Департамент образования города Москвы

Государственное бюджетное общеобразовательное учреждение

города Москвы

«Гимназия № 1505 «Московская городская педагогическая гимназия-лаборатория»»

**ДИПЛОМНОЕ ИССЛЕДОВАНИЕ**

на тему:

**Компьютерное моделирование в экономике: задачи теории расписаний**

Выполнила:

Стулова Елена Вячеславовна, 10А класс

Руководитель

Пяткина Галина Александровна

Москва

2016/2017уч.г.

Оглавление

[Введение 2](#_Toc478680751)

[Глава 1 Информационные модели задач теории расписаний 4](#_Toc478680752)

[§1.1. Информационные модели 5](#_Toc478680753)

[§1.1.1. Этапы разработки информационной модели 6](#_Toc478680754)

[§1.2. Задачи теории расписаний 8](#_Toc478680755)

[§1.2.1. Задача о шлюзе 8](#_Toc478680756)

[§1.2.2. Задача о двух станках 10](#_Toc478680757)

[Глава 2 Компьютерная модель алгоритма Джонсона 12](#_Toc478680758)

[§1.1. Алгоритм Джонсона 12](#_Toc478680759)

[§1.2. Математическая модель алгоритма Джонсона 14](#_Toc478680760)

[§1.2.1. Вывод 16](#_Toc478680761)

[Заключение 17](#_Toc478680762)

[Список литературы 18](#_Toc478680763)

[Книги 18](#_Toc478680764)

[Электронные ресурсы 18](#_Toc478680765)

[Приложение 18](#_Toc478680766)

# Введение

Данная дипломная работа посвящена задачам теории расписаний. Теория расписаний – это раздел дискретной математики, в котором рассматриваются проблемы упорядочения. Задачи теории расписаний рассматривают комплексы работ, связанные с общим объектом, направленные на достижение определенной цели, к примеру, на нахождение наиболее дешевого или быстрого порядка выполнения данных работ.
 К задачам, в которых необходимо найти менее затратное решение, можно отнести задачу о шлюзе, в которой несколько кораблей в порядке очереди должны пройти через шлюз с минимальным ущербом от простоя.
 Задача о двух станках относится к типу задач, в котором требуется найти наиболее быстрый алгоритм решения, то есть минимальное время.

Я считаю эту тему актуальной, так как очень часто мы сталкиваемся с задачами теории расписаний, в которых от нас требуется найти наиболее дешевое и быстрое решение среди большого количества комбинаций и вариантов. К примеру, это связанно с расписанием поездов, кораблей, с этим также можно столкнуться на любом производственном предприятии. Но не стоит забывать, что и вся наша повседневная жизнь неразрывно связанна с расписанием: выбор наиболее быстрого маршрута, организация своего времени, список покупок…

Проблемой дипломной работы я считаю то, что не всегда можно быстро и верно решить задачу с большим количеством данных и условий, когда от времени нахождения решения может зависеть многое.

Целью же диплома является изучение алгоритма решения задач теории расписания, а также создание моделей, которые помогут в решении задач конкретных типов.

Для достижения данных целей я поставила перед собой следующие задачи:

* Изучить литературу по данной теме
* Изучить модели и этапы разработки информационной модели
* Разобраться в задачах теории расписаний
* Разобраться подробнее в алгоритме Джонсона
* Создать математическую модель для алгоритма Джонсона
* Проанализировать полученную модель
* Написать единый текст диплома

Для выполнения этих задач дипломной работы мне потребуются следующие источники:

1. Учебник по информатике углубленного уровня (К. Ю. Поляков, Е. А. Еремин). В этом источнике подробно описан алгоритм решения задач о шлюзе и двух станках.
2. Учебное пособие по исследованию информационных моделей элективного курса (Н. Д. Угринович). Этот источник поможет мне разобраться в создании информационных моделей и научит пользоваться ими.
3. Несколько интернет ресурсов, которые также помогут мне разобраться в алгоритме решения задач теории расписаний.

Данная дипломная работа будет состоять из введения, теоретической и практической частей, заключения и списка литературы. В теоретической части я опишу алгоритм решения задач теории расписаний, а также этапы разработки информационных моделей. В практической же части будет создана математическая модель алгоритма Джонсона.

# Глава 1 Информационные модели задач теории расписаний

## §1.1. Информационные модели

В нашей жизни огромное количество информации, которую сложно понять без некоторой систематизации или применения модели.

