке C#, которая находит оптимальный путь на карте с препятствиями, чтобы в дальнейшем использовать эту программу для решения задачи доставки.

Будем использовать следующие инструменты:

- Microsoft Visual Studio — интегрированная среда разработки от Microsoft [4] с поддержкой многих языков программирования, в частности C#;
- AutoCAD — система автоматизированного проектирования и черчения, с помощью которой можно создавать входные карты с препятствиями;
- FastStone Image Viewer — многофункциональная программа для просмотра изображений (или другая программа такого же функционала для просмотра полученных результатов);
- встроенные в среду разработки библиотеки, которые необходимы для выполнения расчетов.

Разработка интерфейса

Создание приложения началось с разработки интерфейса. Бóльшая часть окна, порядка 75 % всей площади, было выделено под рабочую область, в которой отображается чертеж.

Меню программы вынесено в правую сторону окна и занимает приблизительно 20 % от всего пространства программы. Действиями в данном меню пользователь управляет всей программой, открывает чертежи, рассчитывает карту весов и строит маршруты.

Блок ниже, которому отведено около 4—5 % площади интерфейса, выделен для подсказок. В нем содержится информация о текущем состоянии программы и ожидаемом действии. Также в него выводятся подсказки при наведении на элементы управления.

Макет интерфейса программы в упрощенном виде представлен на рисунке 1.

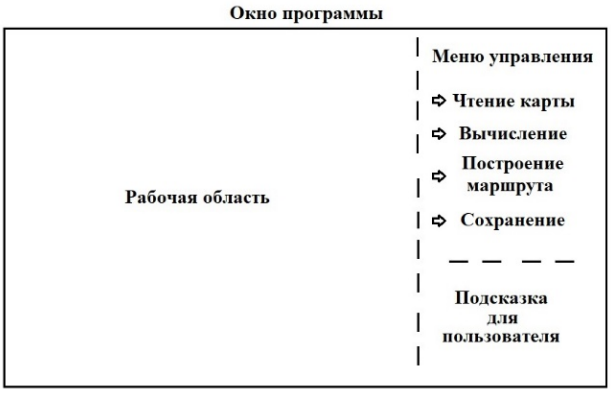


Рис. 1. Макет интерфейса программы

Для создания подобного интерфейса в Microsoft Visual Studio использовались следующие стандартные элементы управления:

— кнопки Button для взаимодействия пользователя с программой;

— контейнеры GroupBox для создания информационных блоков и быстрого управления интерфейсом;

— поля Label для вывода подсказок в соответствующие поля в окне программы;

— контейнер PictureBox для отображения чертежа на экране [5].

Получившееся стартовое окно программы показано на рисунке 2.

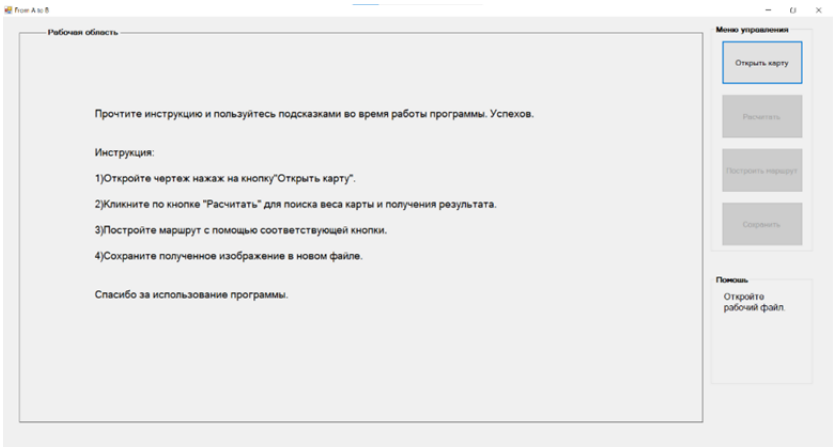


Рис. 2. Стартовое окно программы

Импорт библиотек

Перед началом необходимо импортировать все библиотеки и коллекции элементов, необходимые для написания алгоритмов и их выполнения. При написании программы использовались только встроенные в систему Windows библиотеки, что позволяет беспрепятственно запускать полученное приложение на компьютерах с такой операционной системой. Перечислим их: System; System.Collections.Generic; System.ComponentModel; System.Data; System Drawing; System.IO; System.Linq; System.Text; System.Threading.Tasks; System.Windows.Forms.

