МАТЕМАТИЧЕСКИЕ

МЕТОДЫ

КП.083.

Министерство цифрового развития, связи и массовых коммуникаций РФ

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет

телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

КУРСОВОЙ ПРОЕКТ

по междисциплинарному курсу: «Математические методы»

на тему: «Разработка телеграмм-бота «Финансовый обозреватель»»

Вариант № 5

Выполнил:

Студент группы 083

Новожилов П.А.

Руководитель:

Бурумбаев Д.И.

Екатеринбург 2023

Отзыв руководителя

Министерство цифрового развития, связи и массовых коммуникаций РФ

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет

телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

Цикловая комиссия ИТ и АСУ кафедры ИСТ

ЗАДАНИЕ ДЛЯ КУРСОВОГО ПРОЕКТИРОВАНИЯ

По междисциплинарному курсу Математические методы

Студента 4 курса 083 группы

Новожилова Павла Анатольевича

ТЕМА ЗАДАНИЯ Разработка телеграмм-бота «Финансовый обозреватель»

ИСХОДНЫЕ ДАННЫЕ

ВАРИАНТ 5

Рассмотреть теоретические вопросы, связанные с нахождением кратчайших путей в графе. Составить программу по нахождению кратчайших путей в графе.\_\_\_\_\_\_

При выполнении курсового проекта на указанную тему должны быть представлены:

1. Пояснительная записка

Введение; Задача о кратчайшем пути; Что такое кратчайший путь; Определение кратчайшего пути; Постановки задачи о кратчайшем пути; Задача о кратчайшем пути с учетом дополнительных ограничений; Алгоритмы нахождения кратчайшего пути; Алгоритм Дейкстры; Пример алгоритма Дейкстры; Программная реализация; Разработка блок-схемы; Выбор инструментальных средств разработки; Заключение; Библиография; Приложение А \_\_\_\_\_\_\_\_\_

2. Графическая часть

Дата выдачи Срок окончания

Председатель ЦК ИТиАСУ кафедры ИСТ Тюпина О.М.

Преподаватель – руководитель Бурумбаев Д.И.

Студент Новожилов П.А.

Содержание

Введение 6

1 Инфляция 7

[1.1 Понятие инфляция 7](file:///K:\КУРСАЧ\Отчет%20по%20курсовому%20(СМО-характеристики).docx#_Toc26617250)

[1.2 Рост инфляции 8](file:///K:\КУРСАЧ\Отчет%20по%20курсовому%20(СМО-характеристики).docx#_Toc26617250)

1.3 Контроль инфляции 9

1.4 Проблема неосведомленности 9

2 Программная реализация задачи 11

2.1 Разработка блок-схемы программы 14

2.2 Выбор инструментальных средств разработки 15

2.3 Инструкция по работе с программой 18

Заключение 21

Библиография 22

Приложение A 23

Введение

Данный проект направлен на исследование области инфляционного

роста/падения Рубля, за определенный временной промежуток.

В наше время один из самых развитых троев страны- это капитализм, а в свою очередь самую большую ценность имеют деньги.

В отличие от одного из самых распространенных драгоценных металлов, деньги не имеют ограниченной эмиссии, из-за чего имеет место быть процесс, именуемый - “инфляция”. Инфляция — устойчивое повышение общего уровня цен на товары и услуги; процесс обесценивания денег, падение их покупательной способности вследствие чрезмерного выпуска или сокращения товарной массы в обращении при неизменном количестве выпущенных денег.

Так-как в наше время, не всегда удается уследить за уровнем инфляции, многие люди, даже не представляют, насколько обесценились их накопления всего за 5 лет.

При накоплении определенной суммы, это является большой проблемой, так-как не всегда, ценность накоплений сегодня, будет равна ценности накоплений завтра.

Для решения данной проблемы неосведомленности населения, я напишу телеграмм-бота, который будет наглядно показывать уровень инфляции Российского Рубля.

Целью данного проекта является составление телеграмм-бота, наглядно показывающего уровень обесценивания валюты за определенный временной промежуток .

Для достижения поставленной цели необходимо решить следующие задачи:

—     рассмотреть теоретические материал по теме;

—     разработать блок-схему программы;

—     произвести выбор инструментальных средств разработки;

—     разработать интерфейс программы;

—     написать инструкцию по работе с программой;

—     оформить пояснительную записку.

1 Задача о кратчайшем пути

1 Инфляция

1.1 Понятие инфляции

1.1.1 Что такое инфляция

Инфляция — это устойчивый рост общего уровня цен на товары и услуги. При этом отдельные товары могут заметно дорожать, другие — дешеветь, а третьи — вообще не меняться в цене.

В основном цены на товары и услуги зависят от спроса и предложения на рынке, а некоторые цены регулирует государство. Например, если у фермеров случился хороший урожай овощей, цены на помидоры и картошку будут падать. Если государство в это же время повысило акцизы на алкоголь, цены на спиртное резко пойдут вверх. При этом общий уровень цен может вырасти лишь чуть-чуть.

1.1.2 Единица измерения инфляции

В России инфляцию измеряют так же, как и в большинстве стран мира. Берут так называемую потребительскую корзину — набор продуктов, товаров и услуг, которые регулярно покупает среднестатистический человек или семья. В нее попадает более 500 товаров и услуг — например, продукты, одежда, коммунальные услуги, бытовая техника, автомобили.

При этом надо понимать, что кто-то никогда не ест мясо и не ездит на машине. А кто-то, наоборот, без мяса жить не может и машины меняет каждый год. Потребительская корзина отражает среднее потребление всех жителей страны.

