МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра информатики и автоматизации научных исследований**

Направление подготовки: «Прикладная информатика»

Профиль подготовки: «Прикладной бакалавриат»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**Тема:**

**«Построение расписания канонической задачи сетевого планирования методом многопроходного фронтального алгоритма»**

Допущена к защите Выполнил:

Заведующий кафедрой: студент группы 381507-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Суворов Кирилл Александрович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_подпись подпись

Научный руководитель:

Доцент

Филимонов Андрей Викторович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Рецензент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород  
2019

Оглавление

[**Введение** 2](#_Toc7909739)

[**1. Формальная постановка задачи** 3](#_Toc7909740)

[1.1 Исходные параметры. 3](#_Toc7909741)

[1.2 Математическая модель 4](#_Toc7909742)

[**1.3 Критерий задачи:** 4](#_Toc7909743)

[**1.4 Комбинаторная и математическая сложность** 5](#_Toc7909744)

[**1.5 Индивидуальные постановки задач** 5](#_Toc7909745)

[**2. Алгоритм решения задачи** 6](#_Toc7909746)

[2.1 Общая идея алгоритма 6](#_Toc7909747)

[2.2 Перестановочные процедуры преобразования фронта работ в вектор. 7](#_Toc7909748)

[2.3 Процедуры «построители» 7](#_Toc7909749)

[**3. Программная реализация** 7](#_Toc7909750)

[3.1 Функциональные возможности программы 7](#_Toc7909751)

[**3.2 Основные формы, классы и функции программы** 8](#_Toc7909752)

[3.2.1 Общая информация о программе 8](#_Toc7909753)

[3.2.2 Формы программы 8](#_Toc7909754)

[3.2.3 Структура данных 8](#_Toc7909755)

[3.2.4 Основные функции 8](#_Toc7909756)

[3.2.5 Хранение входной информации 9](#_Toc7909757)

[3.3 Эксперимент 9](#_Toc7909758)

[Заключение 11](#_Toc7909759)

[**Список литературы** 12](#_Toc7909760)

[Приложение 12](#_Toc7909761)

# **Введение**

Время - это самый ценный ресурс в современном мире, потому что его нельзя восполнить или вернуть. Его значение сложно переоценить, ведь даже небольшое опоздание может повлечь за собой материальные потери и снижение репутации. На сегодняшний день, проблема уменьшения временных задержек является одной из основных задач как в жизни отдельно взятого человека, так и в работе огромных международных корпораций. Именно поэтому необходимо правильно управлять не только людьми или машинами, но и временем. Человечество придумало множество инструментов для измерения, анализа и организации времени.

Одним из способов регулирования производства является составление расписания и графиков занятости сотрудников. Этот метод позволяет наглядно отобразить все этапы производства, их порядок и время, затраченное на выполнение. Кроме того, полезным побочным результатом выполнения является однозначное отношение необходимых ресурсов, в том числе трудовых, к конкретному процессу.

Для примера, рассмотрим компанию ООО «Праздник», организующую мероприятия под заказ. В этой организации, каждый сотрудник самостоятельно и последовательно выполняет поставленные перед ним задачи. При этом каждый из них имеет различный уровень знаний и умений, поэтому время выполнения одной и той же задачи разными сотрудниками может отличаться. Также известны все мероприятии, планируемые на расчётный период. В каждом — определен список подзадач и порядок их выполнения, то есть нельзя приступить к работе над очередным этапом, пока не выполнены все предшествующие. Кроме того, существует срок, раньше которого невозможно приступить к выполнению той или иной задачи из-за внешних ограничений, например, по причине погодных условий или доступности ресурсов. С целью определения эффективности работы команды для всех задач установлены предельные сроки, к которым они должны быть выполнены, и штрафы за несвоевременное исполнение. Исходя из данных условий необходимо построить расписание задач для сотрудников с учетом текущей политики компании. В данной работе рассматриваются следующие приоритетные цели руководителя:

* уменьшение суммарного штрафа директивных сроков;
* равномерная нагрузка всех сотрудников.

К сожалению, на данный момент не существует оптимальных алгоритмов для составления расписания, а сложность задач постоянно растет, что повышает актуальность исследований в данном направлении.

Основываясь на выявленных проблемах, была сформулирована главная цель работы – разработать эффективный алгоритм построения расписания для компаний, организационная модель которых аналогична ООО «Праздник»

Для достижения поставленной цели были выполнены следующие этапы работы:

1. Изучение методов построения расписания;
2. Рассмотрение вариаций фронтального алгоритма;
3. Построение математической модели;
4. Разработка алгоритма;
5. Программная реализация алгоритма;
6. Тестирование;
7. Анализ эффективности работы алгоритма.