Открытие карты

Многие компьютерные программы для создания чертежей, например AutoCAD, Компас, nanoCAD, имеют возможность сохранять чертежи в форматах изображений bmp, png, jpg. Использование специализированных форматов, таких как dwg для AutoCAD, лишь усложнит задачу, так как файлы такого формата содержат дополнительные характеристики, не требуемые в задаче.

К входным чертежам были установлены строгие критерии. Черным цветом показываются непреодолимые препятствия (стены, пешеходные зоны), которые нельзя пересекать, и различные другие препятствия. Прямоугольником зеленого цвета на рисунке должно быть отмечено транспортное средство, для которого строится путь, причем размеры должны соответствовать масштабу схемы. Зона доставки на входном изображении отмечается красным прямоугольником. Фон изображения — белый. Любые другие цвета на входном файле будут рассматриваться как свободные к перемещению позиции.

Для чтения карты из файла в программе прописан вызов диалогового окна, в котором пользователь посредством проводника Windows выбирает изображение, хранящееся на компьютере и подходящее под фильтр форматов. Такое окно показано на рисунке 3.

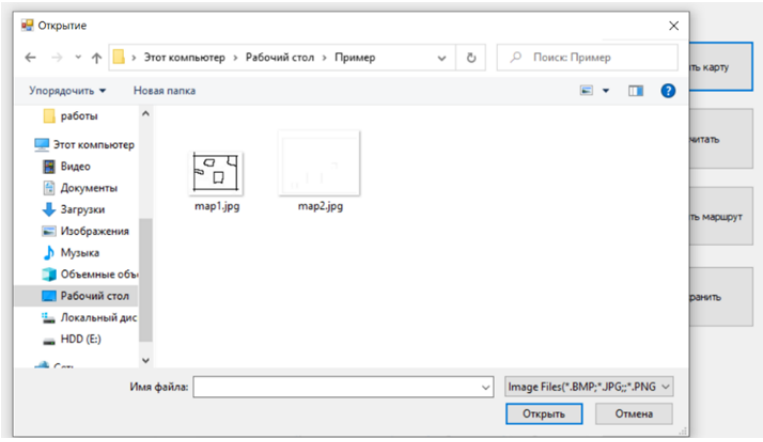


Рис. 3. Открытие файла

В коде программы вызов диалогового окна привязан к нажатию соответствующей кнопки в меню управления. Также сделана проверка на корректность открытия файла. Фрагмент кода представлен на рисунке 4.

```
ссылка: 1
private void OpenFileButton_Click(object sender, EventArgs e)   //Открытие файла
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "Image Files(*.BMP;*.JPG;;*.PNG)|*.BMP;*.JPG;*.PNG|All files (*.*)|*.*";
    if (open_dialog.ShowDialog() == DialogResult.OK) //В диалоге нажата OK
    {
        try
        {
            img = new Bitmap(open_dialog.FileName);
            MyPictureBox1.Image = img;
        }
        catch
        {
            DialogResult rezult = MessageBox.Show("Невозможно открыть выбранный файл",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            MyPictureBox1.Image = null;
        }
    }
}
```

Рис. 4. Фрагмент кода — открытие файла

После выбора файла карта размещается в рабочей области, при этом она масштабируется, чтобы пользователь видел чертеж целиком. Для этого в свойство SizeMode контейнера PictureBox было установлено значение Zoom. Результат показан на рисунке 5.