Некоторые из этих товаров люди покупают часто: хлеб, овощи, мясо, бензин. А другие, например, автомобиль, приобретают редко, но зато эти расходы очень большие по сравнению с тратами на те же хлеб, овощи и мясо. Таким образом, при расчете потребительской корзины автомобиль занимает в ней большую долю, чем хлеб.

Стоимость этой условной корзины меняется от месяца к месяцу. Это изменение и есть инфляция.

1.1.3 Виды инфляции

Низкая — до 6% в год. Такая инфляция комфортна и для потребителей, и для предпринимателей. И при этом позволяет экономике развиваться. Именно на таком уровне инфляцию стараются поддерживать в большинстве стран.

Умеренная — от 6 до 10% в год. Она опасна тем, что может выйти из-под контроля и перейти в высокую инфляцию.

Высокая (галопирующая) — от 10 до 100% в год. Она создает нестабильность на рынке, люди и компании не могут планировать свое будущее.

Гиперинфляция — цены растут на сотни и тысячи процентов, в особо тяжелых случаях люди отказываются от денег и переходят на бартер. Обычно гиперинфляция возникает в период тяжелых кризисов и войн.

В постсоветской России рекордно высокая инфляция была зафиксирована в 1992 году — более 2500% в год.

1.2 Рост инфляции

Инфляция может расти по многим причинам:

Увеличение спроса. Случается, что люди вдруг начинают покупать больше определенных товаров. Например, когда мобильные телефоны стали доступными по цене, все захотели пользоваться сотовой связью. Сотовые операторы не успевали сразу подстроиться под высокий спрос: не хватало вышек и свободных частот. Поэтому цены на мобильную связь были очень высокими. Любой дефицит вызывает быстрый рост цен. Но когда у операторов появились дополнительные мощности, цены стали снижаться.

Сокращение предложения. Дефицит может возникнуть и по другой причине — если спрос остается тем же, но товаров и услуг становится меньше. Это может произойти из-за неурожая, ограничения на ввоз иностранных товаров, прекращения или временной приостановки работы компаний-производителей, действий монополиста на каком-то рынке. Это тоже ускоряет рост цен.

Ослабление национальной валюты. Если курс иностранных валют растет, то импортные товары автоматически дорожают. Это тоже приводит к росту инфляции.

Высокие инфляционные ожидания. Когда люди и компании ждут, что цены будут сильно расти, они зачастую начинают менять свое потребительское поведение: покупают товары впрок и меньше сберегают. Ведь логичнее купить сегодня дешевле, чем переплачивать завтра. Компании со своей стороны начинают повышать цены на свою продукцию. Например, производитель сыра ждет, что цены на молоко вырастут. Чтобы покрыть свои будущие расходы на сырье, он заранее начинает поднимать цену на свой товар. Все это подстегивает инфляцию. Получается замкнутый круг: все ждут роста цен и цены растут именно потому, что люди этого ждут.

Высокая инфляция — это всегда плохо. И для экономики, и для бизнеса, и для финансовых рынков, и, конечно, для жителей страны. Люди принимают очевидные финансовые решения: избавляются от денег, тратят их как можно скорее, вкладывая в ценные товары, недвижимость, покупают иностранную валюту. Становится невыгодно делать сбережения, открывать вклады, клиенты банков снимают деньги со счетов.

Обычно высокая инфляция еще и мало предсказуема — разгоняется или движется скачками. Из-за роста нестабильности на финансовых рынках предпринимателям становится невыгодно брать долгосрочные кредиты. Невозможно планировать наперед — а ведь это важнейшее условие для роста инвестиций и экономики в целом.

Особенно сильно инфляция бьет по бедным людям, которые живут на фиксированные доходы. Ведь у более состоятельных людей есть способы полностью или частично защититься от инфляции — например, получая проценты по вкладам или инвестируя на финансовом рынке. Именно поэтому инфляцию иногда называют налогом на бедных.

Но когда цены постоянно идут вниз, ситуация тоже не так хороша, как могло бы показаться.

Такое явление называется дефляцией, или отрицательной инфляцией. Она останавливает развитие экономики. Люди перестают покупать товары в надежде, что они еще сильнее подешевеют. А компании из-за этого сворачивают производство.

1.3 Контроль инфляции

Движение денег в экономике регулируют центральные банки (в нашей стране это Банк России). С помощью инструментов денежно-кредитной политики (ДКП) центробанки могут контролировать инфляцию.

В 2014 году Банк России объявил, что главной целью ДКП в нашей стране стало как раз тартегирование инфляции. То есть регулятор постарается постоянно удерживать рост цен вблизи 4%.

В 2015 году инфляция поднималась до двузначных значений, но регулятору удалось вернуть ее к цели, и в 2017-2020 годах она находилась вблизи тех самых 4%.

В 2021 году рост цен ускорился из-за последствий пандемии коронавируса. С конца февраля 2022 года после введения зарубежных санкций инфляция разогналась с новой силой, так как внешние условия для российской экономики кардинально поменялись.

В этот момент поддержать финансовую стабильность и предотвратить неконтролируемый рост цен помогли экстренные меры Банка России. Например, регулятор резко поднял ключевую ставку – из-за этого выросли проценты по вкладам и кредитам. Люди стали меньше тратить деньги и больше откладывать, спрос на товары сократился, и это способствовало замедлению роста цен.

