ФГОУ СПО Тульский экономический колледж

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

**к курсовой работе**

**по предмету: “КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ”**

**на тему: Разработка программы**

**«Оптимизация сетевого графика по времени»**

Специальность: 230105 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ.

Выполнил

студент гр. 316-П \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Щербаков Н.В./

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Юрченко В.Н./

Щекино

2009г.

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

По предмету: “Компьютерное моделирование”

Студента гр. 316-П Щербакова Николая Владимировича

Специальность: 230105 Программное обеспечение вычислительной

техники и автоматизированных систем.

ТЕМА: Разработка программы «Оптимизация сетевого графика по времени»

ИСХОДНЫЕ ДАННЫЕ: карточка задания

|  |  |  |  |
| --- | --- | --- | --- |
| I | J | Объем работ | Число ресурсов |
| 0 | 1 | 25 | 5 |
| 0 | 2 | 30 | 6 |
| 0 | 3 | 20 | 4 |
| 1 | 4 | 22 | 5 |
| 2 | 3 | 10 | 4 |
| 2 | 4 | 15 | 3 |
| 2 | 5 | 5 | 4 |
| 3 | 6 | 25 | 5 |
| 4 | 6 | 20 | 4 |
| 5 | 6 | 5 | 4 |
| 6 | 7 | 30 | 6 |

Производительность рабочих = 5 ед.

Общее число задействованных в процессе производства рабочих = 50

ВЫХОДНЫЕ ДАННЫЕ: файл с выходными полями

Определить оптимальное время выпуска продукции.

Сформировать файл с полями: I, J, Необходимое число ресурсов, Число

ресурсов, Продолжительность, События, Резерв времени.

ВЫХОДНЫЕ ДАННЫЕ: файл с выходными полями

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| I | J | Объем работ | Не обход. ресурсов | Число ресурсов | Продолжительность | События | Резерв времени |

ГРАФИЧЕСКАЯ ЧАСТЬ

выполнить: блок – схему программы

Главной целью является: Определить оптимальное время выпуска продукции. Усвоение методов сокращения сроков выпуска продукции путем оптимальной расстановки имеющихся трудовых ресурсов. А так же закрепление практических навыков работы с ЭВМ.

**Содержание**

# Реферат

Ведение

1. Технические характеристики

1.1. Постановка задачи

1.1.1. Сущность задачи

1.1.2. Математическая модель задачи

1.2. Описание логической структуры

1.3. Описание и организация входных данных

1.3.1. Описание входной и выходной информации

1.3.2. Организация ведения информационной базы

1.4. Описание программных средств

1.4.1. Описание среды программирования

1.4.2. Описание программы

1.5. Описание технических средств

1.5.1. Описание и обоснование минимальной конфигурации технических средств

Заключение

Литература

Приложение A

**Реферат**

Курсовая работа по теме «Оптимизация сетевого графика по времени» выполнена в среде программирования Borland Delphi 5. Она состоит из пояснительной записки и графической части. Объем пояснительной записки составляет 35 листов формата A4, а также включает в себя 4 приложения:

* Приложение A – листинг программы;
* Приложение Б – главная форма;
* Приложение В – дополнительные окна и окно помощи;
* Приложение Г – структура меню программы.

Графическая часть представлена схемой программы, выполненной на листе формата A1.

Ключевые слова в курсовой работе:

оптимизация, работа, событие, ресурс, время, критический путь.

# Введение

При проектировании, каких либо намеченных работ строят графики, в которых наглядно отображаются все процессы и порядок проведения этих работ. Чтобы наиболее рационально использовать время, выделенное на эти работы проводят оптимизацию этих графиков таким образом, чтобы последовательность проведения работ, обеспечивала своевременное выполнение каждой из работ, для оптимизации этих работ используются сетевые графики.

Сетевой график - это схематическое изображение операций и элементов производственного процесса (программы), а также взаимосвязи порядка и последовательности их выполнения.

Оптимизация сетевого графика позволяет выяснить самое кратчайшее время выполнения данных работ, а так же позволяет наиболее выгодно распределить ресурсы, затраченные на выполнение данных работ. При оптимизации сетевых графиков определяется наиболее кратчайший путь для выполнения всего комплекса работ.

**1. Технические характеристики**

**1.1. Постановка задачи**

Требуется при ограниченном лимите рабочих минимальное критическое время выпуска продукции в результате оптимальной расстановки трудовых ресурсов.

Задается:

* Общее количество рабочих.
* Необходимое количество изделий по каждой работе.

Производительность труда – одинаковая.

Продолжительность каждой работы обратно пропорциональна количеству рабочих, занятых на этой работе.

Под работами принято понимать определенные процессы, которые могут иметь различное содержание. Прежде всего, это реальные хозяйственные или технологические процессы, требующие затрат времени и ресурсов для их осуществления

**1.1.1. Сущность задачи**

Оптимизация сетевого графика заключается в следующем: требуется при определенном количестве рабочих определить минимальное критическое время выпуска конечной продукции. В результате оптимальной расстановки трудовых ресурсов.