# **1. Формальная постановка задачи**

Рассмотрим план работы коллектива, в котором каждый сотрудник выполняет поставленные перед ним задачи. Кроме того, известны все проекты, планируемые на расчётный период, и детализированы подзадачи каждого. Все полученные работы взаимозависимы, причем ни одна работа не может быть выполнена пока не закончены все предшествующие ей этапы. Таким образом подготовку всех проектов можно описать каноническими сетевыми моделями, которые представляют собой взвешенный ориентированный граф без петель и контуров. Вершинами этого графа являются задачи, которым в соответствие ставят определенный набор характеристик: для всех работ известно время выполнения каждым сотрудником и время максимально раннего начала выполнения (по умолчанию 0 – начало проектируемого отрезка времени), а для некоторых, может указываться крайнее время окончания выполнения. В случае задержки какой-либо работы коллектив несет потери в баллах к премии, в зависимости от важности конкретного проекта.

## 1.1 Исходные параметры.

К исходным параметрам относится:

## 1.2 Математическая модель

**Варьируемые параметры:**

**Ограничения математической модели:**

Естественные ограничения на варьируемые параметры:

Ограничение на время выполнения работ, исходя из канонической сетевой модели:

Ограничение на время выполнения исполнителем работ без перерывов:

Ограничение, связанное с выполнением исполнителем не более одной работы в каждый момент времени:

Ограничение, связанное с выполнением работ не более чем одним исполнителем:

Ограничение на начало выполнения работы:

## **1.3 Критерий задачи:**

## **1.4 Комбинаторная и математическая сложность**

Поставленная задача относится к классу NP-трудных задач, т.е. для ее решения не существует эффективных алгоритмов определения оптимального решения. В случае полного перебора каждой работе нужно выбрать исполнителя. Поэтому число всевозможных вариантов будет равно размещению с повторением из m исполнителей по n работ ().

## **1.5 Индивидуальные постановки задач**

T={1, …, 20}

P={1,2}

J(1) = {1,2,3,8)

J(2)={1,4,5,6,7,8}

J={1, …, 8}

I={1,2}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j | A(j) |  |  |  |
| 1 | Ø | 2 | 1 |  |
| 2 | {1} | 2 | 1 |  |
| 3 | {2} | 2 | 1 | 8 |
| 4 | {1} | 2 | 1 |  |
| 5 | {4} | 2 | 8 |  |
| 6 | {5} | 2 | 1 |  |
| 7 | {4} | 2 | 12 |  |
| 8 | {3,6,7} | 2 | 1 | 20 |

C=

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 3 | 3 |
| 2 | 2 | 3 |
| 3 | 3 | 2 |
| 4 | 4 | 3 |
| 5 | 4 | 2 |
| 6 | 1 | 3 |
| 7 | 5 | 6 |
| 8 | 3 | 3 |

X=(0,3,5,3,8,10,12,17)

Y=(3,5,8,6,10,11,17,20)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# **2. Алгоритм решения задачи**

## 2.1 Общая идея алгоритма

Для решения поставленной задачи был выбран фронтальный подход, являющийся представителем идеологии «жадных» алгоритмов. Под жадными алгоритмами будем понимать алгоритмы, в которых включенная в строящееся расписание работа не может быть исключена из него на последующих шагах построения. Жадные алгоритмы применительно к рассматриваемой задаче являются эффективными алгоритмами (по вычислительной сложности), которые не гарантируют получения точного решения, но находят допустимое решения, являющиеся приближенным к оптимальному.

Выбранный подход решения основывается на построении в каждый такт времени «фронта работ». Пусть t – произвольный такт планирования, . Назовем «фронтом работ» множество F(t) – множество операций, любая из которых может выполняться, начиная с такта t, . Для выбора очередной операции из фронта работ перейдем от множества F(t) к вектору , компоненты которого и будут определять порядок выполнения операций. После установления порядка выполнения выбранная из фронта работ очередная работа включается в строящееся расписание с помощью определенной процедуры «построитель», причем для ее выполнения используются все доступные к данному моменту времени ресурсы в максимально возможном объеме. Для установления порядка могут быть применены специальные перестановочные процедуры.