Санкции нарушили бизнес-процессы многих предприятий. Им потребуется время, чтобы заменить цепочки поставок сырья и комплектующих, наладить новые каналы сбыта. Пока бизнес адаптируется к изменившимся внешним условиям, цены могут быстро расти. Но сейчас регулятор совместно с правительством работает над тем, чтобы период высокой инфляции не затянулся и она постепенно вернулась к целевому значению 4%.

1.4 Проблема неосведомленности

В наше время, мы обрабатываем огромное количество информации из разных источников, из-за этого наш мозг перестает отмечать информацию, про которую мы что-то помним. Российские СМИ (чаще всего негосударственные) часто пишут про уровень инфляции абсолютно разную информацию. Это приводит к тому, что человек неосознанно начинает игнорировать эту информацию, считая ее неинтересной. А это очень важно.

1.4.1 Невозможность планирования

Высокая инфляция мешает планировать расходы: проще сразу все тратить, чем копить. Государство не может обеспечивать малоимущих и пенсионеров достойными социальными выплатами, потому что индексация пособий не успевает за ростом цен. Усиливается социальное расслоение общества.

1.4.2 Торможение экономики

Владельцы предприятий не могут прогнозировать продажи, поэтому объемы производства падают. Растут процентные ставки по кредитам. Торговать становится выгоднее, чем производить: оборот капитала в посредничестве выше, а значит, и прибыль тоже. Экономика не развивается.

1.4.3 Снижение реальных доходов населения

Чем выше темпы инфляции, тем ниже реальная заработная плата, потому что покупательная способность денег снижается.

1.4.4 Падение спроса и инвестиций

Чем выше инфляция, тем меньше товаров и услуг могут купить обычные люди. На некоторые группы товаров, например на предметы роскоши, спрос может упасть очень сильно. Запасы скапливаются на складах, поэтому уменьшаются отгрузки, продажи, а следом и производство. У компаний нет денег, чтобы инвестировать. Деньги остаются только на поддержание текущей деятельности.

Еще большее ускорение инфляции может происходить из-за того, что люди перестают тратить деньги, или, наоборот, из-за ажиотажного спроса на определенные товары. Еще одна возможная причина — рост денежной массы.

1.4.5 Повышение стоимости кредитов и доходности вкладов

Во время инфляции долги возвращаются подешевевшими деньгами, поэтому стоимость кредитов растет. Банкам сложно хеджировать кредитные риски, потому что неизвестно, какая будет инфляция, — это тоже повышает стоимость кредитных денег. Одновременно с этим доходность вкладов тоже растет: проценты поднимают, чтобы привлечь людей.

1.4.6 Безработица

На коротких периодах инфляция и безработица двигаются в разные стороны, то есть чем выше инфляция, тем ниже безработица и наоборот. Эту зависимость в 1958 году выявил новозеландский экономист Олбан Филлипс. Но позже другие экономисты пришли к выводу, что в долгосрочной перспективе инфляция и безработица не связаны между собой.

1.4.7 Ослабление или укрепление валюты

Если курс национальной валюты падает, импортные товары становятся дороже — это увеличивает инфляцию. Если импорт зарубежных товаров значительный, то их резкое подорожание может привести к ценовому шоку.

2 Разработка программы

# 2.1 Разработка блок-схемы программы



Рисунок 9 – Блок схема программы

# 

2.2 Выбор инструментальных средств разработки

При выборе инструментальной среды для проектирования автоматизированной информационной системы среди наиболее популярных средств разработки стоит выделить следующие:

* Delphi;
* C++ Builder;
* Visual Studio.

Рассмотрим возможности каждой из средств разработки систем поподробнее.

C++ Builder - это одна из самых современных и эффективных сред программирования. Builder - это среда, в которой можно осуществлять так называемое визуальное программирование, то есть создавать программы, позволяющие в момент их исполнения взаимодействовать с пользователем с помощью многооконного графического интерфейса.

В момент выполнения программы на экране могут появляться в цветном изображении элементы управления программой: кнопки, на которые можно нажимать мышью, после чего происходят некоторые "привязанные" к этим кнопкам действия; поля для ввода/вывода данных; списки, из которых можно выбирать данные и применять их для дальнейших расчетов в программе; различные меню, позволяющие выбирать и выполнять те или иные действия; элементы, контролирующие состояния каких-то объектов; элементы, помогающие следить за ходом некоторых процессов во время их выполнения; элементы, позволяющие выбирать дату из календаря; элементы, обеспечивающие стандартный выбор файлов, шрифтов, цвета, настройки принтеров и другие; элементы, позволяющие вставлять в разрабатываемую программу ссылки на другие программы-объекты.

Кроме этого, для лучшего понимания ходы выполнения программы на экране могут прорисовываться различные изображения, создаваться иерархические деревья, с помощью которых можно управлять ходом выполнения программы. Среда Builder позволяет работать как с простыми локальными и удаленными базами данных, так и с многозвенными распределенными базами данных, а также позволяет построить общение вашей программы с Интернетом.

Среда Builder - инструмент разработки программ на основе современного метода создания программа - объектно-ориентированного программирования (ООП).

Delphi - это продукт Borland International для быстрого создания приложений. Процесс создания интерфейса будущей программы напоминает забаву с игровым компьютерным конструктором. Поэтому RAD-среды еще называют визуальными средами разработки: какими мы видим рабочие и диалоговые окна программы при проектировании, такими они и будут, когда программа заработает.