Задается общее количество этих самых ресурсов, в нашем случае – рабочих, занятых на рассматриваемом комплексе работ, а вместо продолжительности работ задаются по каждой работе необходимое число изделий подлежащих обработке.

Производительность труда всех рабочих одинакова. Продолжительность каждой работы обратно пропорциональна количеству рабочих, занятых на этой работе.

Важнейшие способы сокращения времени затрачиваемого на выполнения комплексов работ заключаются в следующем:

1) Прежде всего производится пересмотр топологии сети, т.е. изменения состава последовательности выполнения или взаимосвязи между работами. Такой пересмотр топологии сети возможен в том случае, когда для одних и тех же работ могут быть применены различные способы их выполнения или может быть изменена последовательность выполнения работ, в частности заменена последовательность параллельностью при выполнении отдельных работ.

В определенных случаях может обнаруживаться нецелесообразное использование ресурсов на многочисленных параллельных работах. Если параллельные работы выполняются рабочими одной специальности и квалификации при использовании определенных типов машин и механизмов, то целесообразно сделать так, чтобы те из работ, которые не лежат на критическом пути (и, следовательно, имеют резервы времени), выполнялись бы последовательно. Тогда часть ресурсов можно использовать для ускорения критических работ.

Используется также возможность более детального членения работ с тем, чтобы ускорить начало каких-либо критических работ.

2) Другим способом сокращения критического пути является переход от нормальных продолжительностей работ, находящихся на критическом пути, к минимально возможным, в первую очередь за счет перемещения ресурсов с работ, находящихся на некритических путях.

Перемещение ресурсов с некритических работ на критические должно, разумеется, производиться в разумных пределах, с тем, чтобы некритические работы сами не стали критическими.

3) В отдельных случаях для ускорения выполнения критических работ могут привлекаться дополнительные ресурсы со стороны.

Проведя тем или иным способом сокращение продолжительности работ на критическом пути, снова выполняют обработку сетевого графика, чтобы выявить достаточность принятых мер, а также узнать, не появились ли новые критические пути. Если новый вариант сети также не обеспечивает соблюдения директивных сроков выполнения всего комплекса работ, то вся процедура повторяется снова до тех пор, пока не будет получен удовлетворительный результат. Для сокращения количества пересчетов сетевого графика и его показателей проводят мероприятия по одновременному сокращению работ не только на критическом, но и на околокритических (т.е. близких к критическому) путях. Многократные пересчеты сетевого графика не являются обременительными и не занимают много времени и затрат труда специалистов, если все эти изменения осуществляются по специально разработанной документации и если все операции по перерасчету сетевого графика проводятся на ПЭВМ.

В отдельных случаях, особенно при жестких ограничениях в ресурсах, окончательным результатом расчетов по оптимизации сетевого графика может явиться обоснованное доказательство невозможности уложиться в заданный срок без устранения этих ограничений или пересмотра заданных технико-экономических характеристик. Это положение рассматривается соответствующими инстанциями, имеющими право решать указанные вопросы.

После того как определена продолжительность всего комплекса работ, осуществляется привязка всей сети к календарной шкале времени. В результате создается директивный документ – план-график проведения работ. В нем указываются даты начала и окончания каждой работы, а также величины резервов времени.

Директивный план-график рассылается ответственным исполнителям, после чего начинается выполнение комплекса работ в соответствии с разработанным графиком.

Однако нередко возникает необходимость, прежде чем приступить к работам, проверить распределение ресурсов по календарным срокам выполнения работ. Дело в том, что, определяя продолжительность выполнения той или иной работы, включенной в сетевой график, мы исходили из фиксированного количества ресурсов, выделенного для ее выполнения. Правда, в момент составления сетевого графика еще не бывают известны точные календарные сроки выполнения работ. Это выясняется лишь позже, после расчета основных временных показателей сетевого графика. Выделяя для каждой работы определенное количество ресурсов, мы могли предполагать их достаточность, исходя из наших возможностей. Однако не всегда учитываются совпадения сроков выполнения работ, при проведении которых используются одни и те же ресурсы. В результате может оказаться, что на ту или иную календарную дату потребуется ресурсов больше, нежели мы располагаем. Вот это и требуется проверить, прежде чем приступать к выполнению планируемого комплекса работ. Поэтому необходимо быть уверенным, что ограниченные, как правило, ресурсы позволят выполнить намечаемые работы.

Продолжительность всего комплекса работ равна продолжительности критического пути. Анализ сетевого графика позволяет выделить те работы от которого она действительно зависит. Их число как правило, не велико по сравнению с общим количеством работ.

При обычных методах планирования комплекса работ для сокращения общего срока разработки не редко стремятся уменьшить продолжительность всех или большинства работ. Такой подход как правило, надает соответствующего эффекта, так как требует дополнительных затрат и незначительно сокращает продолжительность выполнения всего комплекса работ.

Важнейшим способом сокращения критического пути является использование определенной части ресурсов, для ускорения критических работ.