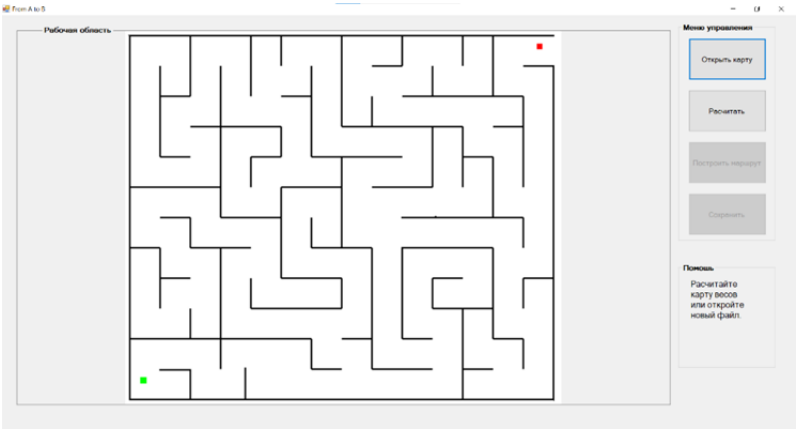


Рис. 5. Успешное открытие файла

Расчет карты весов

Следующим шагом после открытия чертежа является вычисление карты весов. Данное действие вынесено и закреплено за отдельной кнопкой. Сделано это для того, чтобы при ошибочном выборе файла не проводить вычисления — самый ресурсоемкий процесс программы.

Весами заполняются не все поля начального изображения, так как некоторые из них для этого не подходят (например, если они находятся

очень близко к границе препятствий). Поэтому сперва программа определяет положение и габариты перемещаемого объекта. В этом же цикле прохода по чертежу в коде определяется и конечное положение объекта.

После того как получены габариты объекта транспортировки, в программе появляются два значения, равные половине длины и ширины объекта. Так как маршрут будет рассчитываться для центра объекта, именно столько необходимо отступать от всех препятствий на карте.

При повторном чтении карты циклами в матрицу весов переносятся все ограничения, а также заносятся отступы, которые были описаны выше.

Получается следующее графическое представление данного процесса (рис. 6).

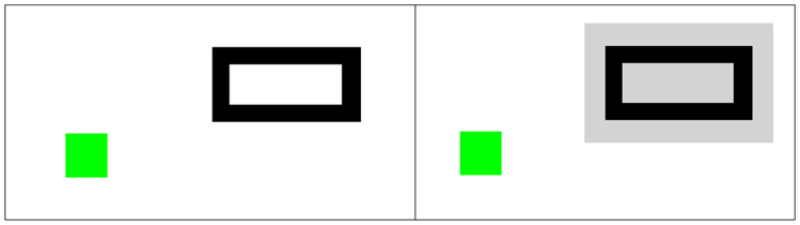


Рис. 6. Добавление отступов от препятствий

Для заполнения карты весов и определения оптимального маршрута в программе был выбран волновой алгоритм [6, с. 21—24] — метод поиска кратчайшего пути в планарном графе, основанный на поиске в ширину. Для данного метода характерен поиск путей двух видов: ортогонального (переходы в окрестность фон Неймана) и ортогонально-диагонального (переходы в окрестность Мура). Изображение окрестностей представлено на рисунке 7.

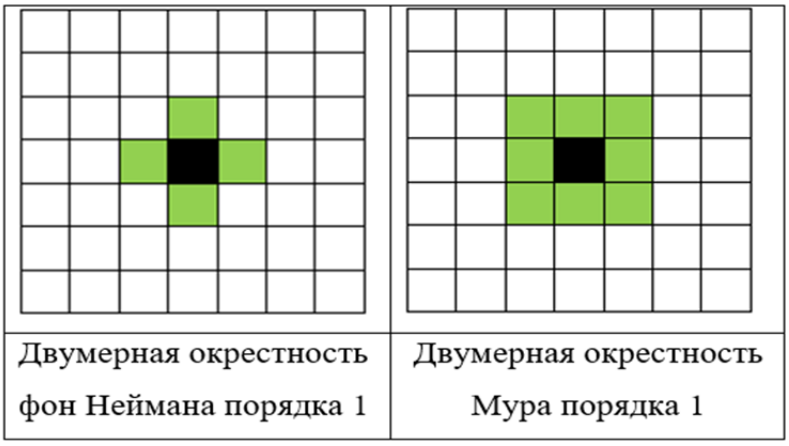


Рис. 7. Двумерные окрестности Мура и фон Неймана порядка 1

Используя окрестности Мура и придавая весу значение длины перехода, необходимо прибегнуть к матрице весов, состоящей из действительных чисел (тип double). Считая переходы по горизонтали и вертикали одинаковыми по весу и равными 1, получим, что вес перехода по диагонали равен $\sqrt{2}$. Оптимальность маршрутов, найденных в таком случае, будет увеличиваться, но также будут расти размер и вес таблицы и сложность ее заполнения. Помимо этого, такие маршруты менее предсказуемы для людей, которые будут находиться рядом с маршрутом следования транспортного средства, в результате чего могут возникать аварийные ситуации. Поэтому было принято решение использовать окрестность фон Неймана. Кроме того, при таких расчетах можно использовать матрицу весов, состоящую только из целых значений (тип integer), что уменьшит объем памяти, занимаемый матрицей, и упростит вычисления.