Высокопроизводительный инструмент визуального построения приложений включает в себя настоящий компилятор кода и предоставляет средства визуального программирования, несколько похожие на те, что можно обнаружить в Microsoft Visual Basic или в других инструментах визуального проектирования. В основе Delphi лежит язык Object Pascal, который является расширением объектно-ориентированного языка Pascal. В Delphi также входят локальный SQL-сервер, генераторы отчетов, библиотеки визуальных компонентов, и прочее, необходимое для того, чтобы чувствовать себя совершенно уверенным при профессиональной разработке информационных систем или просто программ для Windows-среды.

Прежде всего Delphi предназначен для профессиональных разработчиков, желающих очень быстро разрабатывать приложения в архитектуре клиент-сервер. Delphi производит небольшие по размерам высокоэффективные исполняемые модули (. exe и. dll), поэтому в Delphi должны быть, прежде всего, заинтересованы те, кто разрабатывает продукты на продажу. С другой стороны небольшие по размерам и быстро исполняемые модули означают, что требования к клиентским рабочим местам существенно снижаются - это имеет немаловажное значение и для конечных пользователей.

Преимущества Delphi по сравнению с аналогичными программными продуктами. быстрота разработки приложения (RAD); высокая производительность разработанного приложения; низкие требования разработанного приложения к ресурсам компьютера; наращиваемость за счет встраивания новых компонент и инструментов в среду Delphi; возможность разработки новых компонентов и инструментов собственными средствами Delphi (существующие компоненты и инструменты доступны в исходных кодах); удачная проработка иерархии объектов.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же работодателей интересует, прежде всего, скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Delphi полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

Microsoft Visual Studio - линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE,.net Framework, Xbox, Windows Phone.net Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения.

Из перечня рассматриваемых сред программирования была выбрана среда Visual Studio на языке C++, как наиболее фундаментальный язык программирования среда в сфере обучения, позволяющий познать все основные моменты в проектировании

2.3 Инструкция по работе с программой

Запустите программу, нажав два раза ЛКМ по файлу программы.

Откроется главное окно программы (Рисунок 10).

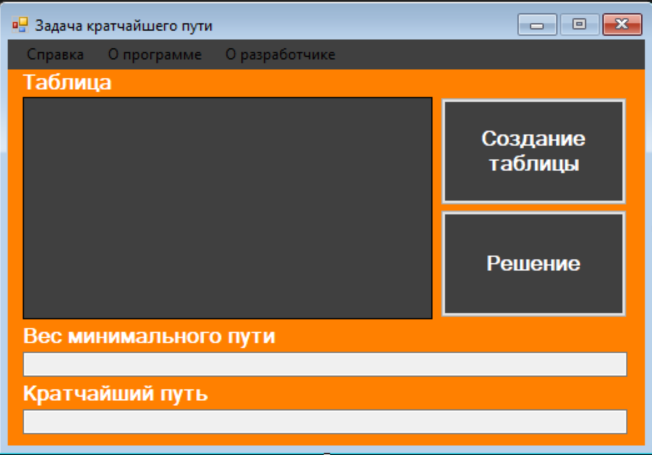


Рисунок 10 – Основное окно программы.

В верхней строчке серого цвета, вы увидите три поля: «Справка», «О программе» и «О разработчике».

Нажав ЛКМ на поле «Справка» у вас откроется окно, в котором будет написано краткое руководство пользования программой (Рисунок 11).

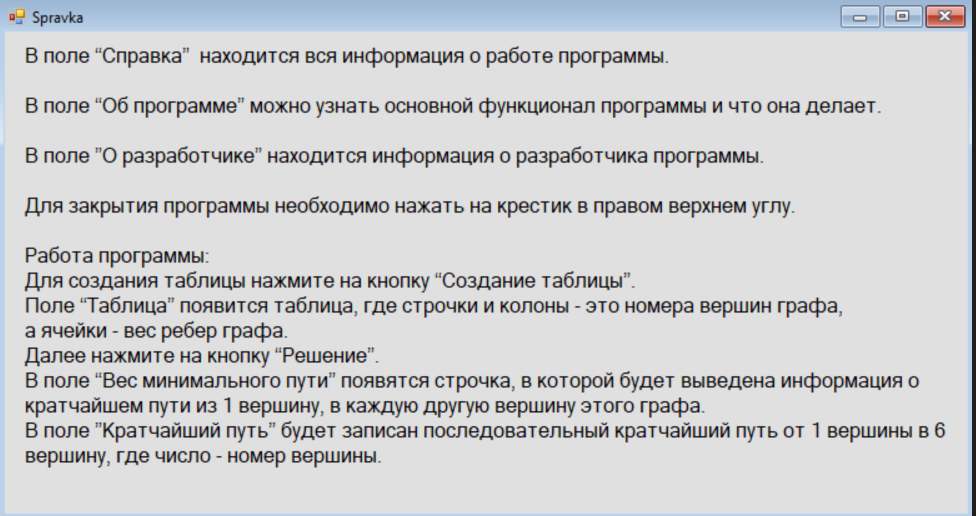


Рисунок 11 – Окно «Справка».

Нажав ЛКМ на поле «О программе» у вас откроется окно, в котором будет написана информация о задаче программы (Рисунок 12).

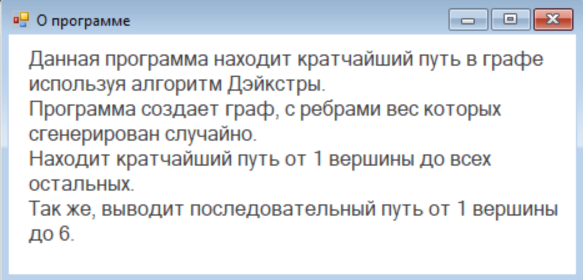


Рисунок 12 – Окно «О программе».