**1.1.2. Математическая модель задачи**

Для того чтобы выполнить оптимизацию сетевого графика, необходимо рассчитать и определить:

1. Необходимое число ресурсов для каждой работы:

#### A = B / C

где A - необходимое число ресурсов;

B - объем работ;

C – производительность.

2. Продолжительности работ:

### D = A / F

где D – продолжительность работы;

A – необходимое число ресурсов;

F – число ресурсов.

3. Ранние сроки наступления событий:

tр (j) = max {tр (i) + t (ij)}, i<j, tр (0) = 0

где i, j – номера событий;

tр (i, j) – ранний срок наступления (i, j) события;

t (ij) – срок выполнения работы (i-j);

i<j – событие i предшествует j.

4. Поздние сроки наступления событий:

tп (i) = min {tп (j) - t (i,j)}, i<j

где i, j – номера событий;

tп (i, j) – поздний срок наступления (i, j) события;

t (ij) – срок выполнения работы (ij);

5. Полный резерв времени для каждой работы:

Rп (i,j)= tп (j) - tр (i) - t (i,j)

где i, j – номера событий;

tп (j) – поздний срок наступления события j;

tр (i) – ранний срок наступления события i;

t (ij) – срок выполнения работы (i-j).

6. События, лежащие на критическом пути. Такие события имеют равные ранние и поздние сроки наступления.

7. Критические работы. То есть работы, у которых начальное и конечное события критические.

8. Критический путь. Рассчитывается как сумма продолжительностей критических работ.

Далее проводится непосредственно оптимизация сетевого графика, т.е. в целях сокращения продолжительности критического пути программа предлагает пользователю перераспределить ресурсы с работ, находящихся на некритических путях, на выполнение работ, не имеющих резервов времени

1.2. Описание логической структуры

Блок-схема системы приведена на листе формата А1.

Блок 1 –начало работы;

Блок 2 – выбор из пунктов меню;

Блок 3 – выбор пункта меню файл;

Блок 4 - выбор подпункта меню “Открыть файл”;

Блок 5 – загрузка файла;

Блок 6 – выбор подпункта меню «сохранить файл»;

Блок 7 – сохранение файла

Блок 8 – выбор подпункта меню «Выход»;

Блок 9 – запрос на окончание работы;

Блок 10 – завершение работы программы;

Блок 11 – выбор меню «действие»;

Блок 12 – выбор подпункта меню «Оптимизация»;

Блок 13 –происходит оптимизация;

Блок 14 – выбор подпункта меню «прибавить строку»;

Блок 15 –прибавление строки;

Блок 16 – выбор подпункта меню «удалить строку»

Блок 17 –удаление строки;

Блок 18 - выбор пункта подменю “Очистить все”;

Блок 19 – очистка содержимого таблицы;

Блок 20 – выбор пункта меню “Помощь”;

Блок 21 – выбор из пунктов подменю;

Блок 22 – выбор подпункта меню “О задаче”

Блок 23 – открытие новой формы и вывод её на экран;

Блок 24 – выбор подпункта меню “О разработчике”

Блок 25 – открытие новой формы и вывод её на экран;

Блок 26 – выбор пункта подменю “Help”

Блок 27 – открытие новой формы и вывод её на экран;

Листинг программы приведен в Приложении “А”

**1.3. Описание и организация входных данных**

**1.3.1. Описание входной и выходной информации**

Данная задача использует внутреннюю входную информацию в виде полей и переменных описание которых приведено ниже:

|  |  |  |
| --- | --- | --- |
| Название поля | Идентификатор | Тип |
| Начальное событие | I | Целый |
| Конечное событие | J | Целый |
| Объем работ | Объем работ | Целый |
| Число ресурсов | Число ресурсов | Целый |
| Производительность | Производительность | Целый |
| Общее число ресурсов | Общее число ресурсов | Целый |

Начальное событие, Конечное событие, Объем работ, Число ресурсов и Производительность берутся из карточки задания, а Общее число ресурсов вычисляется автоматически как сумма по столбцу, Число ресурсов и выводятся в не редактируемое поле ввода.

Выходная информация также полностью отображается на главной форме в таблице и в не редактируемых полях ввода и содержит:

|  |  |  |
| --- | --- | --- |
| Наименование реквизита | Идентификатор | Тип |
| Необходимо ресурсов | Необход. ресурсов | вещественный |
| Продолжительность | Продолжительность | вещественный |
| События | События | целый |
| Резерв времени | Рез. времени | вещественный |
| Продолжительность критического пути | Продолжительность критического пути | вещественный |
| Всего необходимо ресурсов | Общее число ресурсов | вещественный |

Необходимо ресурсов, Продолжительность, События, Ранний срок, Поздний срок, Резерв времени выводятся в таблицу, Продолжительность критического пути, Всего необходимо ресурсов выводятся в не редактируемые поля ввода.

**1.3.2. Организация ведения информационной базы**

Организация ведения информационной базы осуществляется за счет наличия таких функций как ввод и сохранение исходных данных. В эту информационную базу данных входят такие элементы программы как промежуточные массивы, адреса памяти где хранятся результаты вычислений и т. д.

