МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Политехнический институт: Заочный
Кафедра «Системы автоматического управления»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой

_____/ В.И. Ширяев

«_____» _____ 2018 г.

Разработка программы автоматизированного планирования производственного задания

ООО "Фабрика ЮжУралКартон"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 09.03.01.2018. 854.00 ПЗ ВКР

Руководитель работы

главный энергетик ООО «Фабрика ЮжУралКартон»

_____/ *В.П. Фролов*

«____» _____ 2018 г.

Автор работы

студент группы **ПЗ-597**

_____/ *И. А. Фролов*

«____» _____ 2018 г.

Нормоконтролер

к. т. н., доцент

_____/ *В.Б. Садов*

«____» _____ 2018 г.

Челябинск 2018

Аннотация

ФРОЛОВ И.А. Разработка программы автоматизированного планирования производственного задания ООО "Фабрика ЮжУралКартон": ЮУрГУ (НИУ), ПИ: Заочный; 2018, 105 с. 22 ил., библиогр. список –18 наим., 13 листов слайдов презентации ф. А4.

В данном дипломном проекте рассматривается процесс создания программы автоматизированного планирования производственного задания ООО «Фабрика ЮжУралКартон». Программа подразумевает наличие баз данных сырьевых комбинаций и изделий, а так же наличие расчетной части раскроя гофрополотна и вывода статистических данных о загрузке машин и технологических потерях. Кроме того, программа обладает дополнительными опциями, такими как: редактирование баз данных; поиск по базам данных с учетом выбранных пользователем отклонений характеристик изделий; анализ сменного задания и предложение вариантов по его доработке. Программа написана на языке высокого уровня C++ в интегрированной среде разработки программного обеспечения C++ Builder 6.

В экономическом обосновании дипломного проекта описаны расчеты стоимости технологических потерь. Главная цель работы программы – их минимизация.

Выполнено описание работы программы и проведен расчет экономического эффекта.

В заключении описаны плюсы использования программы на производстве.

					09.03.01.2018.854.00 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Фролов И.А.			Разработка программы автоматизированного планирования производственного задания ООО "Фабрика ЮжУралКартон"	Лит.	Лист	Листов
Провер.		Фролов В.П.				Д	4	105
						ЮУрГУ Кафедра САУ		
Н. Контр.		Садов В.Б.						
Утверд.		Ширяев В.И.						

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Промышленная революция середины XVIII века открыла перед человечеством новые горизонты. Сложные машины и механизмы прочно вошли в нашу жизнь. Производство товаров и полуфабрикатов было поставлено на поток, таким образом, одна отрасль промышленности требовала развития других, сопутствующих отраслей. Именно здесь берет свое начало производство тары из гофрированного картона.

С конца XIX века и по сей день этот вид упаковки является одним из самых востребованных в силу своей экологичности, универсальности и низкой стоимости. Упаковка из такого материала обладает легкостью, пластичностью, прочностью и ударостойкостью, что делает её незаменимой для хранения и транспортировки товаров широкого спектра: от стекла до строительных смесей.

Производственная линия по выпуску гофрокартона называется – гофроагрегат. Эти машины развивались и совершенствовались год от года. Современный гофроагрегат – это целый комплекс автоматизированных станков, работающих как единое целое. Мощные микроконтроллеры и современное программное управление способствовали росту производительности этих машин. Однако, несмотря на все старания инженеров и программистов, человеку отводится большая роль в управлении гофроагрегатом.

Надо понимать, что сам по себе гофроагрегат – это всего лишь линия по выпуску полуфабрикатов. Чтобы заготовка стала коробкой, её необходимо «пропустить» через линию переработки. Печатно-высекательный агрегат или ПВА завершает работу, начатую гофроагрегатом, нанося флексопечать на изделие, формируя клапана короба, склеивая и складывая.

Весь путь от рулона бумаги до готового изделия нуждается в тщательной проработке. Этим и занимается отдел планирования на производстве. Необходимо учесть множество факторов, прежде чем сформированное сменное задание поступит на выполнение.

Все вышеперечисленное делает актуальным разработку программы планирования сменного задания для повышения эффективности производства и его оптимизации.

1 ОПИСАНИЕ ТЕХНОЛОГИЧЕСКОГО ПРОЦЕССА

1.1 Структура и состав предприятия

Фабрика «ЮжУралКартон» успешно занимается производством гофроупаковки различной конфигурации с 2009 года. Предприятие расположено в Челябинской области в городе Коркино, и занимает 11,3 Га земли. Деятельность компании направлена на создание качественной гофрированной упаковки, максимально соответствующей потребностям клиентов. Мощность фабрики составляет 180 миллионов квадратных метров гофрированного картона в год. Общая площадь промышленных помещений составляет около 22 000 кв.м.

Компания располагает мощными техническими ресурсами и с каждым годом увеличивает объемы производства гофротары. С момента открытия, фабрика увеличила выпуск упаковки в 7,4 раз – с 11 000 до 82 000 кв.м в год.

Фабрика осуществляет сотрудничество с предприятиями, которые используют гофротару для упаковки своей продукции, частными предпринимателями и др.

Основные клиенты:

- ОАО «Хлебпром»
- Произвостенная камания «Увелка»
- Сюзпищепром
- Равис
- Макфа
- Юниливер

Основные поставщики сырья:

- ОАО «Илим» г.Братск (целлюлоза)
- Архангельский ЦБК(целлюлоза)
- ОАО «Монди СЛПК»(целлюлоза)
- БКФ «Каменская»
- Николь-Пак
- ООО ПЦБКг.Пермь

На фабрике установлено следующее оборудование:

- Гофроагрегат итальянской фирмы «Fosber», обрезная ширина 2500мм.; максимальная рабочая скорость до 350 м/мин.
- Технологическая линия «9PA-FG (2х цветная)», Тайвань, для

изготовления 4-х клапанных ящиков.

- Технологическая линия «9PA-FG (3х цветная)», Тайвань, для изготовления 4-х клапанных ящиков.
- Технологическая линия «5PA-FG (2х цветная)», Тайвань, для изготовления 4-х клапанных ящиков.
- Технологическая линия «Ishikawa 1525», Тайвань, для изготовления ящиков сложной высечки.

1.2 Характеристика изготавливаемой продукции

Гофрированный картон – тарный картон, состоящий из чередующихся, склеенных между собой плоских и гофрированных слоев.

Необходимыми условиями для изготовления гофрокартона является наличие: пара, клея, бумаги для гофрирования и картона или бумаги для плоских слоев, а также соответствующих температурных режимов, влажности, натяжения и давления.

Типы гофрированного картона:

- **Д** – двухслойный, состоящий из одного плоского и одного гофрированного слоя;
- **Т** - трехслойный, состоящий из двух плоских и одного гофрированного слоя;
- **П** - пятислойный, состоящий из трех плоских (двух наружных и одного внутреннего) и двух гофрированных слоев.

Типы, классы и марки гофрированного картона (по ГОСТ 7376-89) представлены в таблице 1.1.

Таблица 1.1 – Типы, классы и марки гофрированного картона

Тип	Класс	Марка
Д	-	Д
Т	1	Т11, Т12, Т13, Т14, Т15
	2	Т21, Т22, Т23, Т24, Т25, Т26, Т27
П	3	П31, П32, П33, П34, П35, П36, П37

Назначение марок гофрированного картона (по ГОСТ 7376-89) представлено в таблице 1.2.

Таблица 1.2 – Назначение марок гофрированного картона

Класс	Марка	Назначение
-	Д	Изготовление вспомогательных упаковочных средств
1	Т11 - Т15	Изготовление тары и вспомогательных упаковочных средств для упаковывания продукции и изделий, способных воспринимать нагрузки штабеля
2	Т21 - Т27	Изготовление тары и вспомогательных упаковочных средств для упаковывания продукции и изделий, не способных воспринимать нагрузки штабеля
3	П35 - П37	
3	П31 - П34	Изготовление крупногабаритной высокопрочной и жесткой тары, контейнеров

Основными характеристиками марки гофрокартона являются показатели сопротивления торцевому сжатию и абсолютное сопротивление продавливанию. Данные значения по каждой марке гофрированного картона приведены в таблице 1.3.

Таблица 1.3 – Основные характеристики марки гофрокартона

Марка	Сопротивление торцевому сжатию не менее, кН/м	Абсолютное сопротивление продавливанию не менее, МПа
Трехслойный гофрокартон		
Т 21	2.2	0.7
Т 22	3.0	0.9
Т 23	3.8	1.1
Т 24	4.6	1.2
Т 25	5.4	1.3
Т 26	6.2	1.5
Т 27	7.0	1.7
Пятислойный гофрокартон		
П 31	5.0	1.1

Продолжение таблицы 1.3

Марка	Сопротивление торцевому сжатию не менее, кН/м	Абсолютное сопротивление продавливанию не менее, МПа
П 32	6.0	1.4
П 33	8.0	1.7
П 34	10.0	2.0
П 35	12.0	2.3
П 36	15.0	2.5
П 37	17.0	2.8

Профиль гофрированной бумаги в поперечном сечении выглядит как волна (Рисунок 1.1).

Рисунок 1.1 - Профиль гофрированной бумаги

Картон гофрированный изготавливается и поставляется в листах.

Характеристики профиля гофр линии по изготовлению гофрированного картона представлены в таблице 1.4.

Таблица 1.4 – Характеристика гофров

Тип гофра	Наименование гофра	Высота гофра, мм	Шаг гофра, мм
С	средний	от 3,2 до 4,4	от 6,5 до 8,0
В	мелкий	от 2,2 до 3,2	от 4,5 до 6,4
Е	микро	от 1,1 до 1,8	от 3,2 до 3,6

Размеры листов устанавливаются по соглашению сторон.