Таким образом схема фронтального алгоритма состоит из следующих шагов:

* в заданный такт времени t задается фронт работ - множество операций, любая из которых может начать выполняться (нет невыполненных предшествующих работ и самое раннее время возможного старта «пройдено» )
* фронт работ преобразуется в вектор, используя перестановочные процедуры;
* операции в установленном порядке из вектора работ включаются в строящееся расписание, с помощью определенной процедуры «построитель»
* переход к следующему такту планирования

## 2.2 Перестановочные процедуры преобразования фронта работ в вектор.

Для формирования вектора операций из фронта работ используем стратегию сортировки по убыванию штрафа на единицу времени выполнения. Рассмотрим алгоритм выбранного метода:

1. Сформируем V(t) - множество исполнителей, свободных в рассматриваемый момент времени t, а так же создадим копию фронта работ F`(t)=F(t) .
2. Если свободных исполнителей нет, то есть , переходим на шаг 6.
3. Найдем - минимальное значение временных затрат, необходимых на выполнения произвольной работы из F`(t) любым исполнителем из V(t).
4. Добавим операцию с номером h в формируемый вектор работ, и исключим ее из используемого множества работ F`(t), а также исключим исполнителя с номером g из множества свободных исполнителей V(t).
5. Если еще остались доступные работы и исполнители, т.е. , то вернемся к шагу 3.
6. Если еще остались работы, готовые к исполнению (), добавляем их по очереди к строящемуся вектору. (Эти работы не начнут свое выполнение в рассматриваемый момент времени t, так как нет свободных исполнителей (), поэтому их порядок не важен. Однако, добавим их для сохранения размерности вектора)

## 2.3 Процедуры «построители»

Данная процедура необходима для формирования расписания из упорядоченного вектора работ. В рассматриваемой задаче имеется m различных исполнителей, поэтому, в общем случае, соотношение между первой работой из вектора и множеством возможных исполнителей не однозначно. Поэтому при построении расписания будем нагружать такого исполнителя, у которого временные затраты на выполнение этой работы минимальны.

# **3. Программная реализация**

## 3.1 Функциональные возможности программы

1. Программа предоставляет пользователю возможность создать проект, включающий в себя работы, исполнителей, матрицу временных затрат на выполнение и другие характеристики.
2. Созданный проект можно сохранить как Excel-файл, а также возможно загружать или изменить ранее сохраненные проекты.
3. Проверка корректности вводимых данных.
4. Построение допустимого расписания и вывод его в виде графика загруженности исполнителей
5. Подсчет величины штрафа за просрочку выполнения работ.

## **3.2 Основные формы, классы и функции программы**

### 3.2.1 Общая информация о программе

Для решения поставленной задачи был выбран язык объектно-ориентированного программирования C# на платформе .Net Framework 4.5.2.

### 3.2.2 Формы программы

Для работы пользователя с программой было создано 4 формы. Ознакомиться с их видом можно в Приложении №1.

1 форма является основной. Ее задача заключается в общем представлении выбранного проекта и его составляющих: список работ, множество исполнителей и их время выполнения работ. Кроме того, пользователю предлагается открыть существующий, создать новый или изменить текущий проект. Так же здесь выводится построенное для этого проекта расписание.

2 форма позволяет пользователю создать новый проект, заполнив необходимые поля, а также предоставляется полная информация по имеющимся работам и исполнителям.

3 и 4 форма позволяют пользователю создавать новые или изменять текущие работы и исполнителей соответственно.

### 3.2.3 Структура данных

Были сконструированы 3 класса объектов: Project, Job и Worker. Данные классы содержат свойства реальных объектов и временные характеристике, необходимые для решения поставленной задачи. Так, например, работа в нашей задаче включает такие характеристики как, название, время раннего начала и позднего окончания выполнения, количество и множество предшествующих работ, штраф за опоздание и другие.

Более подробно ознакомиться со структурой классов можно в Приложении№2.

### 3.2.4 Основные функции

1. Функция формирования фронта работ в момент времени time

|  |
| --- |
| public Job[] ReadyJobs(int time) |

1. Функция построителя расписания проекта за период time.

|  |
| --- |
| public static int[,] CreateSchedule(Project project, int time) |

1. Функция чтения сохраненного проекта их файла Excel

|  |
| --- |
| public static Project OpenProject(string FileName) |

### 3.2.5 Хранение входной информации

Для хранения созданных проектов были выбраны Excel-таблицы. Это обусловлено тем, что Excel имеет богатый инструмент для работы с данными, в том числе построение различных видов диаграмм. Кроме того, при просмотре созданного файла проекта информация достаточна понятна и удобна для пользователей. Для использования таблиц, в программу добавлена работа с COM(Component Object Model) - объектами, используя библиотеку Microsoft.Office.Interop.Excel.