Модель – это упрощенная копия объекта, которая случит для получения информации о нем. Первые модели появились еще несколько тысяч лет назад и окружают нас до сих пор. Мы сталкиваемся с ними в школе, в магазинах, на улице… Но модель не может полноценно заменить на объект моделирования, она отражает лишь те свойства и характеристики, которые соответствуют поставленной цели. Существуют несколько видов моделей:

* Натуральные модели – модели, созданные с использованием физических материалов, их можно потрогать, они материальны. Примером такой модели будет являться школьный глобус.
* Идеальные модели – модель, построенная в воображении человека, то есть в реальном мире ее не существует. Примерами будут являться идеальный газ и математический маятник.
* Информационные модели – “это совокупность информации об объекте, описывающая свойства и состояние процесса или явления, а также связи и отношения с окружающим миром”1[[1]](#footnote-1). Как и в случае с идеальной моделью, она нематериальна. Примерами будут чертежи и таблицы.

Рассмотрим подробнее информационные модели. Существует несколько видов информационных моделей, таких как:

* Наглядные модели – модели выражают свойства объекта с помощью визуализации. Примерами будут рисунки и графики.
* Воображаемые модели – это один из этапов создания информационной модели, проходящий в воображении человека.
* Образно-знаковые модели – модели, выражающие свойства объекта исследования с помощью знаков любого вида. Примеры: таблицы, чертежи.
* Знаковые модели – это модели, выраженные посредством любого знакового языка. Они, в свою очередь, тоже делимы на группы: математические модели, программные тексты и другие.

Математические модели – это модели, выраженные с помощью математических формул и символики. Именно математическая модель будет далее рассматривать в дипломе.

## §1.1.1. Этапы разработки информационной модели

Наиболее быстрый и простой способ исследования и разработки информационных моделей - это создание их с помощью компьютера. Это позволяет изучать изменения моделей в зависимости он значения той или иной переменной. Процесс создания информационной модели делится на несколько этапов.

 На первом этапе исследования объекта строится описательная информационная модель. Это выполняется для выделения всех важных параметров и свойств объекта, которые важны для успешного проведения исследования. Остальными же параметрами пренебрегают.[[2]](#footnote-2)

На втором этапе создается формализованная модель, во время которого, описательная информационная модель форматируется посредством любого формального языка. В такой модели с помощью математических формул и действий определяются соотношение между положением начальных и конечных значений параметров изучаемого объекта, а также применяются ограничения к допустимым свойствам.

 Тем не менее, не во всех случаях удается подобрать формулы, которые точно определяют искомые величины через данные параметры. В таком случае для получения результатов с заданной точностью применяются приближенные математические методы.

 На третьем этапе формализованная модель преобразуют в компьютерную модель для понимания ее компьютером. Существуют два отличных метода построения этой модели:

* Создание проекта с использованием одного из языков программирования
* Создание компьютерной модели с помощью электронных таблиц или любого другого приложения[[3]](#footnote-3)

В данной дипломной работе будет использоваться первый способ создания компьютерной модели.

В процессе третьего этапа модель визуализируется, это позволяет организовать удобную работу человека с компьютером на этапе исследования модели.

Четвертый этап посвящен компьютерному эксперименту. На данном этапе компьютерная модель в виде программы запускается. Если же компьютерная модель существует в виде, к примеру, электронных таблиц, то строятся диаграммы, графики или проводятся любые действия с моделью.

На пятом этапе проводится анализ полученных результатов и корректировка модели. Если же результат отличаются от тех, что были получены при исследовании информационной модели, то делается вывод, что на предыдущих этапах была допущена ошибка или неточность. К примеру, в самом первом этапе могла быть допущена ошибка при выявлении важных параметров. В случае выявлении ошибки следует провести корректировку модели.

## §1.2. Задачи теории расписаний

Задачи теории расписаний – это задачи, созданные для ускорения выполнения комплекса работ, связанных с поиском наипростейшего решения.[[4]](#footnote-4) Существует два основных типа задач теории расписаний: задача о двух станках и задача о шлюзе. Обе эти задачи относятся к поиску самого дешевого и наименее затратного по времени решения.

 Но поиск самого дешевого решения в большей степени относится к задаче о шлюзе. Эта задача появилась еще в XIX веке. Шлюз может пропускать лишь по одному судну по очереди. И если создается очередь. То нужно определить порядок прохождения кораблей с учетом потерь от простоя.