Допускаемые отклонения по размеру не должны превышать, мм:

- +20; -10 - по длине листа;
- ±5 - по ширине листа;

Косина листа не должна превышать 10 мм на 1 м длины. Измерение проводят металлической линейкой или рулеткой с точностью до 1,0 мм по ГОСТ 21102.

Картон изготавливается с обрезными кромками.

Слои гофрированного картона должны быть склеены между собой по вершинам гофров. Допускаются расклеенные участки не более 20 см² каждый, допускается расслаивание картона по кромке листа на величину не более 10 мм от края кромки. Допускается смятие гофров по кромке листа. Допускается коробление листа не более 20 мм на 1 погонный м листа картона.

На поверхности гофрированного листа картона не допускаются:

- задиры площадью более 80 см²;
- складки и морщины, нарушающие склейку;
- вмятины, нарушающие целостность поверхности картона;
- пятна неволокнистого происхождения размером более 20 мм в наибольшем измерении и общей площадью более 60 мм² на 1 квадратный метр площади листа;
- разрывы и разрезы кромки листа длиной более 10 мм.

1.3 Технологический процесс

Весь технологический процесс производства продукции можно разделить на три больших этапа: производство, переработка и упаковка. Каждый этап разбит на определенные составляющие. А так же присутствуют вспомогательные элементы производства, которые тоже участвуют в технологическом процессе. Для рассмотрения процесса в целом составлена блок-схема, показывающая переходные процессы и составляющие технологического цикла выпуска продукции и утилизации отходов производства тары из гофророванного картона и вспомогательных средств упаковки фабрики ООО «ЮжУралКартон».

Более подробно этот процесс показан на рисунке 1.2.

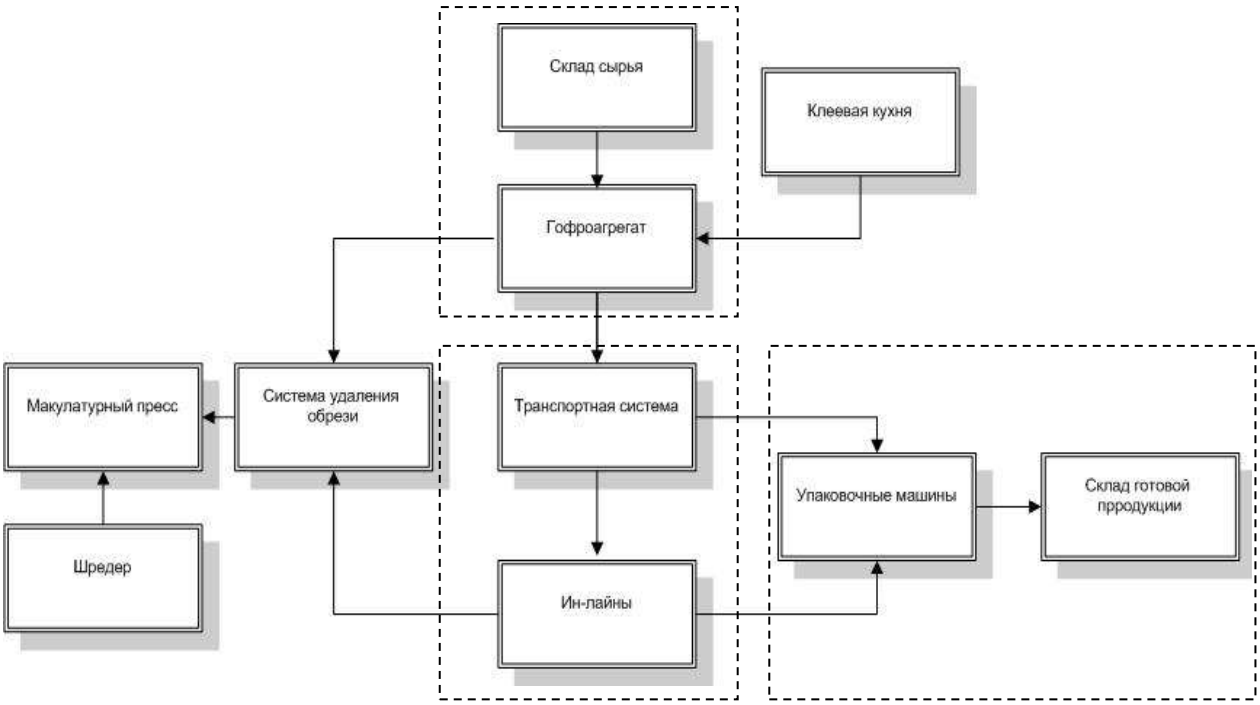

Рисунок 1.2 – Блок-схема технологического процесса производства

Элементы производства:

- Склад сырья – это специально оборудование помещение для хранения рулонов бумаги и картона.
- Автоматическая клеевая кухня SRP предназначена для приготовления крахмального клея используемого в производстве гофрокартона.
- Гофроаграгат – это производственная линия предназначенная для выпуска гофрокартона. Осуществляет гофрорование среднего слоя бумаги(fluting) и склеивание его с двумя плоскими слоями картона(liner). Затем с помощью продольной и поперечной резки полученное полотно раскраивается на заготовки.
- Транспортная система представляет собой моторизированную систему рольгангов и поворотных тележек и предназначена для приема, кратковременного хранения и транспортирования кипы листов гофрокартона к перерабатывающему оборудованию.
- Ин-лайны – машины переработки заготовок из гофрокартона. Осуществляют высечку и фальцесклейку коробов, так же нанесение печати и укладку в стопу.
- Упаковочные машины – предназначены для упаковки готовой

продукции. Их задача обвязка транспортных пакетов ПП-лентой и обмотка стрейч-пленкой.

- Склад готовой продукции – специализированное помещение для хранения готовой продукции, с последующей её отгрузкой клиенту.
- Система удаления обрези предназначена для удаления обрези кромок от рилевочно-резательных секций на ин-лайнах и гофроагрегате, измельчения и транспортировки ее в измельченном виде к месту сбора (макулатурный пресс)
- Макулатурный пресс – устройство для сбора обрези и прессования её в транспортные тюки для дальнейшей продажи.
- Шредер – устройство для измельчения крупных бумажных отходов, бракованных изделий и т.п. с последующей подачей измельченной макулатуры в макулатурный пресс.

Рулоны картона для плоских слоев гофрокартона и бумаги для гофрирования шириной не более 2,5 м и диаметром не более 1,5 м из зоны складирования склада сырья погрузчиком с боковыми захватами, обеспечивающими сохранность продукции от механических повреждений, подаются к тележкам раската гофроагрегата.

Выбор марок бумаги для гофрирования и картона для плоских слоев для изготовления гофрокартона и заготовок ящиков производится в соответствии с требованиями ГОСТов, технических условий.

В это время крахмальный клей из реактора клеевой кухни подается насосом по трубопроводу в танкеры хранения на гофроагрегат. Он необходим для склеивания между собой слоев гофрокартона.

Выпущенные гофроагретом заготовки укладываются в стопу и направляются на транспортную систему. Далее заготовки поступают с транспортной системы на рольгангы, где оператор линии переработки производит загрузку стоп заготовок на автопитатель откуда они поступают на стол подачи и непосредственно в машину.

В свою очередь во время всего производственного процесса система удаления обрези (аспирация) обеспечивает транспортировку измельченных технологических отходов от машин переработки и гофроагрегата к макулатурному прессу.

После прохода через линию переработки, готовые изделия укладываются в стопу, проходят первичную обвязку, затем, согласно схеме укладки, маленькие стопки укладываются в большую стопу, называемую

транспортным пактом, после чего транспортный пакет поступает на специальный обвязчик – паллетайзер, где происходит финальная упаковка: обвязка ПП-лентой и обмотка стрейч-пленкой.

Упакованный ТП погрузчиком вывозится на склад готовой продукции.

Если взглянуть на диаграмму производства, то можно увидеть что данный цикл производства не всегда выполняется полностью. Так, например, заготовка с транспортной системы не всегда поступает на «Ин-лайны», она может сразу поступать на упаковочные машины, так как уже является готовой продукцией (гофролист, вспомогательные упаковочные средства).

2 ПЛАНИРОВАНИЕ ПРОИЗВОДСТВА

2.1 Особенности планирования производства на фабрике ООО «ЮжУралКартон»

Процесс планирования – один из важнейших на производстве. Он является неотъемлемой частью производственной организации. Очень важно, чтобы производство велось максимально эффективно с минимальными затратами, при этом чтобы товары требуемого качества производились своевременно.

Однако планы не рождаются сами собой, так же как и продукция не может быть выпущена без участия человека. Ответственный за планирование производства должен следовать определенным правилам, таким как распределение производственных задач, отслеживание их выполнения и контроль соответствия фактической производительности плановым показателям.

Таким образом, производственное планирование является важной функцией, которая включает в себя координацию и интеграцию различных производственных процессов с целью повышения эффективности производства. Она достигается за счет правильного планирования работ, установки точной последовательности операций, корректного расписания, устанавливающего начало и завершение каждой операции, своевременную выдачу нарядов и принятие мер, необходимых для обеспечения бесперебойного функционирования предприятия.

Другими словами, планирование производства включает в себя планирование, маршрутизацию, составление производственного расписания, выдачу нарядов на изготовление продукции и контроль исполнения.

Чтобы рассмотреть процесс планирования производства на фабрике ООО «ЮжУралКартон» нужно начать с конца производственного цикла, а именно ориентироваться на готовую продукцию, ту, которую заказал клиент.

Отдел продаж на фабрике связывается с покупателем продукции, выясняя какое именно изделие из гофрированного картона необходимо произвести. После этого заказ принимается к обработке.

На этом этапе происходит составление технологических карт на изделия, и после согласования всех аспектов, заказ передается в отдел планирования. Саму же работу отдела планирования можно разбить на пять этапов.