## 3.3 Эксперимент

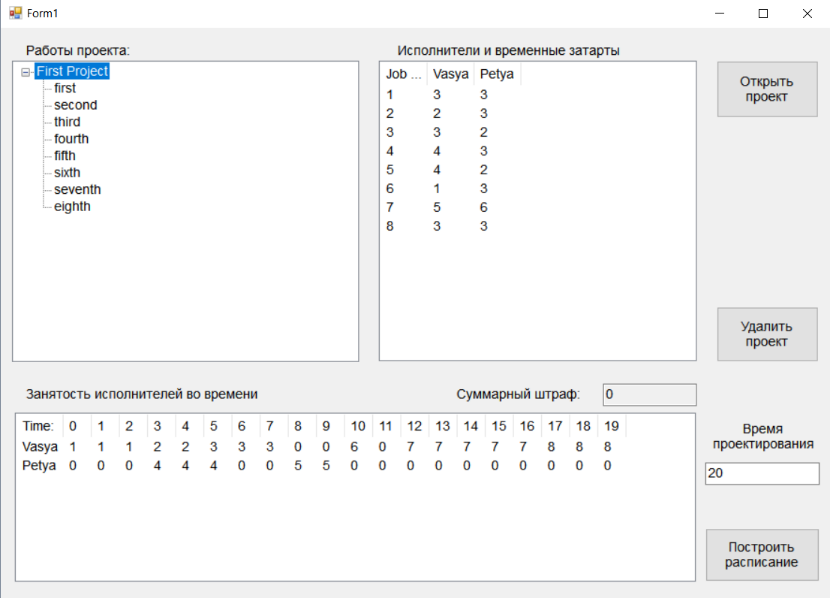
Было проведено ряд различных экспериментов. Рассмотрим 2 элементарных из них.

1. Входные данные: в проект входят 8 работ и 2 исполнителей

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер работы | Название работы | время раннего начала | время позднего окончания | Штраф на 1 такт за позднее выполнение | Количество предшественников | Предшеству-ющие работы | | |
| 1 | first |  |  | 2 | 0 |  | | |
| 2 | second |  |  | 2 | 1 | 1 | | |
| 3 | third |  | 8 | 2 | 1 | 2 | | |
| 4 | fourth |  |  | 2 | 1 | 1 | | |
| 5 | fifth | 8 |  | 2 | 1 | 4 | | |
| 6 | sixth |  |  | 2 | 1 | 5 | | |
| 7 | seventh | 12 |  | 2 | 1 | 4 | | |
| 8 | eighth |  | 20 | 2 | 3 | 3 | 6 | 7 |

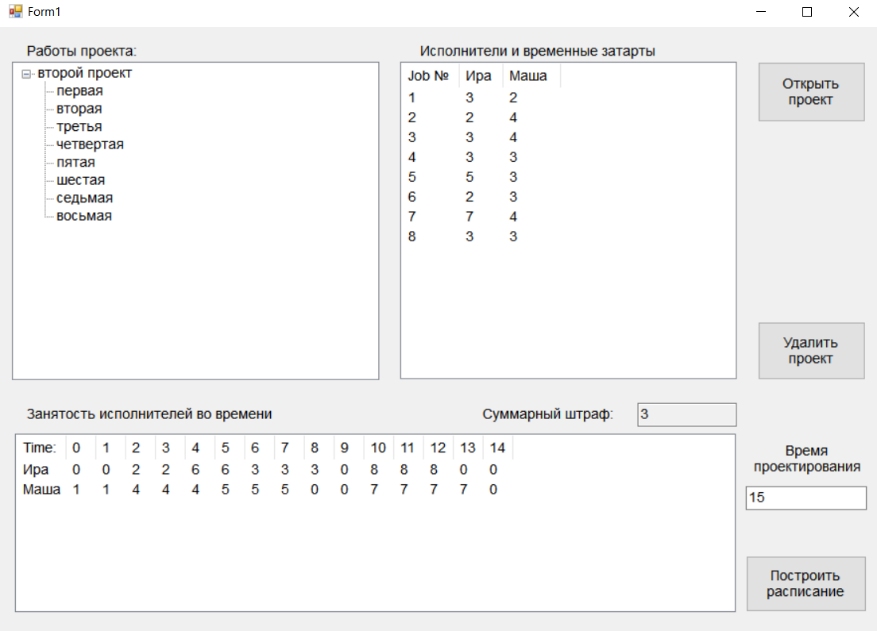
Время выполнения работ исполнителями:

|  |  |  |
| --- | --- | --- |
| Номер работы | Вася | Петя |
| 1 | 3 | 3 |
| 2 | 2 | 3 |
| 3 | 3 | 2 |
| 4 | 4 | 3 |
| 5 | 4 | 2 |
| 6 | 1 | 3 |
| 7 | 5 | 6 |
| 8 | 3 | 3 |