 Задаче о двух станках более всего связана с поиском наиболее быстрого решения. На двух станках нужно обработать N деталей, каждая из которых обрабатывается сначала на одном, а затем на другом станке. Время обработки детали на первом и втором станке известно. Цель задачи состоит в том, что нужно найти порядок обработки с минимальным временем. Порядок обработки деталей с минимальным временем T называется оптимальным.

## §1.2.1. Задача о шлюзе

Рассмотрим пример решения задачи о шлюзе.

Известно количество кораблей N проходящих через шлюз, время прохождения судов сквозь шлюз в часах ti  и ущерб от одного часа простоя в денежных единицах. Индекс времени обозначает порядковый номер судна в очереди. Ui обозначает ущерб от часа простоя. Таким образом, время простоя в очереди судна будет равен: t1 + t2 + t3, а денежный ущерб будет составлять: u4(t1 + t2 + t3).

 Показатель экономической эффективности работы шлюза неразрывно связан с общим ущербом от простоя судов. Например, если в очереди стоят 4 судна, то общий ущерб будет вычисляться так:

S = u2 ⬝t1 + u3(t1 + t2) + u4(t1 + t2 + t3).

Таким образом, если перед шлюзом находится N судов, то суммарный ущерб от этого будет выражен формулой:

$$S= \sum\_{i=2}^{N}u\_{i}\sum\_{j=1}^{i-1}t\_{j}$$

Цель состоит в том, чтобы найти такой порядок прохождения судов через шлюз, при котором S будет минимальна.

Таким образом, *минимум величины S достигается в том случае, если суда пропускаются в порядке убывания величины* $\frac{u\_{i}}{t\_{i}}$ *.*[[5]](#footnote-5)Это можно доказать на примере двух судов, подошедших к шлюзу. Время их прохождения через шлюз одинаково, t1=t2, а стоимость от простоя отличается. В таком случае, первым следует пропустить судно с наибольшей стоимостью простоя. Если же стоимость простоя одинакова, u1=u2, то вперед следует пропустить судно с наименьшим временем шлюзования.
 При вычислении можно использовать как критерий убывания величины $\frac{u\_{i}}{t\_{i}}$, так и равносильный ему критерий возрастания величины $\frac{t\_{i}}{u\_{i}}$ .

Рассмотрим данную задачу на конкретном примере.

## §1.2.2. Задача о двух станках

Рассмотрим математическую модель задачи о шлюзе.

 Допустим, имеются два станка, один – токарный, а другой – шлифовальный. По условию необходимо обработать *m* деталей, каждую из которых следует обработать сначала на одном станке, затем – на втором. Считается, что известно время обработки *i*-й детали на *j*-м станке, и равняется оно *tij.* Задача состоит в том, чтобы найти такой порядок обработки деталей на двух станках, при котором общее время T было минимальным.

 К примеру, у нас имеется 5 деталей, которые нужно обработать сначала на токарном станке, а затем на шлифовальном. В данной таблице представлено время выполнения работ на шлифовальном и токарном станках для каждой детали.

Таблица 1

|  |  |  |
| --- | --- | --- |
| **№ детали** | **Время вытачивания** | **Время шлифовки** |
| 1 | 3 | 6 |
| 2 | 7 | 2 |
| 3 | 4 | 7 |
| 4 | 5 | 3 |
| 5 | 7 | 4 |

Если рассчитывать время выполнения работ, начиная с первой делали. То есть отсчет времени начнется с начала обработки первой детали на первом станке и закончится концом обработки последней детали на втором станке. На токарном станке новая деталь начинает обрабатываться только после окончания обработки предыдущей. На шлифовальном станке новая деталь начинает обрабатываться после того, как закончится ее обработки на токарном станке, а второй станок окажется свободным, то есть предыдущая деталь уже обработалась. Исходя из этого, таблица расчета суммарного времени обработки пяти деталей заполняется следующим образом.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **№ детали** | **Время окончания вытачивания детали** | **Время окончания шлифовки детали** | **Время простоя 2-го станка** |
| 1 | 3 | 3 + 6 = 9 | 3 |
| 2 | 3 + 7 = 10 | 10 + 2 = 12 | 1 |
| 3 | 10 + 4 = 14 | 14 + 7 = 21 | 2 |
| 4 | 14 + 5 = 19 | 21 + 3 = 24 | 0 |
| 5 | 19 + 7 = 26 | 26 + 4 = 30 | 2 |

Столбцы следует заполнять поочередно, то есть сначала первый, потом второй. Во второй столбец вносится время простоя второго станка. В первый раз вносится время ожидания шлифовального станка в момент ожидания окончания обработки первой детали на первом станке, далее время простоя от ожидания последующих деталей. В отличие от 2-го станка, 1-й никогда не простаивает.