Планирование. Первый важный шаг в планировании производства и контроле связан с тщательной подготовкой производственных планов. Производственные планы определяют, что будет производиться и где, какого типа, кем и как. Для детального планирования производственных операций, соответствующая информация может быть получена из нескольких источников на предприятии. Информация о количестве и качестве продуктов, которые будут изготовлены, может быть получена из заказов клиентов и бюджета продаж, а информация о производственных мощностях может быть получена от производственного менеджмента и инженерно-технического отдела. Таким образом, функция планирования формулирует производственные планы и переводит их в требования к персоналу, технике и материалам.

Каков бы ни был плановый период, планирование производства помогает избежать случайностей в производстве, обеспечивая непрерывный поток производственной деятельности, максимального использования производственных мощностей для минимизации эксплуатационных расходов и выполнения сроков поставки; координации различных отделов предприятия для поддержания надлежащего баланса деятельности, и, прежде всего, обеспечивая основу для контроля на предприятии.

Первичное производство на фабрике, как видно из рисунка 1.2 первого пункта, всегда связано с гофроагретатом. Именно эта линия производит гофрокартон – основу будущих коробов и прочих средств упаковки. Поэтому планирование задания прежде всего опирается на составление заказов на гофроагегат, именно здесь можно получить большие технологические потери, либо минимизировать их.

Гофроагрегат Fosber предоставляет возможность одновременно производить два типа разных по размеру заготовок, но одинаковых по качеству (производимых из одного сырья), что значительно повышает гибкость раскроя гофрополотна.

Составление производственного плана (маршрутизация). Следующей важной функцией планирования и управления производством является составление производственного плана, который включает определение маршрута передвижения сырья по различным видам оборудования и операциям на заводе. Маршрутизация включает планирование, где и кем будет проводиться работа, определение пути, по которому должна производиться работа, и необходимая последовательность операций. Чтобы найти этот путь, акцент делается на определении

эксплуатационных данных, которые, как правило, включают в себя планирование того, ГДЕ и КЕМ работа должна быть выполнена, а также требуемую последовательность операций. Эти операционные данные содержатся в стандартной карте технологического процесса, которая помогает в оформлении маршрутной диаграммы, показывающей последовательность операций и перечень станков, которые будут использоваться. Если диаграмма указывает на отсутствие определенных машин, в план может быть включен альтернативный маршрут. Возможно, эффективный маршрут может быть недоступен из-за недоступности определенных станков в определенный момент времени. Другими словами, «маршрутизация устанавливает перечень операций, их последовательность и требуемый класс станков и персонала, необходимого для этих операций".

Исходя из вышеизложенного, можно сделать вывод, что маршрутизация является одним из весьма важных элементов контроля производства, так как многие функции управления производством тесно связаны с производственными процессами и зависят от функции маршрутизации. Таким образом, очень важно решить различные проблемы, связанные с: соответствующим персоналом; полным использованием производственных мощностей и определением точного времени, необходимого на каждом этапе производства.

На фабрике «ЮжУралКартон» для перемещения и временного хранения заготовок используется транспортная система, способная вмещать до 100 тысяч квадратных метров гофрокартона. Перемещение осуществляется в автоматизированном режиме, с участием оператора транспортной системы.

Транспортная система позволяет выпускать заказы «наперед», что повышает гибкость планирования, ведь заказ, нужный в данный момент, можно сочетать с заказом, более позднего планирования.

Оператор доставляет требуемую заготовку на инлайны цеха переработки. Также возможна доставка непосредственно на упаковочные машины, и дальнейшая транспортировка с помощью вилочных погрузчиков на склад готовых изделий.

После чего продукция отгружается покупателю.

Составление расписания. Составление расписания определяет время на каждый этап производства, то есть предварительно определяет "когда должна быть сделана работа". Оно состоит из времени начала и завершения различных операций, которые будут выполнены. Другими словами, функция

составления расписания определяет, когда должна быть выполнена каждая операция и вся работа в целом. Оптимальное расписание определяет время, когда каждая операция начинается и заканчивается на указанной машине для исполнения желаемых сроков поставки. Хорошее управление подразумевает указание не только времени, когда должна начаться каждая операция, но и указание прогресса каждой производственной части, объем работы для каждой машины и наличие каждой машины для выполнения новой задачи.

Расписания могут быть двух типов: главный график (основной производственный план) и детальный график. Задачи, описанные в главном графике, используются для плана загрузки всего завода, в то время как детальные графики используются для планирования производственных и сборочных операции, необходимых для каждого продукта.

При планировании важно не перегружать одну машину, при этом недогружая другую, тем самым теряя в эффективности производства.

Выдача заказов (диспетчеризация). Диспетчеризация является частью производственного контроля, которая переводит запланированные на бумаге работы в реальное производство. Функции диспетчеризации выполняются с учетом всех деталей составленного производственного плана и расписания. Таким образом, диспетчеризация следит за тем, что материал перемещается в нужное место производства, что инструменты находятся в правильном месте, что работа движется по маршрутной карте. Диспетчеризация осуществляет организацию непосредственной работы в соответствии с планом. Таким образом, диспетчеризация подразумевает выдачу нарядов на выполнение работ. Эти заказы на выполнение работ инициируют производство. Наряды содержат следующую информацию:

- Наименование продукции
- Номер заказа
- Количество
- Параметры заготовки (длина, ширина, биговки, если есть и профиль)
- Подразделения, задействованные в каждом этапе
- Используемое сырье и его формат
- Станки и машины, задействованные в каждой операции, и даты начала каждой операции

Контроль исполнения. Контроль исполнения является последним этапом в процессе управления производством. Эта функция предназначена

для отслеживания производственной деятельности. Цель состоит в том, чтобы обеспечить выполнение запланированного. Функция контроля состоит из данных отчетности о производстве и исследовании отклонений от заранее определенных временных графиков. Контроль включает в себя следующие функции:

Проверка, что все нужные для производства по текущим нарядам материалы, инструменты, комплектующие и аксессуары доступны на всех рабочих местах, в указанных количествах для запуска и проведения технологических операций.

Проверка хода работы и выполнения операций на различных этапах производства. Включает в себя сбор информации, относящейся к начальному и финальному времени выполнения задач и даты завершения работы, статус фактической работы относительно запланированных сроков завершения, позиции движения материалов, комплектующих и узлов в производстве и проверка результатов.

Отчетность руководителям производства обо всех значимых отклонениях для принятия корректирующих действий. Также включает отчетность для отдела производственного планирования для корректировки будущих планов.

Таким образом, производственное планирование и контроль, проходят все выше рассмотренные фазы, обеспечивает производство продукции надлежащего качества, количества и обеспечивает отгрузку в заявленные сроки. Необходимо иметь в виду, что планирование производства и контроль – процесс непрерывный, а его функции являются взаимозависимыми.

Трудности с управлением заказами и грамотной загрузкой производственных мощностей заставляют предприятия всерьез задуматься о повышении конкурентоспособности.

2.2 Обзор используемых средств и аналогов

На фабрике ООО «ЮжУралКартон» на данный момент не используется специализированное ПО для планирования сменного задания. Инженер по планированию применяет офисные приложения, такие как MS EXCEL, программу 1С-предприятие. Однако функционал данного программного обеспечения не позволяет в полной мере выполнять поставленные задачи. Остается слишком много работы, которую необходимо

выполнять вручную, что отнимает много времени и сил. Вдобавок нельзя исключать человеческий фактор, нередко приводящий к ошибкам.

В свою очередь ошибки при планировании ведут к дополнительным трудовым и экономическим затратам, что негативно сказывается на экономике предприятия.

Фирма 1С предлагает решение данной проблемы, путем внедрения своего комплекса 1С-Предприятие ERP, однако ввиду того, производство гофротары и вспомогательных элементов упаковки – это весьма специфическая и узкая отрасль легкой промышленности, то соответствующих наработок и решений в этой области фирма 1С предложить не может. В то время как уже имеющиеся у них программные продукты только усложняют процесс планирования производства.

К сожалению, не смотря на всю технологичность и современность предприятия «ЮжУралКартон», Планирование производства остается слабым местом ввиду отсутствия специальных программ планирования производства и сменного задания

2.3 Экономическое обоснование

Опираясь на вышесказанное, становиться очевидно, что вопрос по разработке программы планирования производства стоит очень остро.

Технологические потери на предприятии можно рассчитать, опираясь на имеющиеся данные.

На предприятие поступает сырье. Вся партия поступившего сырья измеряется в тоннах, но выпуск продукции фабрики измеряется в квадратных метрах, встает вопрос о преобразовании одной величины в другую.

Вес одного квадратного метра изделия будет равен сумме граммажей всех слоев гофрокартона, но необходимо учитывать что в зависимости от профиля изготавливаемой продукции, гофрированный слой может составлять от 1,27 до 1,43 от плоского слоя (таблица2.1) .

.

Таблица 2.1 – Потребление флютинга (бумаги)

Профиль	Коэффициент гофрирования
E	1.27
B	1.37
C	1.43

Таким образом 1 квадратный метр изделия из трехслойного гофрокартона будет весить, г:

$$M = L1 + KF + L2 \quad (2.1)$$

где L1 – граммаж плоского лицевого слоя, г/м^2;

K – Коэфициент потребления флютинга;

F – граммаж среднего гофрируемого слоя, г/м^2;

L2 – граммаж плоского внутреннего слоя, г/м^2.

Такой способ эффективен в масштабе одного заказа, в то время как при расчете на все предприятие дает большие погрешности, обусловленные тем, что в конце цикла производства готовое изделие при одинаковой закладке задания (одинаковой комбинации сырья) может иметь разный вес. Комбинация определяет марку изделия, а не его вес, таким образом, например, 25 марку трехслойного гофрокартона можно получить как 125 граммажом, так и 112, если последний имеет более высокие показатели по ECT.