Промежуточные массивы создаются на время выполнения программы, в них производятся промежуточные вычисления, результаты которых потом заносятся в таблицу.

Входная информация. Карточка задания и представляет собой пример задачи которую необходимо решить с использованием компьютера, оттуда также берутся исходные данные которые используются в тех самых промежуточных массивах и решениях, используются как входная информация для программы которая в свою очередь использует ее для решения задачи.

Также имеется так называемая условно-постоянная информация(эта информация может быть изменена по желанию пользователя), которая представляет собой файл с исходными данными например той же самой карточки с задачей то есть это говорит о том что в программе предусмотрена такая функция как сохранение исходных данных и их загрузка.

А также имеется постоянная информация – это помощь, которая в процессе работы программы никак не меняется.

**1.4. Описание программных средств**

**1.4.1. Описание среды программирования**

#### Для реализации проекта «Оптимизация сетевого графика» с использованием персонального компьютера была выбрана среда программирования Borland Delphi 5.0 кампании Borland.

Среда Delphi - это сложный механизм, обеспечивающий высокоэффективную работу программиста. В основе нее лежит язык Object Pascal, разработанный профессором Высшего технического училища (г. Цюрих, Швейцария) Никлаусом Виртом. Предложенный им алгоритмический язык он назвал именем великого французского ученого Блэза Паскаля (1623-1662). В качестве стандартного Pascal был утвержден в 1979 г. Первая версия Delphi была выпущена в начале 1995 года и явилась результатом разработки, которая велась компанией Borland в течение двух с половиной лет. Программный продукт был назван именем греческого города.

В первую очередь Delphi предназначена для профессионалов-разработчиков корпоративных информационных систем. Однако она предназначен не только для программистов-профессионалов, но и для всех тех, кто используют компьютер с чисто прикладной целью, и кому необходимо быстро решить какие-то свои задачи, не привлекая для этого программистов со стороны. В Delphi удачно сочетаются средства визуального проектирования приложений и оптимизирующий компилятор. Delphi является единым полноценным средством промышленной разработки систем клиент/сервер. В состав среды входит обширная библиотека компонентов. С другой стороны, программист всегда сможет прибегнуть к низкоуровневым ассемблерным процедурам. Можно создавать приложения в визуальном режиме. Работая с Delphi, программист может с помощью нажатия одной клавиши создать выполняемый файл в формате EXE, однако, при необходимости, можно компилировать и файлы DLL, драйверов устройств, а также консольных приложений.

Синтаксис Delphi включает буквы, цифры, шестнадцатеричные цифры, специальные символы, пробелы и зарезервированные слова.

Буквы – это буквы латинского алфавита от a до z и от A до Z, а также знак подчеркивания "\_". В языке нет различия между заглавными и строчными буквами алфавита, если только они не входят в символьные и строковые выражения. Цифры – арабские цифры от 0 до 9. Каждая шестнадцатеричная цифра имеет значение от 0 до 15. Первые 10 значений обозначаются арабскими цифрами 0…9, остальные шесть – латинскими буквами A…F или a…f. Специальные символы Delphi – это символы + - \* / > < = <> >= <= : ; ' () [] {} и др.

Особое место в алфавите языка занимают пробелы, к которым относятся любые символы в диапазоне кодов от 0 до 32. Эти символы рассматриваются как ограничители идентификаторов, констант, чисел, зарезервированных слов. Несколько следующих друг за другом пробелов считаются одним пробелом.

Тип данных определяет значения, которые может иметь переменная, и операции, выполняемые над этой переменной. Существует несколько предопределенных типов данных, которые можно разделить на 3 группы: перечисляемые, вещественные типы и строки. Три наиболее важных предопределенных перечисляемых типа – Integer (целочисленные), Boolean (логический) и Char (символьный). Полный список перечисляемых типов: Integer, Cardinal, ShortInt, SmallInt, LongInt, Byte, Word, Boolean, ByteBool, WordBool, LongBool, Char, ANSIChar, WideChar. Вещественные типы представляют собой разнообразные форматы чисел с плавающей запятой, меньше всего памяти требуется для хранения чисел типа Single, затем Real, Double и Extended, кроме них есть еще два необычных типа данных Comp – описывает очень длинные целые числа и Currency (доступен только в Delphi) – имеет 4 десятичных знака после запятой.

В Delphi определенны следующие операции: арифметические операции, логические операции, операции со строками, операции над множествами, операции отношения и операция @ (операция получения адреса). Логические операции применимы к операндам целого и логического типов, но не выделены в отдельную группу. Приоритетность операций достаточно проста. Основной упор объектно-ориентированной модели программных компонент в Delphi делается на максимальном реиспользовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды Delphi. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в Delphi написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать. Исключения представлены в виде объектов, содержащих специфическую информацию о соответствующей ошибке (тип и местонахождение ошибки). Разработчик может оставить обработку ошибки, существовавшую по умолчанию, или написать свой собственный обработчик. Обработка исключений реализована в виде exception-handling blocks (также еще называется protected blocks), которые устанавливаются ключевыми словами try и end. Существуют два типа таких блоков: try...except и try...finally.