При выполнении работ в таком порядке, время обработки деталей занимает 30 минут, учитывая так же 8 минут от простоя 2-го станка.
 Таким образом, выбранный путь изготовления деталей не является оптимальным, так как время, затраченное на работу, не будет минимальным. Но существует алгоритм, который позволяет найти оптимальный путь изготовления, а именно – алгоритм С. М. Джонсона.

# Глава 2 Компьютерная модель алгоритма Джонсона

## §1.1. Алгоритм Джонсона

 Алгоритм для решения задачи о двух станках был разработан С. М. Джонсоном в 1950 году. Главной задачей этого алгоритма было минимизирование времени простоя 2-го станка. Опишу сам алгоритм.

Существует матрица со временем обработки каждой детали на каждом из двух станков. В первом столбце матрицы записаны номера деталей.

Таблица 3

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | t1,1 | t2,1 |
| 2 | t1,2 | t2,2 |
| 3 | t1,3 | t2,3 |
| 4 | t1,4 | t2,4 |
| 5 | t1,5 | t2,5 |

В столбцах со временем ti,j обработки деталей необходимо найти наименьшее значение. Если таких значения два, то можно выбрать любое из них. Если это время относится к первому станку, то вся строка, включая номер детали и время обработки на втором станке, переносится в начало матрицы, если же время относится ко второму, то строка переносится в конец. Затем повторить те же действия для остальных строк, исключая уже перемещенные. Таким образом, после *m* шагов будет получен оптимальный порядок обработки.
 Продемонстрирую применение алгоритма Джонсона на данной выше таблице. Для построения очереди выполнения обработки деталей с наименьшим общим временем нужно проделать 5 шагов. Цветом будут выделены перемещенные строки.

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 2 | 7 | 2 |
| 3 | 4 | 7 |
| 4 | 5 | 3 |
| 5 | 7 | 4 |

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 3 | 4 | 7 |
| 4 | 5 | 3 |
| 5 | 7 | 4 |
| 2 | 7 | 2 |

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 3 | 4 | 7 |
| 4 | 5 | 3 |
| 5 | 7 | 4 |
| 2 | 7 | 2 |

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 3 | 4 | 7 |
| 5 | 7 | 4 |
| 4 | 5 | 3 |
| 2 | 7 | 2 |

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 3 | 4 | 7 |
| 5 | 7 | 4 |
| 4 | 5 | 3 |
| 2 | 7 | 2 |

|  |  |  |
| --- | --- | --- |
| № детали | 1-й станок | 2-й станок |
| 1 | 3 | 6 |
| 3 | 4 | 7 |
| 5 | 7 | 4 |
| 4 | 5 | 3 |
| 2 | 7 | 2 |

 В результате перестановок мы получаем порядок – 1, 3, 5, 4, 2. При выполнении обработки деталей в данном порядке время работы будет минимальным. Вычислим это время:

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| № детали | Время окончания работ на 1-м станке | Время окончания работ на 2-м станке | Время простоя 2-го станка |
| 1 | 3 | 3 + 6 = 9 | 3 |
| 3 | 3 + 4 = 7 | 9 + 7 = 16 | 0 |
| 5 | 7 + 7 = 14 | 16 + 4 = 20 | 0 |
| 4 | 14 + 5 = 19 | 20 + 3 = 23 | 0 |
| 2 | 19 + 7 = 26 | 26 + 2 = 28 | 3 |

 Таким образом, на выполнение обработки деталей на двух станках будет затрачено 28 минут. Это на 2 минуты меньше, за счет уменьшения времени простоя 2-го станка на 2 минуты, чем ранее рассчитанное время без использования алгоритма Джонсона.