Решение было найдено. Процент брака на предприятии равен отношению массы использованного сырья, к массе упакованных макулатурных тюков, но технологические потери определить таким образом невозможно.

Технологические потери при производстве гофрокартона - это потери при производстве и (или) транспортировке товаров (работ, услуг), обусловленные технологическими особенностями производственного цикла или процесса транспортировки, а также физико-химическими характеристиками применяемого сырья.

Технологические потери при производстве гофрокартона бывают следующих видов:

- Потери, обусловленные снятием упаковочных слоев с рулонов сырья (срезы). Сырье на фабрику поставляется в бобинах, перед установкой на гофроагрегат верхние слои – они же упаковочные срезаются оператором, так как они имеют механические повреждения и не пригодны для производства гофрокартона.
- Потери обусловленные отсеканием кромки полотна (обрезь). При производстве гофрокартона кромка полотна всегда будет иметь несовмещение слоев и непроклей, это является недопустимым при

производстве гофрокороба, бракованная кромка всегда отсекается, за исключением специальных заказов, где обрезь не требуется. Ширина отсекаемой кромки варьируется от 30 до 200мм в зависимости от раскроя полотна.

- Потери при запуске линии или настройке машины. Каждый останов линии провоцирует последующую вырубку пересушенного полотна. Так же первые короба после перенастройки машины ПВА являются бракованными и называются настроечными.
- Потери связанные со сложной высечкой. Определенные типы коробов, требуют сложной высечки. Заказчику такая высечка необходима или просто удобна, в то время как на фабрике, отсеченные элементы удаляются при помощи аспирации в брак и считаются технологическими потерями.

Планирование задания способно повлиять на один вид технологических потерь – это потери вызванные неверным раскроем полотна. Уложиться в отведенные рамки недостаточно. Потери при раскрое необходимо минимизировать.

Формула расчета технологических потерь при раскрое полотна на гофроагрегате:

$$T_{\text{п}} = \frac{U_{\text{обр.}}}{U_{\text{фор.}}} * 100 \qquad (2.2)$$

где $U_{\text{обр}}$ – это ширина обрези, мм;

$U_{\text{фор}}$ – это ширина используемого формата, мм.

Потери выражаются в процентах и не должны превышать 5%. Это стандартный норматив для гофроагрегата, использующийся в промышленности.

Теперь, когда есть формула расчета доли технологических потерь и формула расчета массы квадратного метра готового изделия, можно рассчитать и их стоимость для одного заказа (формула 2.3). Для начала посчитаем массу потерь, затем умножим её на стоимость сырья.

$$M_{\text{тп}} = M * S_{\text{обр}} \qquad (2.3)$$

где М – это масса одного квадратного метра изделия, г;

$S_{обр}$ – это площадь всей полученной обрези, рассчитываемая по формуле 2.4.

$$S_{обр} = U_{обр} * L_{зак} \quad (2.4)$$

где $L_{зак}$ - это общая длина заказа, мм;

Стоимость одной тонны сырья условно возьмем 30 тыс. рублей, хотя этот показатель может варьироваться в зависимости от поставщика, марки и типа сырья, так, например стоимость белого картона примерно в полтора раза больше чем бурого.

Для примера возьмем одну строку сменного задания (рисунок 2.1) и проведем расчеты.

Рисунок 2.1 – Фрагмент сменного задания

Используемая комбинация сырья: Т-125 Ф-140 Т140, это значит, что согласно таблице замены сырья (рисунок 3.1 пункта 3) будет использовано следующее сырье:

L1=125 г/м^2 (Учалы)

F=140 г/м^2 (Учалы)

L2 =140 г/м^2 (Учалы)

Используемый профиль В означает что коэффициент потребления флютинга К равен 1.37, согласно таблице 2.1.

Тогда 1м^2 полотна по формуле 2.1 будет иметь массу:

$$M = L1 + KF + L2 = 125 + 1{,}37 * 140 + 140 = 456{,}8 \text{ г/м}^2$$

Длина одного заказа равна 3429 м переведем это значение в мм, тогда:

$L_{зак}$=3429*1000=3429000 мм

Ширина обрези 162 мм, ширина формата 2100, найдем коэффициент технологических потерь по формуле 2.2:

$$T_{\text{п}} = \frac{U_{\text{обр.}}}{U_{\text{фор.}}} * 100 = \frac{162}{2100} * 100 = 7{,}7\%$$

Вычислим площадь обрези по формуле 2.4:

$$S_{\text{обр}} = U_{\text{обр}} * L_{\text{зак}} = 162 * 3429000 = 55549800 \text{ мм}^2 = 555{,}498 \text{ м}^2$$

Зная площадь обрези, найдем массу технологических потерь по формуле 2.3:

$$M_{\text{тп}} = M * S_{\text{обр}} = 456{,}8 * 555{,}498 = 253751{,}4864 \text{ г} = 0{,}254 \text{ т}$$

Вычислим стоимость:

$$\text{М}_{\text{тп}} * 30000 = 0{,}254 * 30000 = 7620 \text{ рублей}$$

Таким образом стоимость технологических отходов при производстве на гофроагрегате будет равна 7620 рублей, при длине заказа 3429 м и ширине обрези 162 мм. Коэффициент тех. потерь равен 7,7% что выше нормы. При таком производстве фабрика терпит убытки.

Конечно, это пример расчета всего лишь одного заказа, для видения картины в целом необходимо посчитать все задание целиком. Сейчас коэффициент технологических потерь на гороагрегате равен 4,5%.

2.4 Постановка задачи

Теперь определимся с какими задачами должна справляться программа и какие функции должна выполнять:

- Уменьшение трудозатрат при планировании сменного задания. Работа планового отдела предприятия всегда сопряжена с активной деятельностью по нескольким направлениям: непосредственно планирование производства, составление производственных планов, составление расписания, выдача

заказов и контроль исполнения. Ввиду того, что первый пункт отнимает слишком много времени на проработку правильного раскроя полотна, принято решение заменить ручной труд на машинный, тем самым освободив время для решения других задач.

- Исключение человеческого фактора из планирования. Ручное выполнение работ зачастую связано с рисками допустить ошибку. В свою очередь ошибки при планировании могут принести значительный экономический ущерб предприятию
- Сведение к минимуму коэффициента технологических потерь за счет более тщательной проработки раскроя гофрополотна. Человек не способен адекватно расценить все возможные сочетания заказов, что необходимо для правильного раскроя полотна – нужно учесть множество факторов. В то время как, машина способна рассчитать все возможные комбинации и выбрать ту, которая свела бы технологические потери к минимуму.
- Оценка загруженности машин. Еще один немаловажный фактор планирования производства. Бывают ситуации, когда на продажу не требуется изделие, производимое той или иной машиной, тогда оборудование простаивает. В таких случаях начальник смены может перераспределить трудовые ресурсы, отправив операторов в помощь на более загруженную машину.

Вся информация, используемая на фабрике при планировании производства, должна быть перенесена в программу. У пользователя не должно возникнуть трудностей с интерпретацией интерфейса. Прежде всего, программа должна содержать базу данных комбинаций, ведь каждая комбинация – это основа будущего изделия, определяющая его характеристики.

Второй пункт по важности – это база данных самих изделий. Пользователь получит возможность выбирать из уже имеющихся в базе изделий необходимые, не вводя каждый раз заново их характеристики.

Инженер по планированию должен самостоятельно выбирать параметры раскроя полотна: вводить и изменять данные форматов сырья, максимальной и минимальной обрези.

Сформированное сменное задание должно быть максимально похоже на уже использующееся на фабрике, чтобы не вызывать трудностей в его прочтении у машинистов, операторов и начальников смен.

2.5 Выбор средств и технологий

Для данного проекта был выбран язык C++, как наиболее распространенный и гибкий язык программирования. Компилятор C++ Builder раскрывает все возможности этого языка.

C++Builder включает язык C++, компилятор, интегрированную среду разработки приложений IDE (Integrated Development Environment), отладчик и различные инструменты. C++Builder содержит комплект общих элементов управления, доступ к Windows API, библиотеку визуальных компонентов VCL (Visual Component Library), компоненты и инструменты для работы с базами данных.

C++Builder добавляет к процессу программирования на языке C++ возможность быстрой визуальной разработки интерфейса приложений.

Кроме библиотек OWL (Object Windows Library) и MFC (Microsoft Foundation Classes), он использует библиотеку VCL и позволяет включить в форму диалоги с пользователем, оставляя разработчику для реализации только функциональную часть, воплощающую алгоритм решения задачи.

C++Builder имеет общую с Delphi библиотеку классов, часть из которых осталась написанной на языке Object Pascal. Благодаря этому, а также включению в C++Builder компиляторов C++ и Object Pascal, в приложениях можно использовать компоненты и код, написанные на Object Pascal, а также формы и модули Delphi.

Создание пользовательского интерфейса приложения заключается в добавлении в окно формы объектов, называемых компонентами. C++Builder позволяет разработчику создавать собственные компоненты и настраивать Палитру компонентов.

2.6 Требования к интерфейсу Windows-приложений

При разработке программы немало внимания нужно уделить и интерфейсу пользователя. Под интерфейсом подразумевается тип экранного представления, при котором пользователь может вводить данные, запускать

команды и выполнять задачи, а так же визуально наблюдать за ходом выполнения программы.

C++ Builder предоставляет возможность разработчику приложения использовать широкую гамму инструментов для проектирования различных элементов программы: окон, кнопок, меню и т.д. Есть определенные принципы построения графического интерфейса пользователя, несоблюдение которых влечет за собой сложное, а, порой, и невозможное визуальное восприятие программы.

Для пользователя одним из главных преимуществ работы с Windows является то, что большинство программ устроены схожем образом. Поработав с несколькими интерфейсами программ, можно обнаружить их сходство. В результате поиск той или иной функции в программе становиться более простым и понятным.