Полученные результаты:

1. Входные данные: 8 работ, 2 исполнителей

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер работы | Название работы | время раннего начала | время позднего окончания | Штраф на 1 такт за позднее выполнение | Количество предшественников | Предшествующие работы | | | |
| 1 | первая | 0 | 3 | 2 | 0 |  | | | |
| 2 | вторая | 0 | 5 | 8 | 1 | 1 | | | |
| 3 | третья | 0 | 5 | 1 | 1 | 1 | | | |
| 4 | четвертая | 0 | 5 | 10 | 1 | 1 | | | |
| 5 | пятая | 0 | 8 | 3 | 1 | 1 | | | |
| 6 | шестая | 0 | 10 | 7 | 1 | 1 | | | |
| 7 | седьмая | 10 | 15 | 2 | 4 | 2 | 3 | 4 | 5 |
| 8 | восьмая | 10 | 20 | 2 | 2 | 3 | 6 | 7 | |

Время выполнения работ исполнителями:

|  |  |  |
| --- | --- | --- |
| Номер работы | Ира | Маша |
| 1 | 3 | 2 |
| 2 | 2 | 4 |
| 3 | 3 | 4 |
| 4 | 3 | 3 |
| 5 | 5 | 3 |
| 6 | 2 | 3 |
| 7 | 7 | 4 |
| 8 | 3 | 3 |

Полученные результаты:

По полученным результатам мы можем сделать некоторые выводы о работе программы:

1. Программа работает, на выход выдает допустимое решение. В некоторых случаях оно является оптимальным, как в 1ом эксперименте.
2. Но во 2 эксперименте, не трудно видеть, что можно получить более хорошее решение поменяв порядок выполнения 3 и 6 работы, что приведет к уменьшению суммарного штрафа.

Таким образом, мы можем сделать вывод, что программа работает. Однако, есть потенциал для улучшения получаемого расписания. Для этого стоит уделить внимание перестановочным процедурам преобразования фронта работ в упорядоченное множество.

# Заключение

В рамках данной работы получилось реализовать систему построения расписания с User Interface для OC Windows на основе фронтального алгоритма. Данное приложение способно получить допустимое решение и предоставить график занятости каждого исполнителя.

В течение данного проекта было получено большое количество опыта и знаний в различных областях программирования, проектирования и алгоритмизации. Были изучены метод фронтального построения расписания их виды и модификации, получен опыт в работе с COM-объектами на примере работы с таблицами Excel.

Однако на этом данная тема не исчерпана. Есть много возможностей улучшить полученный результат и усовершенствовать полученное приложение. Уже сейчас есть представление ориентировочных шагов дальнейшей работы, рассмотрим несколько из них:

* В первую очередь необходимо уделить внимание получаемому расписанию: усовершенствовать алгоритм путем добавления новых перестановочных процедур и временных характеристик для множества работ.
* Во-вторых, добавить возможность построения расписания для нескольких проектов. При этом множество исполнителей придется выносить за рамки одного проекта, что вызовет некоторые трудности при заполнении матрицы временных затрат.
* В-третьих, использовать более удобные и надёжные методы хранения входной информации, например, базы данных.
* В-четвертых, для большей автоматизации и удобства добавить разделение работ на разные типы со своей сложностью. В свою очередь исполнители смогут выполнять определенный тип работ с определенной эффективностью. Эти новые характеристики будут формировать матрицу временных ресурсов, что избавит пользователя от большого объема работы.
* Ну и конечно, улучшить интерфейс программы, сделать его более дружелюбным к пользователю, сведя заполнение входных данных к минимуму.

# **Список литературы**

1. Прилуцкий М.Х., Кумагина Е.А. Задачи распределения разнородных ресурсов в сетевых канонических структурах // Перспективные информационные технологии и интеллектуальные системы. 2000. № 4. С. 46–52.
2. Молчанов В.В. Интернет книга «Практика программирования на С# для Windows и Web в Microsoft Visual Studio» http://wladm.narod.ru/C\_Sharp/
3. М.Х.Прилуцкий, С.Е.Власов. Многостадийные задачи теории расписаний с альтернативными вариантами выполнения работ
4. Кожухов Н.И. Алгоритм решения задачи оптимального распределения работ в сетевых канонических структурах // Лесотехнический журнал 4/2014 УДК 004.02

# Приложение