## §1.2. Математическая модель алгоритма Джонсона

Основной проблемой при работе с алгоритмом Джонсона – это большое количество числовых значений, а также подсчетов, связанных с нахождением времени обработки деталей. Целью практической части является написание программы по алгоритму Джонсона, которая поможет быстрее справляться с решением задачи. Средой для реализации данной программы выбран язык программирования Delphi. Эта среда является языком высокого уровня объектно-ориентированного программирования. Она проста в изучении и использовании, дает возможность наглядного представления результатов. Проектная работа состоит из формы с размещенными на ней визуальными объектами: кнопками, полями ввода и вывода, - и программного кода, связанного к каждым из объектов.



Рисунок 1

На рисунке 1 представлена форма проекта в среде Delphi. Пользователь вводит в поле с подписью “количество деталей” число, которое соответствует количеству деталей. Далее следует нажать кнопку “создать таблицу”. После создания таблицы, в колонки нужно занести время выполнения работ на первом и втором станке построчно. При нажатии кнопки “Алгоритм Джонсона” выполняется алгоритм Джонсона, то есть идет сортировка строк для нахождения оптимального порядка с наименьшим временем выполнения работ. “Подсчет времени” активирует подсчет времени обработки, а также подсчитывает время простоя. Конечное время выводится во второе поле “время выполнения работ”. Кнопка “закрыть” отвечает за закрытие программы. Далее приведены блок-схемы для перечисленных выше кнопок.

StringGrid1.Cells[2,0]:='2станок' ;

StringGrid1.DefaultColWidth := 80;

начало

StringGrid1.ColCount:=3;
задаем количество столбцов в таблице

StringGrid1.RowCount:=StrToInt(Edit1.Text)+1;
Задаем количество строк

StringGrid1.Visible:=True;

v:=StrToInt(Edit1.Text);

StringGrid1.Cells[0, y]:=IntToStr(y);

StringGrid1.Cells[1,0]:='1станок';

StringGrid1.Height:=35\*(StrToInt(Edit1.Text)+1);

StringGrid1.Width:=82\*4;

Y:=1..StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text) StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text)

StringGrid1.DefaultRowHeight:= 30;

Кнопка “создать”

нет

j:=0..length(a[0])- 1

s := 0;

f := n;

начало

v := StrToInt(Edit1.Text);

SetLength(a, v, 3);

SetLength(b, v, 3);

a[i,j]:=StrToInt(StringGrid1.Cells[j,i+1]);

n := v-1;

i:=0..length(a)- 1 (Edit1.Text) StrToInt(Edit1.Text) StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text)

Кнопка “Алгоритм Джонсона”

n>(-1)

min := a[0, 1];

j:=1..2

i:=0..n (Edit1.Text) StrToInt(Edit1.Text) StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text)

a[i, j] <= min

нет

да

да

Примечания:

-v- количество строк

-n- количество строк в массиве

-f,s-границы массива

- length-длина массива

- SetLength-задание длины массива

imin := i;

jmin := j;

min := a[i, j];

jmin = 2

j :=0 .. 2

b[f, j] := a[imin, j];

f := f - 1;

да

нет

j := 0 .. 2

b[s, j] := a[imin, j];

s := s + 1;

imin < n

i := imin .. n - 1

нет

да

j := 0 .. 2

a[i, j] := a[i + 1, j];

n := n - 1;

StringGrid1.Cells[3, i] := IntToStr(StrToInt(StringGrid1.Cells[1, i]) - StrToInt(StringGrid1.Cells[2, i - 1]));

StringGrid1.Cells[2, 1] := IntToStr(StrToInt(StringGrid1.Cells[3, 1]) + StrToInt(StringGrid1.Cells[2, 1]));

начало

StringGrid1.ColCount := 4;

StringGrid1.Cells[3, 0] := 'простой 2 ст.';

StringGrid1.Cells[1, i] := IntToStr(StrToInt(StringGrid1.Cells[1, i]) + StrToInt(StringGrid1.Cells[1, i - 1]));

StringGrid1.Cells[3, 1] := StringGrid1.Cells[1, 1];

i := 2 .. v (Edit1.Text) StrToInt(Edit1.Text) StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text)

Кнопка “считать”

нет

i := 2 .. v (Edit1.Text) StrToInt(Edit1.Text) StrToInt(Edit1.Text)StrToInt(Edit1.Text) StrToInt(Edit1.Text)

StringGrid1.Cells[3, i] := '0';

StrToInt(StringGrid1.Cells[3, i]) < 0

StringGrid1.Cells[2, i] := IntToStr(StrToInt(StringGrid1.Cells[2, i - 1]) + StrToInt(StringGrid1.Cells[3, i]) + StrToInt(StringGrid1.Cells[2, i]));

да

Edit2.Text := StringGrid1.Cells[2, v];

## §1.2.1. Вывод

На рисунке 2 представлен вид программы после применения алгоритма Джонсона, то есть нажатия на одноименную кнопку. Изначально в таблице были введены данные из таблицы 1.