Для реализации волнового алгоритма в программе используется очередь Queue из системной коллекции System.Collections.Generic, в которую изначально помещается стартовая позиция. Далее, пока очередь не пуста или пока не достигнута конечная позиция, выполняется следующий алгоритм:

1) из очереди достается вершина;

2) считывается ее вес k;

3) среди соседних вершин выбираются доступные для перехода и еще не заполнение вершины;

4) выбранным в пункте 3 вершинам присваивается вес k + 1;

5) вершины заносятся в конец очереди [6; 7].

Если на каком-то этапе был присвоен вес конечной вершине, то программа завершит расчет карты весов. Если очередь опустела, но конечной вершине так и не присвоен вес, отличный от 0, это значит, что начальная и конечная точки не могут быть соединены маршрутом при таких значениях габаритов транспорта или находятся в разных компонентах связанности и пути между ними не существует. Пользователь в этом случае получит соответствующее сообщение.

Для выполнения пунктов 2—5 в программе реализована рекурсивная функция [7] Krest(ref int[,] map, Point P). Код данной функции представлен ниже.

```
void krest(ref int[,] map, Point P)
    {
      int N = map.GetLength(0); //длина карты
      int M = map.GetLength(1); //ширина карты
      int ves = map[P.x, P.y];
      if (P.x - 1 >= 0)
      {//проверка соседа слева
        if (map[P.x - 1, P.y] == 0)
        {
          map[P.x - 1, P.y] = ves + 1;
          //в очередь
          Point add = new Point();
          add.x = P.x - 1;
```

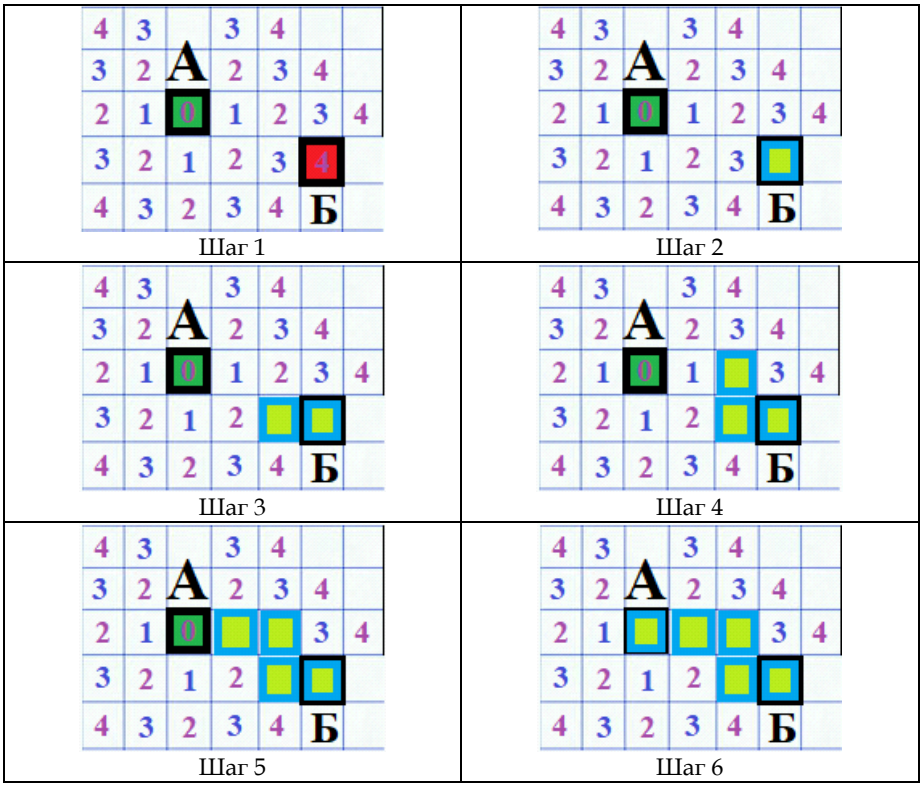
```
            add.y = P.y;
            Volna.Enqueue(add);
          }
        }
        if (P.x + 1 < N)
        {//проверка соседа справа
          if (map[P.x + 1, P.y] == 0)
          {
            map[P.x + 1, P.y] = ves + 1;
            //в очередь
            Point add = new Point();
            add.x = P.x + 1;
            add.y = P.y;
            Volna.Enqueue(add);
          }
        }
        if (P.y - 1 >= 0)
        {//проверка соседа снизу
          if (map[P.x, P.y - 1] == 0)
          {
            map[P.x, P.y - 1] = ves + 1;
            //в очередь
            Point add = new Point();
            add.x = P.x;
            add.y = P.y - 1;
            Volna.Enqueue(add);
          }
        }
        if (P.y + 1 < M)
        {//проверка соседа сверху
          if (map[P.x, P.y + 1] == 0)
          {
            map[P.x, P.y + 1] = ves + 1;
            //в очередь
            Point add = new Point();
            add.x = P.x;
            add.y = P.y + 1;
            Volna.Enqueue(add);
          }
        }
    }
```