Чаще всего, сложное приложение не может ограничиться одним окном. Поэтому, прежде всего, необходимо решить вопрос управления окнами. Есть две различные модели приложений: с интерфейсом одного документа (SDI) и с интерфейсом множества документов (MDI).

Интерфейс SDI удобнее для пользователя, большинство программ работает именно по этому принципу. Наличие одного окна не предполагает работу только с одним интерфейсом. При выборе нужной задачи приложение переводит пользователя на другую форму, при этом, не создавая бессмысленно большое количество окон. Так, например, работает проводник Windows. Однако бывают ситуации, когда пользователю необходимо работать сразу с двумя или более окнами программы. Тот же проводник способен создавать несколько окон, для удобства переноса или копирования данных.

Основным элементом любого приложения является форма — контейнер, в котором размещаются другие визуальные и не визуальные компоненты. С точки зрения пользователя форма — это окно, в котором он работает с приложением.

К внешнему виду окон в Windows предъявляются определенные требования. К счастью, C++ Builder автоматически обеспечивает стандартный для Windows вид окон любого приложения. Но перед началом работы необходимо продумать и указать, какие кнопки в полосе системного меню должны быть доступны в том или ином окне, должно ли окно допускать изменение пользователем его размеров, каким должен быть

заголовок окна. Все эти характеристики окон обеспечиваются установкой и управлением свойствами формы.

Цвет является мощным средством воздействия на психику человека. Именно поэтому обращаться с ним надо очень осторожно. Неудачное цветовое решение может приводить к быстрому утомлению пользователя, работающего с вашим приложением, к рассеиванию его внимания, к частым ошибкам. Слишком яркий или неподходящий цвет может отвлекать внимание пользователя или вводить его в заблуждение, создавать трудности в работе. А удачно подобранная гамма цветов, осмысленные цветовые акценты снижают утомляемость, сосредоточивают внимание пользователя на выполняемых в данный момент операциях, повышают эффективность работы. С помощью цвета вы можете на что-то намекнуть или привлечь внимание к определенным областям экрана. Цвет может также связываться с различными состояниями объектов.

Исходя из этого, везде, где это имеет смысл, следует использовать для своего приложения палитру системных цветов. Это те цвета, которые устанавливает пользователь при настройке Windows. Когда вы создаете новую форму или размещаете на ней компоненты, C++ Builder автоматически присваивает им цвета в соответствии со схемой цветов, установленной в Windows.

Использование шрифтов по умолчанию: System или MSSansSerif, чаще всего позволяет избежать неприятностей. Если в программе используется для надписей русские тексты, то при запуске приложения на компьютере с нерусифицированным Windows иногда возможны неприятности. Для предотвращения подобных случаев необходимо приложить файлы использованных шрифтов к программе.

Практически любое приложение должно иметь меню, поскольку именно меню дает наиболее удобный доступ к функциям программы. Существует несколько различных типов меню: главное меню с выпадающими списками разделов, каскадные меню, в которых разделу первичного меню ставится в соответствие список подразделов, и всплывающие или контекстные меню, появляющиеся, если пользователь щелкает правой кнопкой мыши на каком-то компоненте.

Основное требование к меню — их стандартизация. Это требование относится ко многим аспектам меню: месту размещения заголовков меню и их разделов, форме самих заголовков, клавишам быстрого

доступа, организации каскадных меню. Цель стандартизации — облегчить пользователю работу с приложением.

Группы функционально связанных разделов отделяются в выпадающих меню разделителями. Названия разделов меню должны быть привычными пользователю, краткими и понятными. Названия разделов должны начинаться с заглавной буквы.

Названия разделов меню, связанных с вызовом диалоговых окон, должны заканчиваться многоточием, показывающим пользователю, что при выборе этого раздела ему предстоит установить в диалоге еще какие либо параметры. Разделы, к которым относятся каскадные меню должны заканчиваться стрелкой, указывающей на наличие дочернего меню данного раздела.

В каждом названии раздела должен быть выделен подчеркиванием символ, соответствующий клавише быстрого доступа к разделу (клавиша Alt плюс подчеркнутый символ). Многим разделам могут быть поставлены в соответствие «горячие» клавиши, позволяющие обратиться к команде данного раздела, даже не заходя в меню. Комбинации таких «горячих» клавиш должны быть традиционными. Например, команды вырезания, копирования и вставки фрагментов текста практически всегда имеют «горячие» клавиши Ctrl-X, Ctrl-C и Ctrl-V соответственно. Заданные сочетания клавиш отображаются в заголовках соответствующих разделов.

Каждое окно, должно быть тщательно продумано и скомпоновано. Удачная компоновка может стимулировать эффективную работу пользователя, а неудачная — рассеивать внимание, отвлекать, заставлять тратить лишнее время на поиск нужной кнопки или индикатора.

Управляющие элементы и функционально связанные с ними компоненты экрана должны быть зрительно объединены в группы, заголовки которых коротко и четко поясняют их назначение. Такое объединение позволяют осуществлять различные панели. Можно рекомендовать, как правило, размещать компоненты не непосредственно на форме, а на панелях. Но и внутри панелей надо продумывать размещение компонентов, как с точки зрения эстетики, так и с точки зрения визуального отражения взаимоотношений элементов.

Каждое окно должно иметь некоторую центральную тему, которой подчиняется его композиция. Пользователь должен понимать, для чего предназначено данное окно и что в нем наиболее важно. При этом недопустимо перегружать окно большим числом органов управления, ввода

и отображения информации. В окне должно отображаться главное, а все детали и дополнительную информацию можно отнести на вспомогательные окна. Недопустимо, чтобы сходные по функциям органы управления в разных окнах назывались по-разному или размещались в разных местах окон. Все это мешает работе с приложением, отвлекает пользователя, заставляет его думать не о сущности работы, а о том, как приспособиться к тому или иному окну.

При проектировании приложения важно правильно определить последовательность табуляции оконных компонентов. Под этим понимается последовательность, в которой переключается фокус с компонента на компонент, когда пользователь нажимает клавишу табуляции Tab. Приложение должно предельно облегчать работу пользователя, снабжая его системой подсказок, помогающих сориентироваться в приложении.

При работе программы могут возникать различного рода ошибки: переполнение, деление на нуль, попытка открыть несуществующий файл и т.п. При возникновении таких исключительных ситуаций, программа генерирует так называемое исключение, а выполнение дальнейших вычислений в данном блоке прекращается.

Ошибок так же помогает избежать контроль ввода данных, когда в определенные поля пользователь может вносить только определенные данные, например, только цифры. Сокрытие не нужных в данных момент элементов программы, мешающих её работе, тоже помогает избежать ошибок программы.

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА

3.1 Создание баз комбинаций и готовых изделий

Работа программы невозможна без проработанных должным образом баз данных комбинаций и готовых изделий. За основу возьмем уже использующуюся на производстве таблицу замену сырья (рисунок. 3.2) и таблицу комбинаций (рисунок 3.3).

На каждое изделие существует технологическая карта (рисунок 3.4). Основную информацию об изделии занесем базу данных изделий.

Для хранения информации в табличной форме в программе используем форму TSStringGrid, позволяющую хранить текстовую информацию в формате AnsiString.

База данных комбинаций. Во второй форме программы создаем элемент StringGrid1, состоящий из 13 столбцов и двух строк. Имена столбцов по порядку: №, Марка, ЕСТ, L, F, L, F, L, Профиль, Лицевой слой, Внутренний слой, Уникальный код, Тип. Поля таблицы: Марка, Лицевой слой, Внутренний слой свяжем с соответствующими таблицами StringGrid2, StringGrid3, StringGrid4, содержащими поля ID и тип.

Используемый на фабрике код комбинации слишком «размыт» и не подходит для машинной обработки, таким образом, введем понятие «Уникальный код комбинации» содержащий в себе информацию о типе внутреннего слоя, лицевого слоя, профиле, марке и дополнительный код ЕСТ. Расшифровка кода представлена на рисунке 3.5. Структура её показана на рисунке 3.1.

№	Мар	ЕСТ	L	F	L	F	L	Проф	ЛС	ВС	УК	Тип
1	22	(2)	Ф-112	Ф-112	Ф-112			В	2	3	Т223(2)В	Т
…	…	…	…	…	…	…	…	…	…	…	…	…
n												

марка
1
…
35

id	тип
1	Целлюлозный
2	Макулатурный
3	Белый

id	тип
1	Влагостойкий
2	Целлюлозный
3	Макулатурный

Рисунок 3.1 – Структура базы данных комбинаций

ФАБРИКА ЮЖУРАЛКАРТОН

ТАБЛИЦА ЗАМЕН СЫРЬЯ от 20.03.2018

Лицевой плоский слой			
Обозначение	Рекомендованное сырье		
Ф 112	1	Б-100	Учалы
	2	Б-100	Капитал
		Б-100	Уссурийск
	3	Б-112	Учалы
	4	Б-125	Туймазы
Т 125	1	Т-125	Учалы
	2	Т-120	Пермь
Т 140	1	Т-140	Учалы
	2	Т-135	Пермь
Т-150	1	Т-150	Учалы
	2	Т-150	Пермь
Т-160	1	Т-160	Учалы
Т 175	1	Т-175	Учалы
	2	Т-175	Пермь
Т 200	1	Т-200	Пермь
К105Б	1	К-105бел	Монди
	2	К-110бел	Илим
К 115Б	1	К-110бел	Илим
	2	К-115бел	Монди
	3	К-105бел	Монди
К-140	1	К-140	Илим
	2	К-140	Выборг
	3	К-150	Селенга
К-125	1	К-125	Илим