Процедуры и функции позволяют разделить логику программы на более мелкие и управляемые фрагменты и аналогичны подпрограммам в других языках. Как и в основном блоке программы, все действия в процедурах и функциях заключаются в begin и end. Каждый из этих сегментов кода выполняет конкретную задачу. Если в содержащемся в процедуре операторе внутри модуля процедуры используется идентификатор процедуры, то процедура будет выполняться рекурсивно (будет при выполнении обращаться сама к себе). В Delphi существуют разница между функциями и процедурами. Стандартными процедурами являются Exit и ShowMessage, стандартными функциями являются StrToInt, FloatToStr, Random, математические функции и д.р.

Ввод/вывод связан с обменом информацией между оператором памяти и вешнеми носителями информации. Для ввода и вывода данных в языке предусмотрены стандартные встроенные программы, которые называются процедурами. Процедура ввода вызывается с помощью оператора ввода Read , процедура вывода – с помощью оператора вывода Write .

Оператор ввода служит для ввода данных в процессе выполнения программы. Значение вводимых данных получают переменные. Данные могут быть разбиты на отдельные строки. Различают два вида операторов ввода: Read, Readln. Допускается целые, действительные и символьные данные. Логические данные вводить не разрешается. Для правильного ввода символьных данных рекомендуется перед каждым оператором ввода символьных данных ставить оператор перехода на новую строку Readln , чтобы их ввод всегда осуществлялся с новой строки.

Оператор вывода имеет две формы записи: Write, Writeln. Допускается вывод данных с форматами и без них. Файловая переменная в Delphi - это любая переменная файлового типа. В Delphi имеются три класса файлов: типизированный файл, текстовый файл и нетипизированный файл. Работа с файлами осуществляется просто, присутствует возможность произвольного доступа к данным. В Delphi имеется возможность стыковки с другими программными средствами. Например, в ней можно использовать DLL (динамически компонуемые библиотеки), написанные на других языках программирования. Кроме того, программы, написанные на других языках, могут использовать DLL, написанные на Borland Delphi. DLL, таким образом, идеально подходит для использования в программных проектах, реализуемых на нескольких языках.

На Delphi имеется стандарт. В состав документации среды программирования входят: руководство пользователя; руководство по написанию компонент; документация по построению приложений, работающих с базами данных; руководство по генератору отчетов ReportSmith; руководство по SQL Links. В состав Delphi входит 5 интерактивных обучающих систем, документация в электронном виде и около 10 Мб справочной информации, а также множество дополнительных программ.

#### С помощью среды программирования Borland Delphi 5 была разработана программа «Оптимизация сетевого графика по времени». Пользовательский интерфейс в программе довольно прост (Главная форма приведена в приложении Б) в него входят множество визуальных компонентов(так как язык программирования Object Pascal – объектно-ориентированный, а его порождение Delphi – язык визуального программирования) из Delphi такие как:

* GroupBox - Компонент представляет собой окно для группировки объектов, этот компонент был использован, чтобы сгруппировать такие объекты как кнопки SpeedButton (о них будет рассказано ниже)
* MainMenu – Компонент используется, чтобы создать главное меню для формы.
* StringGrid – Это компонент - Таблица он необходим для ввода исходных данных.
* Panel - Это компонент используется для создания панели с надписью.
* Label - Это компонент – Метка необходим для ввода текста в форму
* SpeedButton - кнопка, которая используется, чтобы выполнить команды. На кнопки могут быть установлены легко запоминающиеся графические образы, которые помогают запомнить действия, происходящие после их нажатия.
* OpenDialog - Компонент представляет собой диалог Окна для выбора и открытия файлы.
* SaveDialog- Компонент представляет собой диалог Окна для выбора и сохранения файлы.

**1.4.2. Описание программы**

Программа «Оптимизация сетевого графика» разработана в среде программирования Borland Delphi 5 и работает под управлением операционной системы Windows (Текст программы приведен в приложении А).

Интерфейс программы достаточно прост и лёгок в обращении. Программа работает довольно просто: в ней имеется главное меню (Приложение Г) и «Панель операций» с кнопками.

Главное меню имеет структуру:

<Файл> <Помощь>

<Сохранить данные> <Справка>

<Загрузить данные> <О программе>

<Выход>

Для пунктов меню <Сохранить данные>, <Загрузить данные>, <Выход>, <Справка> имеются «Горячие клавиши» (Горячие клавиши – клавиши используемые для быстрого вызова программы) :

<Сохранить данные> F2 <Загрузить данные> F3

<Выход> F10 <Справка> F1

Если в программе нажать функциональную клавишу F2 или при выборе пункта меню <Файл>/< Сохранить файл> то программа автоматически проверит все ли ячейки заполнены при обнаружении что хотя бы одна ячейка не заполнена программа выводит сообщение об ошибке и сохранение не происходит, а если при проверке выясняется что все ячейки заполнены, то появляется диалоговое окно сохранения файла (приложение В), в котором указывается папка, в котором будет находиться файл с данными и в нем будут (если они есть) указанны другие файлы с данными, требуется ввести имя этого файла, программа автоматически прибавит к имени файла расширение \*.Gra, используемое для обозначения файла с данными, затем происходит сам процесс сохранения.