Нажав ЛКМ на поле «О разработчике» у вас откроется вкладка, в которой будет написана информация о разработчике программы (рисунок 13).

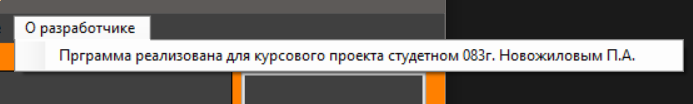


Рисунок 13 – Вкладка «О разработчике».

Работа программы:

Для создания таблицы нажмите на кнопку “Создание таблицы”.

В поле «Таблица» появится таблица, где строчки и колоны - это номера вершин графа, а ячейки - вес ребер графа.

Далее нажмите на кнопку «Решение».

В поле «Вес минимального пути» появятся строчка, в которой будет выведена информация о кратчайшем пути из 1 вершину, в каждую другую вершину этого графа.

В поле «Кратчайший путь» будет записан последовательный кратчайший путь от 1 вершины в 6 вершину, где число - номер вершины.

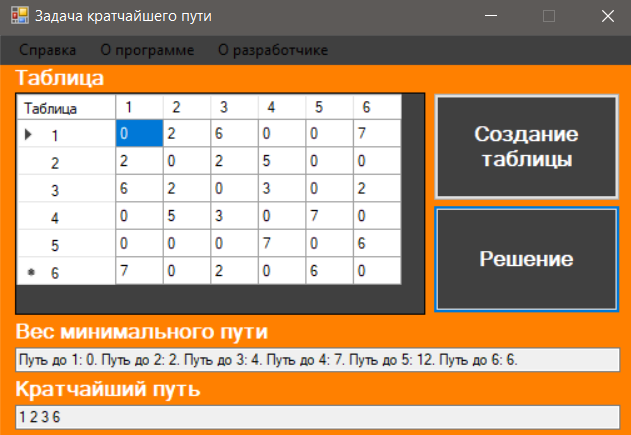


Рисунок 14 – Главное окно программы с решением графа.

# Заключение

# В заключении курсового проекта сделаем вывод. Задача кратчайшего пути в наше время часто используется в больших сетях транспорта и является очень важным этапом планирования в крупных компаниях. Так же ее используют: в навигаторах, в недетерминированных машинах и во многих других отраслях.

# Узнав это, я понял насколько полезно уметь писать программу для решения такого образа задач. Также я познакомился с новыми методами решения задач кратчайшего пути, что помогло мне найти оптимальный алгоритм решения, а именно алгоритм Дейкстры. Граф является математической моделью самых разнообразных объектов, явлений и процессов, которые исследуются и используются в науке, технике и на практике. Графы позволяют строить математическую модель связей между заданными элементами. Например, в виде графа могут быть изображены электрические, транспортные, информационные и компьютерные и другие сети, карты автомобильных, железнодорожных, воздушных путей, лабиринты, модели кристаллов, структуры молекул химических веществ, и так далее.

Примерами применения теории графов является поиск связных компонентов и поиск кратчайших, самых „дешевых” и самых „дорогих” путей в коммуникационных сетях. Для построения таких путей используются разнообразные алгоритмы на графах. Следовательно, практическая ценность теории графов бесспорна.Библиография

# 1 Учебное пособие: Алгоритм Дейкстры нахождения кратчайшего пути [Электронный ресурс] – Режим доступа: https://prog-cpp.ru/deikstra;

# 2 Учебно-методическое пособие: Пути на графе. Задача коммивояжёра на C++ [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/650643;

# 3 Учебно-методическое пособие: Нахождение кратчайших путей от заданной вершины до всех остальных вершин алгоритмом Дейкстры для разреженных графов [Электронный ресурс] – Режим доступа: https://e-maxx.ru/algo/dijkstra\_sparse;

# 4 Учебно-методическое пособие: Кратчайшие пути в графах. BFS. Dijkstra. [Электронный ресурс] – Режим доступа: https://algocode.ru/page/c-12-bfs.

# Приложение А

(справочное)

Листинг программы

Ниже представлен листинг программы:

Myform.cpp

#include "MyForm.h"

#include <time.h>

using namespace System; // формальности для работы интерфэйса

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(array<String^>^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

ZKP::MyForm form;

Application::Run(% form);

}

//данные о матрице

float\*\* A;

int N\_A = 6, M\_A = 6;

// начинка интерфэйса

System::Void ZKP::MyForm::выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

return System::Void(); // начинка справки

}

System::Void ZKP::MyForm::label1\_Click(System::Object^ sender, System::EventArgs^ e)

{

return System::Void(); // начинка подписи таблицы

}

System::Void ZKP::MyForm::Obprogramme\_Click(System::Object^ sender, System::EventArgs^ e)

{

return System::Void(); // начинка кнопочки об программе

}

System::Void ZKP::MyForm::Sozdanietablici\_Click(System::Object^ sender, System::EventArgs^ e)

{

// Размерность матрицы

// Создание матрицы

A = new float\* [SIZE];

for (int i = 0; i < SIZE; i++)

A[i] = new float[SIZE];

// Заполнение значений матрицы

for (int i = 0; i < SIZE; i++)

{

matrix[i][i] = 0;

for (int j = i + 1; j < SIZE; j++) {

matrix[i][j] = rand() % 10 + 1;

matrix[0][3] = 0;

matrix[0][4] = 0;

matrix[1][4] = 0;

matrix[1][5] = 0;

matrix[2][4] = 0;

matrix[3][5] = 0;

matrix[5][5] = 0;

matrix[j][i] = matrix[i][j];

}

}

//создание таблицы

Tablica->RowCount = N\_A;

Tablica->ColumnCount = M\_A;

// вывод таблицы

Show();

//Авто выравнивание

Tablica->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

Tablica->AutoResizeColumns();

return System::Void(); // начинка кнопуи создание таблицы

}

System::Void ZKP::MyForm::Reshenietablici\_Click(System::Object^ sender, System::EventArgs^ e)

{

for (int i = 0; i < SIZE; i++)

{

d[i] = 10000;

a[i] = 1;//отмечает все вершины как необработанные.