Слой для гофрирования			
Обозначение	Рекомендованное сырье		
Ф 100	1	Б-100,110	Каменка
	2	Б-100	Учалы
		Б-100	Уссурийск
	4	Б-112	Капитал
Ф 112	1	Б-112	Капитал
	2	Б-112	Учалы
	3	Б-100	Уссурийск
	4	Б-125	Туймазы
Ф 112 пермь	1	Б-0-112	Пермь
Ф 125	1	Б-125	Учалы
	2	Б-125	Капитал
	3	Б-112	Марий
	4	Б-112	Уссурийск
Ф 140	1	Б-140	Учалы
	2	Б-125	Марий
	3	Б-125	Уссурийск
Ф 160	1	Б-160	Учалы
	2	Б-140	Марий
	3	Б-140	Уссурийск
		Т-150	Пермь
Ф 175	1	Б-175	Учалы
	2	Т-175	Пермь
Ф 160пермь	1	Б-0-160	Пермь
	3	Б-0-170	Пермь
Ф 200	1	Т-200	Пермь

Внутренний плоский слой (гофропресс)			
Обозначение	Рекомендованное сырье		
Ф 100	1	Б-100,110	Каменка
	2	Б-100	Учалы
		Б-100	Уссурийск
	3	Б-112	Капитал
	4	Б-112	Учалы
Ф 112	1	Б-112	Капитал
	2	Б-112	Учалы
	3	Б-112	Уссурийск
	4	Б-125	Туймазы
Т 125	1	Т-120	Пермь
	2	Т-125	Учалы
Т 140	1	Т-135	Пермь
	2	Т-140	Учалы
Т 150	1	Т-150	Учалы
	2	Т-150	Пермь
Т-160	1	Т-160	Учалы
	2	Т-150	Пермь
Т 175	1	Т-175	Учалы
	2	Т-175	Пермь
	3	Т-160	Уссурийск
Т 200	1	Т-200	Пермь
К 175	1	К-175	Илим
	2	К-175	Выборг
	3	К-170	Илим
	4	К-165	Селенга
Т 125 V	1	Т-120V	Учалы
	2	Т-125V	Учалы
Т135 V	1	Т-135V	Учалы

Рисунок 3.2 – Таблица замены сырья

ФАБРИКА
ЮЖУРАЛКАРТОН
456550, Челябинская обл., г. Коркино, ул. 30 лет ВЛКСМ, 189 а
тел.: (35152) 3-03-48, факс: (35152) 3-03-49
www.uralkarton.ru

УТВЕРЖДАЮ:
Директор по производству
ООО «Фабрика ЮжУралКартон»
В.Н. Кушнерев

Типовые комбинации сырья от 08.09.17г.

Марка	Показатель ECT, кН/м	Код комбинации	Основная комбинация
Бурый макулатурный покровный слой (трехслойный гофрокартон)			
Т-21 «В», «Е»	2,2кН/м	12B,E	Ф-112 Ф-100 Ф-100
Т-22 «В», «С», «Е»	3,0кН/м	22 B,E,C	Ф-112 Ф-112 Ф-112
Т-23(1)	3,8кН/м	32(1)B,E	Т-125 Ф-112 Ф-125
Т-23(1)	3,8кН/м	32(1)C	Т-125 Ф-112 Т-125
Т-23(2)	4,1кН/м	32(2)B,C,E	Т-125 Ф-125 Т-125
Т-23(3)	4,4кН/м	32(3)B,C,E	Т-125 Ф-125 Т-140
Т-24(1)	4,6кН/м	42(1)B,C,E	Т-125 Ф-140 Т-125
Т-24(2)	4,9кН/м	42(2)B,C,E	Т-125 Ф-160 Т-125
Т-24(3)	5,2кН/м	42(3)B,C,E	Т-125 Ф-160 Т-140
Т-25(1)	5,4кН/м	52(1)B,C,	Т-160 Ф-140 Т-160
Т-25(1)	5,4кН/м	52(1)E	Т-160 Ф-125 Т-160
Т-25(2)	5,7кН/м	52(2)B,C,E	Т-140 Ф-175 Т-140
Т-25(3)	6,0кН/м	52(3)C	Т-175Пермь Б0-125ПермьТ-175Пермь
Т-25(3)	6,0кН/м	52(3)B,E	Т-140 Ф-175 Т-160
Т-26(1)	6,2кН/м	62(1)E	Т-200Пермь Б0-125Пермь Т-200Пермь
Т-26(1)	6,2кН/м	62(1)B,C	Т-140 Ф-175 Т-175
Т-26(2)	6,5кН/м	62(2)B,C,E	Т-175 Ф-175 Т-175
Т-26(3)	6,8кН/м	62(3)B,C,E	Т-175 Ф-200 Т-200
Т-27(1)	7,0кН/м	72(1)C	Т-200Пермь Б0-170Пермь Т-200Пермь
Т-27(1)	7,0кН/м	72(1)B,,E	Т-200 Ф-200 Т-200
Бурый целлюлозный покровный и (или) внутренний слой (трехслойный гофрокартон)			
Т-23(1)	3,8кН/м	34(1)B,C,E	К-115 Ф-125 К-115
Т-24(1)	4,6кН/м	43(1)B,C,E	Т-125 Ф-140 К-125
Т-24(1)	4,6кН/м	44(1)B,C,E	К-125 Ф-140 К-125
Т-25(1)	5,4кН/м	50(1)B,C,E	К-140 Ф-160 Т-140
Т-25(1)	5,4кН/м	53(1)B,C	Т-140 Ф-160 К-140
Т-25(1)	5,4кН/м	54(1)B,C	К-140 Ф-160 К-140
Т-26(1)	6,2кН/м	63(1)B,C,E	Т-140 Ф-175 К-175
Т-26(1)	6,2кН/м	60(1) B,C	К-140 Ф-175 Т-175
Т-26(1)	6,2кН/м	64(1)B,C	К-150 Ф-160 К-175
Т-26(1)	6,2кН/м	64(1)E	К-175 Ф-140 К-175
Т-27(1)	7,0кН/м	70(1)B,C,E	К-175 Ф-200 Т-200
Т-27(1)	7,0кН/м	73(1)B,C,E	Т-200 Ф-200 К-175
Т-27(1)	7,0кН/м	74(1)B,C,E	К-175 Ф-200 К-175
Бурый покровный слой (трехслойный влагостойкий гофрокартон)			
Т-23(2)	4,1кН/м	33(2)BV,CV	Т-125 Ф-125 Т-125 V
Т-23(3)	4,4кН/м	33(3)BV,CV	Т-140 Ф-125 Т-125 V
Т-24(1)	4,6кН/м	43(1)BV,CV	Т-125 Ф-140 Т-125V
Т-24(3)	5,2кН/м	43(3)BV,CV	Т-125 Ф-160 Т-135V
Т-25(2)	5,7кН/м	53(2)BV,CV	Т-140 Ф-175 Т-125V