А если нажать функциональную клавишу F3 или при выборе пункта меню <Файл>/< Открыть файл> то появляется диалоговое окно открытия файла (приложение В), в котором указывается папка, в котором находиться файлы с данными, требуется выбрать файл (в диалоговом окне отображаются только те файлы, в которых сохранены данные из графика) с расширением \*.Gra, при необходимости имеется такая возможность как открытие любого файла если, например данные из графика были сохранены без расширения после выбора необходимого файла и нажатия на кнопку «Открыть» программа открывает файл и он выводится в таблицу.

При нажатии функциональной клавиши F10 или при выборе пункта меню

<Файл>/<Выход> программа выводит диалог (приложение В) в котором спрашивает пользователя действительно ли он хочет завершить работу?: и если да то завершает свою работу.

При нажатии функциональной клавиши F1 или при выборе пункта меню

<Помощь>/<Справка> программа выводит справку(приложение В) в виде HTML – документа.

**1.5. Описание технических средств**

Для создания программы были использованы программные средства:

* Borland Delphi 5.0 – написание, отладка, компиляция и исполнение текста программы.
* Microsoft Office XP – Новейший пакет офисных программ фирмы Microsoft
1. Microsoft Word – создание помощи в формате HTML (Web документ).
2. Microsoft Excel – создание таблиц для отчёта.
* CorelDraw – создание изображений для помощи.
* Xara 3D – создание 3х мерных надписей для справки
* Photoshop 6.0 – Редактирование изображений.
* Благодаря тому что программа создана в среде программирования Borland Delphi 5.0 она поддерживается любыми версиями Microsoft Windows: Microsoft Windows 9x, Microsoft Windows Millennium, Microsoft Windows 2000, Microsoft Windows XP.

**1.5.1. Описание и обоснование минимальной конфигурации технических средств**

Для нормальной работы данной программы требуется компьютер с минимальной конфигурацией:

Процессор с частотой не ниже Pentium 100 MHz

Оперативной памятью не менее 16 Mb

Места на диске требуется не менее 5 Mb.+30 Mb (Для Windows)

Для наилучшей работы программы желателен монитор поддерживающий разрешение экрана 800х600 с углублением цветовой палитры от 8 (256 цветов) до 16 (65535 цветов) бит.

Для более комфортной работы необходима мышь.

Из программного обеспечения не требуется ни каких дополнительных пакетов. В качестве операционной системы подходит любая версия Windows начиная с Windows 95 .

**Заключение**

Курсовая работа на тему «Оптимизация сетевого графика по времени» была выполнена в заданный срок. Для разработки программы использовалась среда программирования Borland Delphi 5.0. в программе разработан удобный пользовательский интерфейс в виде меню. Каждый пункт меню содержит несколько пунктов подменю. Для демонстрации программы была разработан пример ввода данных в матрицу. Разработана форма в которой отображается входная и выходная информация. Организована справочная система. Составлена пояснительная записка.

## Литература

1. Экономико-математические методы в снабжении. –М.: «Экономика», 1971.

2. И. М. Разумов. Сетевые графики в планировании. –М.: 1981.

3. Бобровский С. Delphi 5: учебный курс. –СПб.: Издательство “Питер”, 2000.

# Дата выдачи \_\_\_\_\_\_\_\_\_\_ Срок окончания \_\_\_\_\_\_\_\_\_\_\_

# Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Приложение A**

Листинг программы

unit Unit1;

interface

uses

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

 Menus, StdCtrls, Grids, Buttons, ExtCtrls, Errors, OleCtnrs;

type

 TGeneral = class(TForm)

 MainMenu1: TMainMenu;

 GroupBox1: TGroupBox;

 GroupBox2: TGroupBox;

 N1: TMenuItem;

 SpeedButton1: TSpeedButton;

 GJvjo1: TMenuItem;

 N2: TMenuItem;

 N3: TMenuItem;

 N4: TMenuItem;

 N5: TMenuItem;

 SpeedButton2: TSpeedButton;

 Sg1: TStringGrid;

 SpeedButton3: TSpeedButton;

 SpeedButton4: TSpeedButton;

 SpeedButton5: TSpeedButton;

 OpenDialog1: TOpenDialog;

 SaveDialog1: TSaveDialog;

 SpeedButton6: TSpeedButton;

 N6: TMenuItem;

 PopupMenu1: TPopupMenu;

 N7: TMenuItem;

 Sg4: TStringGrid;

 Edit1: TEdit;

 Edit4: TEdit;

 Label4: TLabel;

 Label5: TLabel;

 Label6: TLabel;

 Label7: TLabel;

 CheckBox1: TCheckBox;

 N8: TMenuItem;

 N9: TMenuItem;

 N10: TMenuItem;

 N11: TMenuItem;