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;//индекс вершины с минимальным весом

min = 10000;// мин вес

for (int i = 0; i < SIZE; i++)

{ // Если вершину ещё не обошли и вес меньше min

if ((a[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

for (int i = 0; i < SIZE; i++)

{

if (matrix[minindex][i] > 0)

{

save = min + matrix[minindex][i];

if (save < d[i])

{

d[i] = save;

}

}

}

a[minindex] = 0;

}

} while (minindex < 10000);

int ver[SIZE]; // массив посещенных вершин

int end = 5; // индекс конечной вершины = 6 - 1

ver[0] = end + 1; // начальный элемент - конечная вершина

int k = 1; // индекс предыдущей вершины

int weight = d[end]; // вес конечной вершины

while (end != begin\_index) // пока не дошли до начальной вершины

{

for (int i = 0; i < SIZE; i++) // просматриваем все вершины

if (matrix[end][i] != 0) // если связь есть

{

int temp = weight - matrix[end][i]; // определяем вес пути из предыдущей вершины

if (temp == d[i]) // если вес совпал с рассчитанным

{ // значит из этой вершины и был переход

weight = temp; // сохраняем новый вес

end = i; // сохраняем предыдущую вершину

ver[k] = i + 1; // и записываем ее в массив

k++;

}

}

}

// Вывод кратчайших расстояний до вершин

//for (int i = 0; i < SIZE; i++) {}

// Вывод пути (начальная вершина оказалась в конце массива из k элементов)

String^ str = "";

String^ ves = "";

for (int ip = 0; ip < SIZE; ip++) {

kek = ip + 1;

ves += ("Путь до " + kek + ": " + d[ip] + ". ");

}

for (int i = k - 1; i >= 0; i--)

str += (ver[i] + " ");

Oknootveta-> Text = str;

Okno\_vesa->Text = ves;

}

System::Void ZKP::MyForm::Razmernost\_ValueChanged(System::Object^ sender, System::EventArgs^ e)

{

return System::Void(); //начинка размерности таблицы

}

System::Void ZKP::MyForm::Tablica\_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e)

{

return System::Void(); // начинка таблицы для матрицы

}

System::Void ZKP::MyForm::Oknootveta\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

return System::Void(); // Начинка окна вывода решений

}

void ZKP::MyForm::Show() // метод создания таблицы для матрицы

{

// вывод матрицы в таблицу

for (int i = 0; i < N\_A; i++)

{

for (int j = 0; j < M\_A; j++)

{

// название таблицы

Tablica->TopLeftHeaderCell->Value = "Таблица";

// вывод номеров стоблцов

Tablica->Columns[j]->HeaderCell->Value = Convert::ToString(j + 1);

// вывод номеров строк

Tablica->Rows[i]->HeaderCell->Value = Convert::ToString(i + 1);

// Вывод значений в таблицу

Tablica->Rows[i]->Cells[j]->Value = matrix[i][j];

}

}

}

void ZKP::MyForm::alg\_kratchayshego\_pyti()

{

throw gcnew System::NotImplementedException();

}

void ZKP::MyForm::v\_p()

{

throw gcnew System::NotImplementedException();

}

System::Void ZKP::MyForm::Okno\_vesa\_TextChanged(System::Object^ sender, System::EventArgs^ e)

{

return System::Void();

}

MyForm.f

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#define SIZE 6// количество вершин в нашем графе

int matrix[SIZE][SIZE] = { 0 }; // матрица связей

int a[SIZE]; // посещенные вершины

int d[SIZE]; // минимальное расстояние

int save;//временное хранение различных величин

int begin\_index = 0;//задаем индекс начальной вершины, от которой будем искать путь

int min;

int minindex;

int kek;

namespace ZKP {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

protected:

private: System::Windows::Forms::Label^ Podpistablici;

private: System::Windows::Forms::DataGridView^ Tablica;

private: System::Windows::Forms::Button^ Sozdanietablici;

private: System::Windows::Forms::Button^ Reshenietablici;

private: System::Windows::Forms::TextBox^ Oknootveta;

private: System::Windows::Forms::Label^ Podpisotveta;

private: System::Windows::Forms::ToolStripMenuItem^ Spravka;

private: System::Windows::Forms::ToolStripMenuItem^ Obprogramme;

private: System::Windows::Forms::ToolStripMenuItem^ праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::TextBox^ Okno\_vesa;

private: System::Windows::Forms::ToolStripMenuItem^ оРазработчикеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->Podpistablici = (gcnew System::Windows::Forms::Label());

this->Tablica = (gcnew System::Windows::Forms::DataGridView());

this->Sozdanietablici = (gcnew System::Windows::Forms::Button());

this->Reshenietablici = (gcnew System::Windows::Forms::Button());

this->Oknootveta = (gcnew System::Windows::Forms::TextBox());

this->Podpisotveta = (gcnew System::Windows::Forms::Label());

this->Spravka = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->Obprogramme = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->оРазработчикеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->label1 = (gcnew System::Windows::Forms::Label());

this->Okno\_vesa = (gcnew System::Windows::Forms::TextBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Tablica))->BeginInit();