Построение маршрута

После просчета карты пользователю необходимо нажать на кнопку построения маршрута. В данном случае маршрут строится очень просто. У конечной точки ищется сосед с весом на единицу меньше, они связываются между собой и далее от полученной точки ищется сосед с весом, отличным на единицу от нее. Это повторяется, пока не будет достигнута начальная вершина, вес которой 0.

Покажем пример построения такого маршрута длины 5 в следующей таблице.

Пример построения маршрута



Таким образом могут быть построены несколько маршрутов. Все они будут оптимальны по расстоянию и будут отличаться лишь количеством поворотов. Путь, найденный в программе, отобразится в рабочей области, на изображении появится ломаная линия синего цвета. Ниже (рис. 8) приведен пример решения несложного лабиринта, показанного на рисунке 5. После расчета карты весов и построения маршрута окно программы выглядит следующим образом.

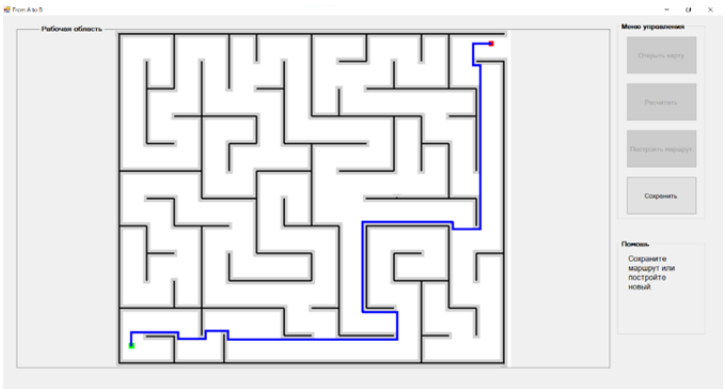


Рис. 8. Решение лабиринта

Сохранение результата

Полученный маршрут пользователь может сохранить во внешнем файле в одном из доступных форматов (bmp, png, jpg). Интерфейс сохранения реализован с помощью схожего с пунктом открытия диалогового окна SaveFileDialog. В нем предлагается ввести имя файла, выбрать его формат, а также расположение, как показано на рисунке 9. Фрагмент программного кода, которым обрабатывается нажатие на кнопку «Сохранить», представлен на рисунке 10.