 Panel1: TPanel;

 Edit2: TEdit;

 Edit3: TEdit;

 Label1: TLabel;

 Label2: TLabel;

 N12: TMenuItem;

 procedure FormCreate(Sender: TObject);

 procedure N3Click(Sender: TObject);

 procedure SpeedButton1Click(Sender: TObject);

 procedure SpeedButton4Click(Sender: TObject);

 procedure SpeedButton3Click(Sender: TObject);

 procedure SpeedButton2Click(Sender: TObject);

 procedure N5Click(Sender: TObject);

 procedure SpeedButton6Click(Sender: TObject);

 procedure SpeedButton5Click(Sender: TObject);

 procedure N6Click(Sender: TObject);

 procedure FormClose(Sender: TObject; var Action: TCloseAction);

 procedure N7Click(Sender: TObject);

 procedure Edit1KeyPress(Sender: TObject; var Key: Char);

 procedure N12Click(Sender: TObject);

 private

 { Private declarations }

 public

 l:integer;

 Count,Sum:integer;

 buf1:array[1..100] of real;

 buf2:array[0..30] of real;

 err:integer;

 { Public declarations }

 end;

var

 Proizvod : String;//Производительность

 Str : String;//колличество строк

 Rab\_I : String;//Работы

 Rab\_J : String;//Работы

 OB\_Rabot : String; //Обьем работ изделий

 KolRes : String; //Колличество рабочих

 Prodolgit : String; //Продолжительность

 Err,i,j : Integer;

 General : TGeneral;

 F : Textfile;

implementation

uses Unit2, Unit3, Unit4;

{$R \*.DFM}

procedure TGeneral.FormCreate(Sender: TObject);

begin

//General.WindowState:=WsMaximized;

General.Position:=poDesktopCenter;

General.Caption:='Оптимезация сетевого графика по времени.';

GroupBox1.Caption:='';

GroupBox2.Caption:='';

SG1.cells[0,0]:='I';

SG1.cells[1,0]:='J';

SG1.cells[2,0]:='Объем работ';

SG1.cells[3,0]:='Необход. ресурсов';

SG1.cells[4,0]:='Число ресурсов';

SG1.cells[5,0]:='Продолжительность';

SG1.cells[6,0]:='События';

SG1.cells[7,0]:='Рез. времени';

SG1.RowCount:=2;

end;

procedure TGeneral.N3Click(Sender: TObject);

begin

Close;

end;

procedure TGeneral.SpeedButton1Click(Sender: TObject);

var

i,j:Integer;

 Sum,Sum2,a: Real;

begin

Panel1.Visible:=true;

label1.Visible:=true;

label2.Visible:=true;

label4.Visible:=true;

label5.Visible:=true;

label7.Visible:=true;

Sg4.Visible:=true;

edit2.Visible:=true;

edit3.Visible:=true;

edit4.Visible:=true;

//edit5.Visible:=true;

{Выясним необходимое число ресурсов}

for i:=1 to sg1.RowCount-1 do

Begin

a:=StrToInt(sg1.Cells[2,i])/StrToInt(edit1.text);

sg1.Cells[3,i]:=FloatToStr(a);

{Выясним необходимое число ресурсов}

Sum2:=Sum2+StrToFloat(sg1.Cells[4,i]);

Edit3.Text:=FloatToStr(Sum2);

{Всего необходимых ресурсов}

Sum:=Sum+StrToFloat(sg1.Cells[3,i]);

Edit2.Text:=FloatToStr(Sum);

end;

{Продолжительность работ}

for i:=1 to sg1.RowCount-1 do

Begin

a:=StrToFloat(sg1.Cells[3,i])/StrToFloat(sg1.Cells[4,i]);

sg1.Cells[5,i]:=FloatToStr(a);

end;

end;

procedure TGeneral.SpeedButton4Click(Sender: TObject);

begin

close;

end;

procedure TGeneral.SpeedButton3Click(Sender: TObject);

begin

For I:=0 to 2 do

For j:=1 to Sg1.RowCount-1 do

Begin

if Sg1.Cells[i,j]= '' then err:=err+1;

end;

For j:=1 to Sg1.RowCount-1 do

if Sg1.Cells[4,j]= '' then err:=err+1;

if err <>0 then

begin

 MessageDlg('Ошибка'+#13+'Таблица заполнена не полностью.'+#13+'Сохранение не произведено!'