Рисунок 2

На рисунке 3 представлен вид программы после применения кнопки “подсчет времени”. После применения кнопки появляется столбец со временем простоя второго станка. Идет подсчет времени обработки каждой детали на каждом станке, итоговый ответ выводится в поле “Время выполнения работ”. Итоговый ответ совпадает с ответом, полученным после выполнения подсчета в таблице 4. Это позволяет убедиться, что программа работает верно, для данного набора чисел. Полный код программы представлен в разделе приложения.



Рисунок 3

# Заключение

В ходе исследовательской работы были выполнены поставленные цели – изучение задач теории расписаний, в частности алгоритма Джонсона, объяснение принципа их действия, создание программы для подсчета времени обработки деталей на двух станках и реализации алгоритма Джонсона. Задачи, решенные для достижения поставленных целей:

* Изучить литературу по данной теме
* Изучить модели и этапы разработки информационной модели
* Разобраться в задачах теории расписаний
* Разобраться подробнее в алгоритме Джонсона
* Создать математическую модель для алгоритма Джонсона
* Проанализировать полученную модель
* Написать единый текст диплома

Первая часть дипломной работы включает теорию по теме данной работы. В ней представлена общая информация по построению информационных моделей, описаны задачи теории расписаний, в частности – задача о двух станках и задача о шлюзе. Во второй части дано подробное описание решения задачи о двух станках с помощью алгоритма Джонсона, а также демонстрирует верность работы программы созданной на основе данного алгоритма. Тема остается актуальной до момента прекращения применения в мире задач теории расписаний.

# Список литературы

## Книги

Поляков, К.Ю. Информатика. Углублённый уровень: учебник для 11 класса: в 2 ч. Ч.1 / К.Ю. Поляков, Е. А. Ерёмин. – М.: БИНОМ. Лаборатория знаний, 2013. -240с.

 Угринович Н.Д. Исследование информационных моделей. Элективный курс: Учебное пособие. / Н.Д. Угринович. - 2-е изд., исп. и доп. - М.: БИНОМ. Лаборатория знаний, 2006. – 200с.: ил.

## Электронные ресурсы

А. А. Корбут и Ю. Ю. Финкельштейн. Дискретное программирование, Серия «Экономико-математическая библиотека» [Электронный ресурс]. 1969. — 368 с. – Режим доступа: <http://edu.alnam.ru/book_dpr.php?id=15> .

Разработка Института дистантного образования Российского университета дружбы народов, Информационное моделирование [Электронный ресурс]. -2006. - <http://www.ido.rudn.ru/nfpk/inf/inf9.html>.

Шпаргалки по дисциплине планирование на предприятии, Плановый баланс доходов и расходов предприятия [Электронный ресурс]. -2006-20012. – Режим доступа: <http://financial-opp.ru/shpargalki-po-planirovaniyu-na-predpriyatii/30-shpargalki-po-operativnoproizvodstvennomu/708-zadacha-o-dvuh-stankah.html>.

# Приложение

unit Unit1;

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, StdCtrls, Grids;

type

 TForm1 = class(TForm)

 StringGrid1: TStringGrid;

 Edit1: TEdit;

 Label1: TLabel;

 Button1: TButton;

 Button2: TButton;

 Button5: TButton;

 Label2: TLabel;

 Edit2: TEdit;

 Button3: TButton;

 procedure Button1Click(Sender: TObject);

 procedure Button2Click(Sender: TObject);

 procedure Button5Click(Sender: TObject);

 procedure Button3Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form1: TForm1;

 v:integer;

implementation

{$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

var

y : integer;

begin

StringGrid1.ColCount := 3;

StringGrid1.RowCount := StrToInt(Edit1.Text)+1;

StringGrid1.Visible:=True;

v:=StrToInt(Edit1.Text);

for y := 1 to StrToInt(Edit1.Text) do

StringGrid1.Cells[0, y] := IntToStr(y);