Рисунок 3.3 - Таблица комбинаций

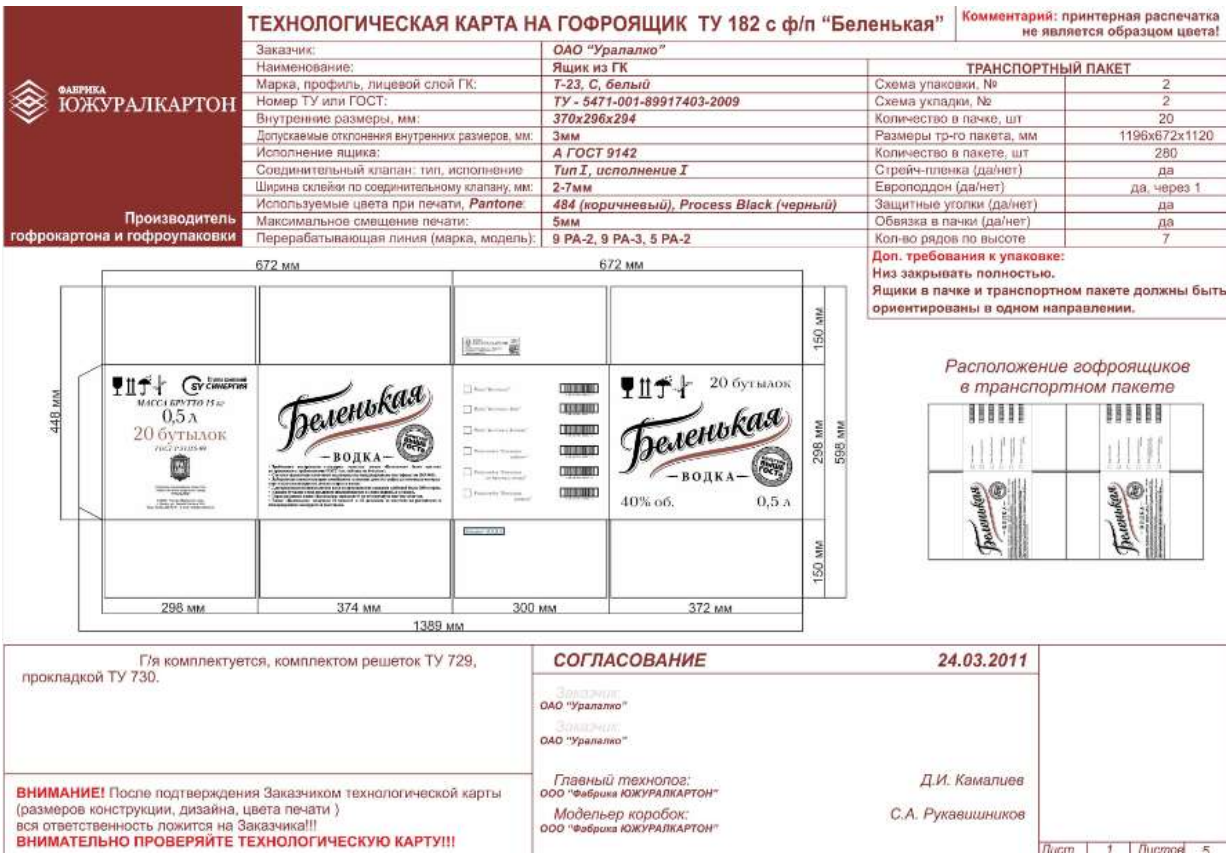

Рисунок 3.4 – Технологическая карта на изделие

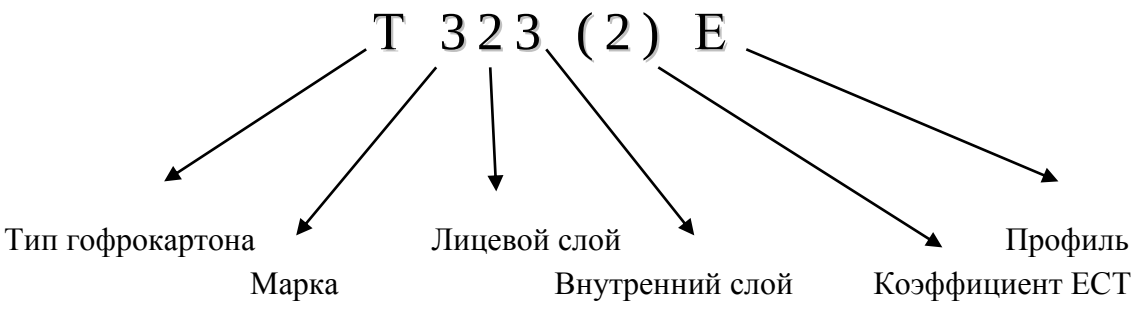

Рисунок 3.5 – Расшифровка кода комбинации

Уникальный код формируется автоматически, после ввода всех необходимых данных. Первичный ключ кода – это номер комбинации, он остается неизменным и заносится в базу только один раз. Внешние ключи: ЛС и ВС ссылаются на соответствующие записи соответствующих таблиц, и так же остаются неизменны. В поле «Марка» записывается значение, соответствующее марке этой комбинации, при этом 1-27 – это марки трехслойного гофрокартона, а 31-35 – пятислойного. Записи полей L ,F и L соответствуют граммажам и типам используемого сырья.

База данных готовых изделий. В первой форме программы с помощью компонента SringGrid1 создаем базу данных готовых изделий. Она содержит информацию о наименовании, длине, ширине, расположении биговок, марке, площади изделия в квадратных метрах, профиле, перерабатывающей линии, уникальном коде и id комбинации, а так же содержит собственный порядковый номер. Структура базы данных показана на рисунке 3.6

№	Наим.	Дл.	Шир.	Биг.	Мар.	S изд.	Проф.	Линия	УК	Id кач.
1	ТУ-1	1420	1050		23	1,491	В	IS	Т232(2)В	10
…	…	…	…	…	…	…	…	…	…	…
n										

№	Мар	ЕСТ	L	F	L	F	L	Проф	ЛС	ВС	УК	Тип
1	22	(2)	Ф-112	Ф-112	Ф-112			В	2	3	Т223(2)В	Т
…	…	…	…	…	…	…	…	…	…	…	…	…
n												

Рисунок 3.6 – Структура базы данных готовых изделий

3.2 Создание расчетной части программы

Самой главной частью программы является её расчетная часть. От правильности раскроя зависит показатель технологических потерь, чем он меньше – тем эффективнее производство.

В предварительную таблицу заносятся данные заказа: количество и наименование изделие. Таблица сама формирует запросы из базы данных и заносит соответствующие ключи в поля: id и уникальный код. Таким образом программа уже знает об изделии всю необходимую информацию.

После заполнения предварительной таблицы, пользователь нажимает кнопку старт. Вступает в работу алгоритм расчета.

Первым делом программа осуществляет поиск по таблице заготовок с нулевой обрезью. Так как коэффициент технологических потерь таких заготовок равен нулю, то их сразу можно раскроить под подходящий формат и вывести из предварительной таблицы.

Следующим шагом программа осуществляет поиск заготовок, одинаковых по уникальному коду. Так как такие заготовки будут выпускаться из одной сырьевой комбинации, то их можно сочетать между

собой, для получения минимально возможных технологических потерь. Если таковые имеются, то программа проверяет все возможные сочетания и, подходящие под параметры заданные пользователем, она заносит в таблицу предварительного сменного задания. Изделия, не удовлетворяющие заданным параметрам, записываются в таблицу «Остатки».

Если уникальный код комбинации встречается в предварительной таблице только один раз, то программа так же просчитывает все возможные варианты раскроя, но уже без сочетания с другими заготовками, опираясь, только лишь на ширину изделия и имеющиеся форматы сырья. Подходящий параметрам пользователя вариант раскроя заносится в таблицу предварительного задания.

Не редки ситуации, когда длина одного заказа в погонных метрах не соответствует длине второго, но они прекрасно раскраиваются по ширине, тогда программа формирует строку, соответствующую длине наименьшего заказа, а остаток от наиболее длинного заказа она возвращает в таблицу, для дальнейшей обработки.

Алгоритм работает до тех пор, пока в таблице требуемых изделий есть хотя бы одна заполненная строка. После завершения расчетов выводиться сообщение пользователю «The end».

Таблица «Остатки» сканируется на предмет наличия идентичных изделий, имеющихся в предварительном задании. Зеленым цветом выделяются строки, уже присутствующие в предварительном задании, но не вошедшие в него полностью. Программа разрешает, по желанию пользователя, перенести эти строки в задание с последующим пересчетом количества сопутствующей заготовки, для того, чтобы заказ был выполнен полностью. Красным цветом выделяются те строки, которые не вошли в сменное задание и не могут быть раскроены, так как параметры этих изделий не позволят сделать раскрой полотна, по настройкам, заданным оператором. В таком случае сам пользователь уже решает что делать, либо изменить настройки раскроя полотна, либо, воспользовавшись поиском в базе данных, по определенным параметрам найти изделие, сочетаемое с требуемым. Так же всегда актуален прикрой дополнительных средств упаковки, используемых для собственных нужд фабрики, если это, конечно, экономически целесообразно.

Блок-схема алгоритма расчета показана на рисунках 3.7, 3.8,3.9.