 , mtError,[mbOk], 0);

 err:=0;

end

else

 Begin

 Proizvod:=edit1.text;

 Str:=IntToStr(sg1.RowCount);

 For i:=1 to Sg1.RowCount-1 do

 Begin

 Rab\_I :=(Sg1.Cells[0,i]);

 Rab\_J :=(Sg1.Cells[1,i]);

 OB\_Rabot:=(Sg1.Cells[2,i]);

 write(f,Rab\_I);

 write(f,Rab\_J);

 write(f,OB\_Rabot);

 End;

 For I:=1 to Sg1.RowCount-1 do

 KolRes:=(Sg1.Cells[4,i])

 end;

CloseFile(f);

end;

procedure TGeneral.SpeedButton2Click(Sender: TObject);

begin

General.N7.Click;

 if OpenDialog1.Execute then

 {$I-}

 AssignFile(F,OpenDialog1.FileName); Reset(F);

 Proizvod:=edit1.text;

 Str:=IntToStr(sg1.RowCount);

 For i:=1 to Sg1.RowCount-1 do

 Begin

 Rab\_I :=(Sg1.Cells[0,i]);

 Rab\_J :=(Sg1.Cells[1,i]);

 OB\_Rabot:=(Sg1.Cells[2,i]);

 write(f,Rab\_I);

 write(f,Rab\_J);

 write(f,OB\_Rabot);

 End;

 For I:=1 to Sg1.RowCount-1 do

 KolRes:=(Sg1.Cells[4,i])

 {$I+}

CloseFile(f);

end;

procedure TGeneral.N5Click(Sender: TObject);

begin

if OpenDialog1.Execute then

end;

procedure TGeneral.SpeedButton6Click(Sender: TObject);

begin

Edit1.Text:='5';

sg1.RowCount:=12;

sg1.Cells[0,1]:=INttoStr(0);

sg1.Cells[1,1]:=INttoStr(1);

sg1.Cells[2,1]:=INttoStr(25);

sg1.Cells[4,1]:=INttoStr(5);

sg1.Cells[0,2]:=INttoStr(0);

sg1.Cells[1,2]:=INttoStr(2);

sg1.Cells[2,2]:=INttoStr(30);

sg1.Cells[4,2]:=INttoStr(6);

sg1.Cells[0,3]:=INttoStr(0);

sg1.Cells[1,3]:=INttoStr(3);

sg1.Cells[2,3]:=INttoStr(20);

sg1.Cells[4,3]:=INttoStr(4);

sg1.Cells[0,4]:=INttoStr(1);

sg1.Cells[1,4]:=INttoStr(4);

sg1.Cells[2,4]:=INttoStr(22);

sg1.Cells[4,4]:=INttoStr(5);

sg1.Cells[0,5]:=INttoStr(2);

sg1.Cells[1,5]:=INttoStr(3);

sg1.Cells[2,5]:=INttoStr(10);

sg1.Cells[4,5]:=INttoStr(4);

sg1.Cells[0,6]:=INttoStr(2);

sg1.Cells[1,6]:=INttoStr(4);

sg1.Cells[2,6]:=INttoStr(15);

sg1.Cells[4,6]:=INttoStr(3);

sg1.Cells[0,7]:=INttoStr(2);

sg1.Cells[1,7]:=INttoStr(5);

sg1.Cells[2,7]:=INttoStr(5);

sg1.Cells[4,7]:=INttoStr(4);

sg1.Cells[0,8]:=INttoStr(3);

sg1.Cells[1,8]:=INttoStr(6);

sg1.Cells[2,8]:=INttoStr(25);

sg1.Cells[4,8]:=INttoStr(5);

sg1.Cells[0,9]:=INttoStr(4);

sg1.Cells[1,9]:=INttoStr(6);

sg1.Cells[2,9]:=INttoStr(20);

sg1.Cells[4,9]:=INttoStr(4);

sg1.Cells[0,10]:=INttoStr(5);

sg1.Cells[1,10]:=INttoStr(6);

sg1.Cells[2,10]:=INttoStr(5);

sg1.Cells[4,10]:=INttoStr(4);

sg1.Cells[0,11]:=INttoStr(6);

sg1.Cells[1,11]:=INttoStr(7);

sg1.Cells[2,11]:=INttoStr(30);

sg1.Cells[4,11]:=INttoStr(6);

end;

procedure TGeneral.SpeedButton5Click(Sender: TObject);

begin

form2.show;

end;

procedure TGeneral.N6Click(Sender: TObject);

begin

form2.show;

end;

procedure TGeneral.FormClose(Sender: TObject; var Action: TCloseAction);

begin

case Application.MessageBox('Вы уверены, что хотите выйти?','Выход',mb\_yesno or mb\_iconquestion or mb\_defbutton2) of

idyes:action:=caFree;

idno:action:=caNone;

end;

end;

procedure TGeneral.N7Click(Sender: TObject);

var q,w:byte;

i:integer;

begin

for q:=0 to SG1.ColCount do

for w:=1 to SG1.RowCount do

 SG1.Cells[q,w]:='';

SG1.RowCount:=2;

Edit1.Text:='';

Edit2.Text:='';

Edit3.Text:='';

Edit4.Text:='';

//Edit5.Text:='';

end;

procedure TGeneral.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

case key of

'0'.. '9', chr (8):;

',':;

else key:=#0;

end;

end;

procedure TGeneral.N12Click(Sender: TObject);

begin

Sg1.RowCount:=Sg1.RowCount+1;

If Sg1.RowCount>30 then

 MessageDlg('Ошибка'+#13+'строк не может быть больше 30 !!!', mtError,[mbOk], 0);

Sg1.RowCount:=30;

end;

end.