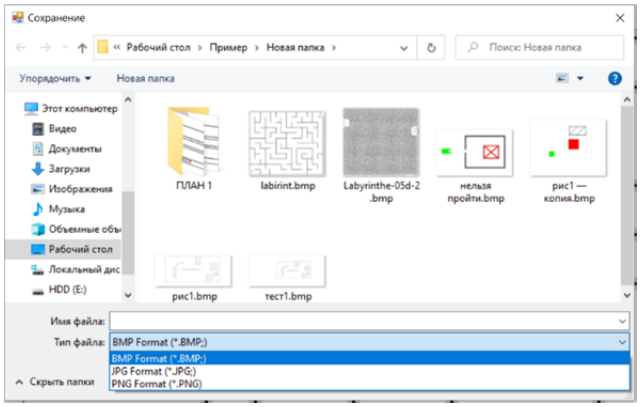


Рис. 9. Окно сохранения файла

```
private void SaveButton_Click(object sender, EventArgs e)   //Сохранение в файл
{
    SaveFileDialog save_dialog = new SaveFileDialog();
    save_dialog.DefaultExt = ".jpg";
    save_dialog.Filter = "BMP Format (*.BMP;)|*.BMP;|JPG Format (*.JPG;)|*.JPG;|PNG Format (*.PNG)|*.PNG;";
    if(save_dialog.ShowDialog() == DialogResult.OK && save_dialog.FileName.Length > 0)
    {
        try
        {
            risunok.Save(save_dialog.FileName);
            label1text(5);
            OpenFileButton.Enabled = true;
            Calculate.Enabled = false;
            Road.Enabled = true;
            SaveButton.Enabled = true;
        }
        catch
        {
            DialogResult rezult = MessageBox.Show("Невозможно сохранить файл",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Рис. 10. Фрагмент кода — нажатие кнопки «Сохранить»

После сохранения файла его можно будет открыть как в любом графическом редакторе или программе для просмотра изображений, так и в специализированных программах, например AutoCAD. Наложив полученное изображение на существующий чертеж с некоторым значением прозрачности, пользователь сможет перенести маршрут на схему, выполненную в формате dwg, указать на ней ключевые точки, рассчитать расстояния, координаты и многое другое.

Тестирование программы

Написанная программа была протестирована на чертежах компании «Автотор». Получены результаты как для новых цехов, находящихся на стадии планирования, так и для уже существующий решений. Представители компании остались довольны результатом, во время обсуждения были предложены варианты развития и усовершенствования данного программного обеспечения.

Выводы

В результате работы была написана компьютерная программа с несложным для понимания пользователем графическим интерфейсом, позволяющая по входному изображению карты с препятствиями строить оптимальные маршруты доставки в конечный пункт с учетом габаритов перемещаемого объекта. Полученное приложение представляет собой отдельный самостоятельный продукт. Кроме того, есть возможность несложным образом перестроить его для работы в виде подпрограммы для более сложной системы.

Список литературы

1. *Zhang Q., Ding X., Zhou J.* Research on aircraft route planning optimization problem with multi-constraints and dual-targets // J. Math. Industry. 2020. Vol. 10, iss. 26.

2. *Автотор :* [офиц. сайт]. URL: http://www.avtotor.ru/kompaniya/o-kompanii (дата обращения: 01.06.2021).

3. *Jan G.E., Chang K. Yin, Parberry I.* Optimal Path Planning for Mobile Robot Navigation // IEEE/ASME Transactions on Mechatronics. 2008. Vol. 13, iss. 4.

4. *Microsoft* Visual Studio : [офиц. сайт]. URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 01.06.2021).

5. *Элементы* управления для использования в формах Windows Forms. URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/controls-to-use-on-windows-forms?view=netframeworkdesktop-4.8 (дата обращения: 01.06.2021).

6. *Конструкторское* проектирование : методические указания к лабораторному практикуму и курсовому проектированию / сост. В.В. Муханов, А.В. Серегин. Екатеринбург, 2006. URL: https://study.urfu.ru/Aid/Publication/421/1/KnstPrA4.pdf (дата обращения: 01.05.2021).

7. *Лекция 4:* Рекурсивные функции // НОУ ИНТУИТ. URL: https://intuit.ru/studies/courses/648/504/lecture/11423 (дата обращения: 01.06.2021).

Об авторах

Максим Владимирович Головенко — магистрант, Балтийский федеральный университет им. И. Канта, Россия.

E-mail: golovenko.mv@yandex.ru

Олег Владимирович Толстель — канд. техн. наук, доц., Балтийский федеральный университет им. И. Канта, Россия.

E-mail: tolstel.oleg@mail.ru

Дмитрий Викторович Михеенко — начальник отдела промышленной электроники ООО «Автотор Холдинг», Россия.
E-mail: miheenkodv@kld.avtotor.ru

The authors

Maxim V. Golovenko, Master's Student, Immanuel Kant Baltic Federal University, Russia.
E-mail: golovenko.mv@yandex.ru

Dr Oleg V. Tolstel, Associate Professor, Immanuel Kant Baltic Federal University, Russia.
E-mail: tolstel.oleg@mail.ru

Dmitriy V. Mikheenko, Head of the Industrial Electronics Department of Avtotor Holding, Russia.
E-mail: miheenkodv@kld.avtotor.ru