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// Podpistablici

//

this->Podpistablici->AutoSize = true;

this->Podpistablici->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->Podpistablici->ForeColor = System::Drawing::Color::White;

this->Podpistablici->Location = System::Drawing::Point(8, 24);

this->Podpistablici->Name = L"Podpistablici";

this->Podpistablici->Size = System::Drawing::Size(80, 20);

this->Podpistablici->TabIndex = 1;

this->Podpistablici->Text = L"Таблица";

this->Podpistablici->Click += gcnew System::EventHandler(this, &MyForm::label1\_Click);

//

// Tablica

//

this->Tablica->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->Tablica->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->Tablica->Location = System::Drawing::Point(12, 46);

this->Tablica->Name = L"Tablica";

this->Tablica->Size = System::Drawing::Size(328, 178);

this->Tablica->TabIndex = 2;

this->Tablica->CellContentClick += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &MyForm::Tablica\_CellContentClick);

//

// Sozdanietablici

//

this->Sozdanietablici->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->Sozdanietablici->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Sozdanietablici->ForeColor = System::Drawing::Color::Transparent;

this->Sozdanietablici->Location = System::Drawing::Point(346, 46);

this->Sozdanietablici->Name = L"Sozdanietablici";

this->Sozdanietablici->Size = System::Drawing::Size(150, 87);

this->Sozdanietablici->TabIndex = 4;

this->Sozdanietablici->Text = L"Создание таблицы";

this->Sozdanietablici->UseVisualStyleBackColor = false;

this->Sozdanietablici->Click += gcnew System::EventHandler(this, &MyForm::Sozdanietablici\_Click);

//

// Reshenietablici

//

this->Reshenietablici->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->Reshenietablici->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Reshenietablici->ForeColor = System::Drawing::Color::White;

this->Reshenietablici->Location = System::Drawing::Point(346, 136);

this->Reshenietablici->Name = L"Reshenietablici";

this->Reshenietablici->Size = System::Drawing::Size(150, 87);

this->Reshenietablici->TabIndex = 5;

this->Reshenietablici->Text = L"Решение";

this->Reshenietablici->UseVisualStyleBackColor = false;

this->Reshenietablici->Click += gcnew System::EventHandler(this, &MyForm::Reshenietablici\_Click);

//

// Oknootveta

//

this->Oknootveta->Location = System::Drawing::Point(12, 296);

this->Oknootveta->Name = L"Oknootveta";

this->Oknootveta->ReadOnly = true;

this->Oknootveta->Size = System::Drawing::Size(484, 20);

this->Oknootveta->TabIndex = 7;

this->Oknootveta->TextChanged += gcnew System::EventHandler(this, &MyForm::Oknootveta\_TextChanged);

//

// Podpisotveta

//

this->Podpisotveta->AutoSize = true;

this->Podpisotveta->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->Podpisotveta->ForeColor = System::Drawing::Color::White;

this->Podpisotveta->Location = System::Drawing::Point(8, 273);

this->Podpisotveta->Name = L"Podpisotveta";

this->Podpisotveta->Size = System::Drawing::Size(157, 20);

this->Podpisotveta->TabIndex = 8;

this->Podpisotveta->Text = L"Кратчайший путь";

this->Podpisotveta->Click += gcnew System::EventHandler(this, &MyForm::Podpisotveta\_Click);

//

// Spravka

//

this->Spravka->Name = L"Spravka";

this->Spravka->Size = System::Drawing::Size(65, 20);

this->Spravka->Text = L"Справка";

this->Spravka->Click += gcnew System::EventHandler(this, &MyForm::выходToolStripMenuItem\_Click);

//

// Obprogramme

//

this->Obprogramme->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem });

this->Obprogramme->Name = L"Obprogramme";

this->Obprogramme->Size = System::Drawing::Size(94, 20);

this->Obprogramme->Text = L"О программе";

this->Obprogramme->Click += gcnew System::EventHandler(this, &MyForm::Obprogramme\_Click);

//

// праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem

//

this->праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem->Name = L"праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem";

this->праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem->Size = System::Drawing::Size(704, 22);

this->праграммаРешенияЗадачКратчайшегоПутиV10ToolStripMenuItem->Text = L"Данная программа находит кратчайший путь вааааааааааааааааааааааааааааааааааааааа"

L"аааааааааааааааааааааа ";

//

// menuStrip1

//

this->menuStrip1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->menuStrip1->ImageScalingSize = System::Drawing::Size(20, 20);

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->Spravka, this->Obprogramme,

this->оРазработчикеToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(510, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"menuStrip1";

//

// оРазработчикеToolStripMenuItem

//

this->оРазработчикеToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem });

this->оРазработчикеToolStripMenuItem->Name = L"оРазработчикеToolStripMenuItem";

this->оРазработчикеToolStripMenuItem->Size = System::Drawing::Size(110, 20);

this->оРазработчикеToolStripMenuItem->Text = L"О разработчике ";

this->оРазработчикеToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::оРазработчикеToolStripMenuItem\_Click);

//

// прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem

//

this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem->Name = L"прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem"