StringGrid1.Cells[1,0]:='1станок';

StringGrid1.Cells[2,0]:='2станок' ;

StringGrid1.DefaultColWidth := 80;

StringGrid1.DefaultRowHeight:= 30;

StringGrid1.Height:=35\*(StrToInt(Edit1.Text)+1);

StringGrid1.Width:=82\*4;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

close

end;

procedure TForm1.Button5Click(Sender: TObject);

var a:array of array of integer;

b:array of array of integer;

i, j, imin, jmin, s, f, n, min: integer;

begin

v := StrToInt(Edit1.Text);

SetLength(a, v, 3);

SetLength(b, v, 3);

for i:=0 to length(a)- 1 do

begin

for j:=0 to length(a[0])- 1 do

begin

a[i,j]:=StrToInt(StringGrid1.Cells[j,i+1]);

end;

end;

n := v-1;s := 0;f := n;

while n > (-1) do

 begin

 min := a[0, 1];

 for i := 0 to n do

 for j := 1 to 2 do

 if a[i, j] <= min then

 begin

 min := a[i, j];

 imin := i;

 jmin := j;

 end;

 if jmin = 2 then begin

 for j := 0 to 2 do b[f, j] := a[imin, j];

 f := f - 1; end

 else begin

 for j := 0 to 2 do b[s, j] := a[imin, j];

 s := s + 1; end;

 if imin < n then begin

 for i := imin to n - 1 do

 for j := 0 to 2 do a[i, j] := a[i + 1, j]; end;

 n := n - 1;

 end;

for i:= 0 to length(b) - 1 do

for j:= 0 to length(b[0]) - 1 do

StringGrid1.Cells[j,i+1]:=IntToStr(b[i,j]);

end;

procedure TForm1.Button3Click(Sender: TObject);

var

i: integer;

begin

StringGrid1.ColCount := 4;

StringGrid1.Cells[3, 0] := 'простой 2 ст.';

 for i := 2 to v do StringGrid1.Cells[1, i] := IntToStr(StrToInt(StringGrid1.Cells[1, i]) + StrToInt(StringGrid1.Cells[1, i - 1]));

StringGrid1.Cells[3, 1] := StringGrid1.Cells[1, 1];

StringGrid1.Cells[2, 1] := IntToStr(StrToInt(StringGrid1.Cells[3, 1]) + StrToInt(StringGrid1.Cells[2, 1]));

 for i := 2 to v do

 begin

StringGrid1.Cells[3, i] := IntToStr(StrToInt(StringGrid1.Cells[1, i]) - StrToInt(StringGrid1.Cells[2, i - 1]));

if StrToInt(StringGrid1.Cells[3, i]) < 0 then StringGrid1.Cells[3, i] := '0';

StringGrid1.Cells[2, i] := IntToStr(StrToInt(StringGrid1.Cells[2, i - 1]) + StrToInt(StringGrid1.Cells[3, i]) + StrToInt(StringGrid1.Cells[2, i]));

end;

Edit2.Text := StringGrid1.Cells[2, v];

end;

end.

1. [Электронный ресурс].- <http://www.ido.rudn.ru/nfpk/inf/inf9.html> - данные соответствуют 14.12.16 [↑](#footnote-ref-1)
2. Угринович Н.Д. Исследование информационных моделей. Элективный курс: Учебное пособие. / Н.Д. Угринович. - 2-е изд., исп. и доп. - М.: БИНОМ. Лаборатория знаний, 2006. – 200с.: ил. [↑](#footnote-ref-2)
3. Угринович Н.Д. Исследование информационных моделей. Элективный курс: Учебное пособие. / Н.Д. Угринович. - 2-е изд., исп. и доп. - М.: БИНОМ. Лаборатория знаний, 2006. – 200с.: ил. [↑](#footnote-ref-3)
4. Поляков, К.Ю. Информатика. Углублённый уровень: учебник для 11 класса: в 2 ч. Ч.1 / К.Ю. Поляков, Е. А. Ерёмин. – М.: БИНОМ. Лаборатория знаний, 2013. -240с. [↑](#footnote-ref-4)
5. Поляков, К.Ю. Информатика. Углублённый уровень: учебник для 11 класса: в 2 ч. Ч.1 / К.Ю. Поляков, Е. А. Ерёмин. – М.: БИНОМ. Лаборатория знаний, 2013. -240с. [↑](#footnote-ref-5)