L"";

this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem->Size = System::Drawing::Size(535, 22);

this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem->Text = L"Прграмма реализована для курсового проекта студетном 083г. Новожиловым П.А.";

this->прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label1->ForeColor = System::Drawing::Color::White;

this->label1->Location = System::Drawing::Point(8, 227);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(216, 20);

this->label1->TabIndex = 9;

this->label1->Text = L"Вес минимального пути ";

this->label1->Click += gcnew System::EventHandler(this, &MyForm::label1\_Click\_1);

//

// Okno\_vesa

//

this->Okno\_vesa->Location = System::Drawing::Point(12, 250);

this->Okno\_vesa->Name = L"Okno\_vesa";

this->Okno\_vesa->ReadOnly = true;

this->Okno\_vesa->Size = System::Drawing::Size(484, 20);

this->Okno\_vesa->TabIndex = 10;

this->Okno\_vesa->TextChanged += gcnew System::EventHandler(this, &MyForm::Okno\_vesa\_TextChanged);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(0)));

this->ClientSize = System::Drawing::Size(510, 325);

this->Controls->Add(this->Okno\_vesa);

this->Controls->Add(this->label1);

this->Controls->Add(this->Podpisotveta);

this->Controls->Add(this->Oknootveta);

this->Controls->Add(this->Reshenietablici);

this->Controls->Add(this->Sozdanietablici);

this->Controls->Add(this->Tablica);

this->Controls->Add(this->Podpistablici);

this->Controls->Add(this->menuStrip1);

this->ForeColor = System::Drawing::Color::Black;

this->MainMenuStrip = this->menuStrip1;

this->MaximizeBox = false;

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Задача кратчайшего пути";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Tablica))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e); // обьявление справки

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e); // обьявление подписи таблицы

private: System::Void Obprogramme\_Click(System::Object^ sender, System::EventArgs^ e); // обьявление информации ОБ программе

private: System::Void Sozdanietablici\_Click(System::Object^ sender, System::EventArgs^ e); // обьявление

private: System::Void Reshenietablici\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void Razmernost\_ValueChanged(System::Object^ sender, System::EventArgs^ e);

private: System::Void Tablica\_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e);

private: System::Void Oknootveta\_TextChanged(System::Object^ sender, System::EventArgs^ e);

private: void Show();

void alg\_kratchayshego\_pyti();

void v\_p();

private: System::Void Podpisotveta\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label1\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void Okno\_vesa\_TextChanged(System::Object^ sender, System::EventArgs^ e);

private: System::Void прграммаРеализованаДляКурсовогоПроектаСтудетном083гНовожиловымПАToolStripMenuItem\_Click(System::Object^ sender,

System::EventArgs^ e) {

}

private: System::Void оРазработчикеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

} // код сгенерированый канструктором

Spravka.h

#pragma once

namespace ZKP {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для Spravka

/// </summary>

public ref class Spravka : public System::Windows::Forms::Form

{

public:

Spravka(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~Spravka()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(Spravka::typeid));

this->label1 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// label1

//

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(12, 9);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(747, 371);

this->label1->TabIndex = 0;

this->label1->Text = resources->GetString(L"label1.Text");

//

// Spravka

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(224)), static\_cast<System::Int32>(static\_cast<System::Byte>(224)),

static\_cast<System::Int32>(static\_cast<System::Byte>(224)));

this->ClientSize = System::Drawing::Size(771, 386);

this->Controls->Add(this->label1);

this->Name = L"Spravka";

this->Text = L"Spravka";

this->ResumeLayout(false);

}

#pragma endregion

};

}

MyForm1.h

#pragma once

namespace ZKP {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm1

/// </summary>

public ref class MyForm1 : public System::Windows::Forms::Form

{

public:

MyForm1(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm1::typeid));

this->label1 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// label1

//

this->label1->BackColor = System::Drawing::Color::White;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->label1->Location = System::Drawing::Point(12, 9);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(435, 171);

this->label1->TabIndex = 0;

this->label1->Text = resources->GetString(L"label1.Text");

this->label1->Click += gcnew System::EventHandler(this, &MyForm1::label1\_Click);

//

// MyForm1

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::White;

this->ClientSize = System::Drawing::Size(454, 192);

this->Controls->Add(this->label1);

this->Name = L"MyForm1";

this->Text = L"О программе ";

this->Load += gcnew System::EventHandler(this, &MyForm1::MyForm1\_Load);

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void MyForm1\_Load(System::Object^ sender, System::EventArgs^ e) {

}

};

}

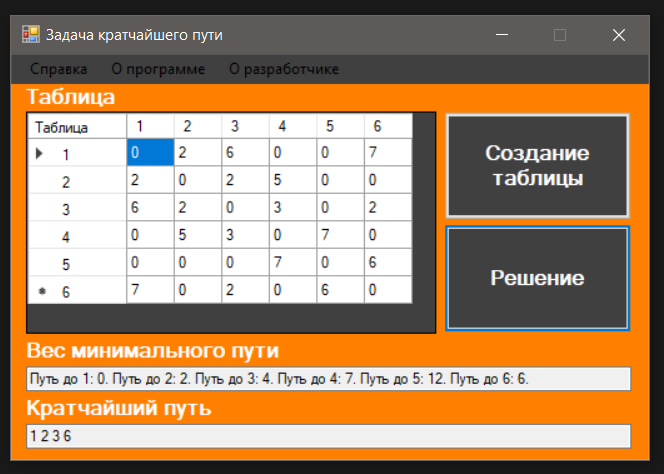


Рисунок 15 – Основное окно программы.