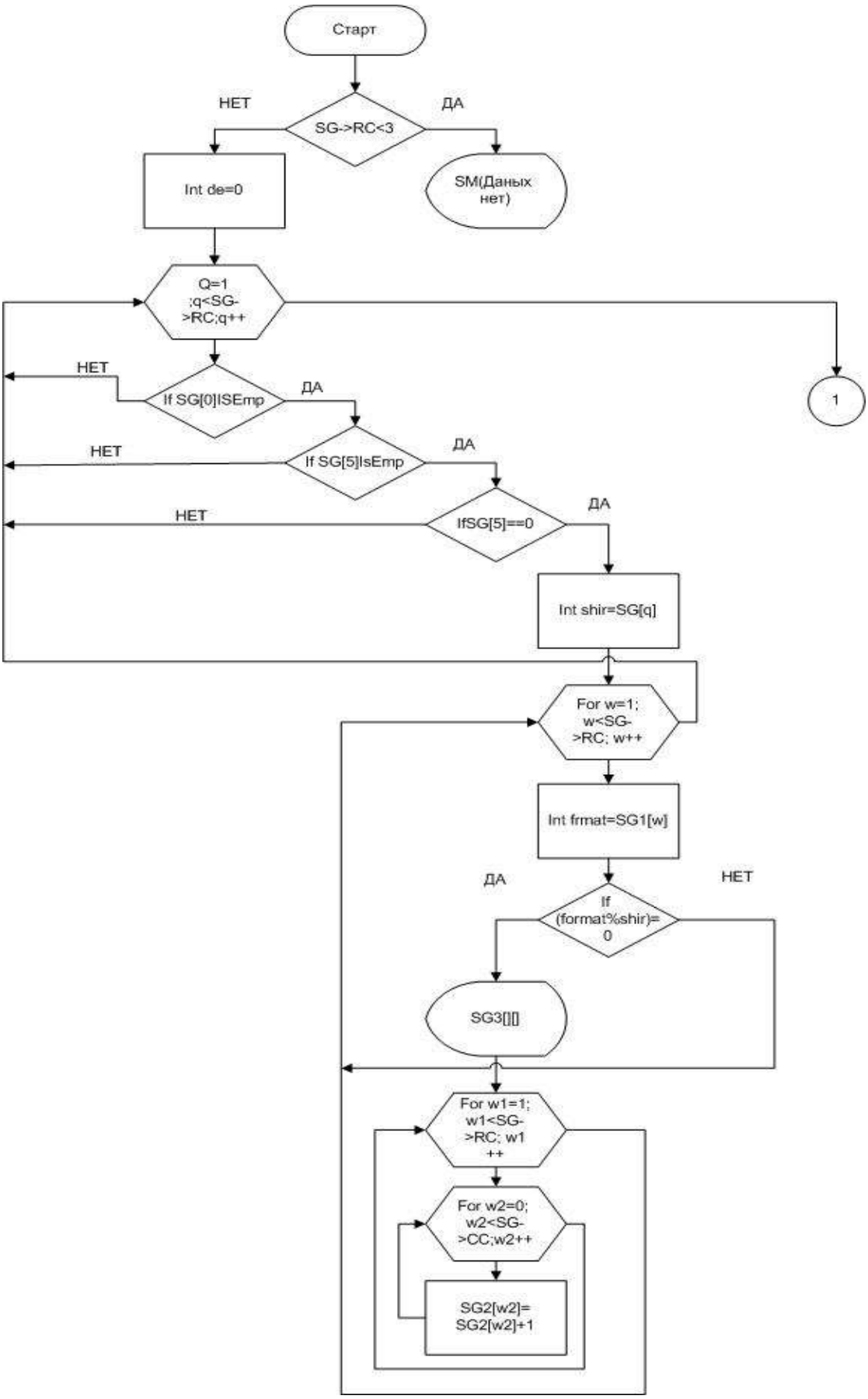

Рисунок 3.7 – Блок схема алгоритма расчетной части программы часть 1

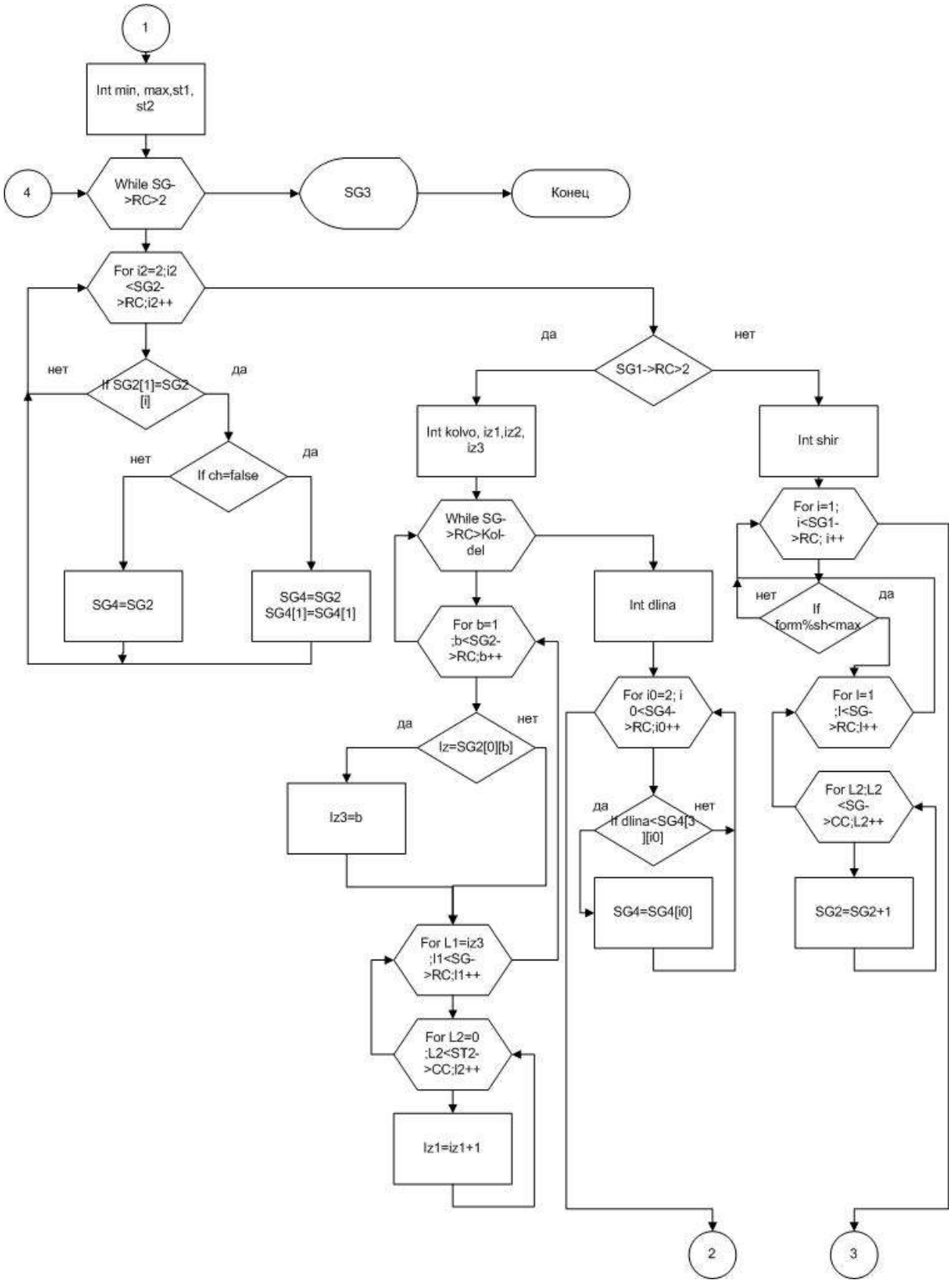

Рисунок 3.8 – Блок схема алгоритма расчетной части программы часть 2

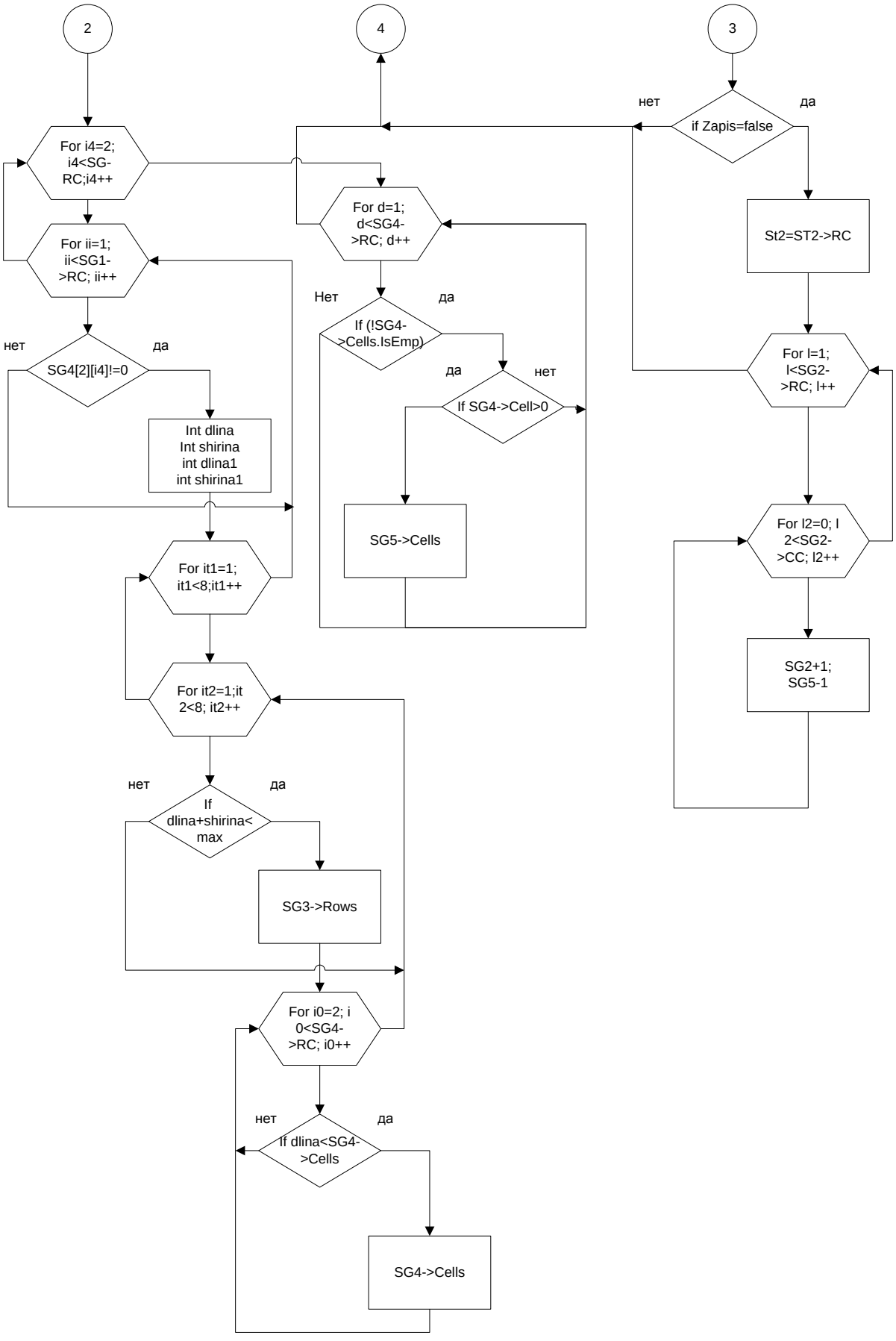

Рисунок 3.9 – Блок схема алгоритма расчетной части программы часть 3

3.3 Разработка статистической части программы и формирование задания

Кроме выполнения операций раскроя полотна, программа выполняет еще одну немаловажную функцию – статистику загруженности машин.

После заполнения всех полей, после формирования производственного задания, программа оценивает количество заготовки, предназначенное для каждой машины. Принято, что, производительность гофроагрегата измеряется в квадратных метрах, а производительность линий ПВА в штуках выпущенных изделий.

Пользователь может самостоятельно менять значения нормы выработки в смену на свое усмотрение, однако по умолчанию установлены следующие нормативы:

- Гофроагрегат Fosber – 150000 м^2
- 9РА-2 – 70000 шт
- 9РА-3 – 90000 шт
- 5РА-2 – 50000 шт
- ISHIKAVA – 50000 шт
- ASAHI – 20000 шт
- WARD – 40000 шт
- Vega – 15000 шт

Исходя из этих данных программа рассчитывает загруженность машин в смену в процентах.

Для лучшей наглядности был использован инструмент ProgressBar , визуально передающий информацию о загруженности машин.

После того, как пользователь убедился в правильности составленного производственного задания, его необходимо сгруппировать по блокам. Блоком условно назовем элемент таблицы, объединяющий позиции задания с одинаковым уникальным кодом и форматом сырья. Проще говоря, все строки задания объединяются по принципу: все, что можно произвести из одного сырья, нужно выпускать сразу. Таким образом, уменьшается количество перенастроек и смен сырья на гофроагрегате.

С помощью инструмента PrintDialog задание можно вывести на печать.

4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

4.1 Планирование

В главном окне программы нажимаем кнопку «База» (рисунок 4.1)

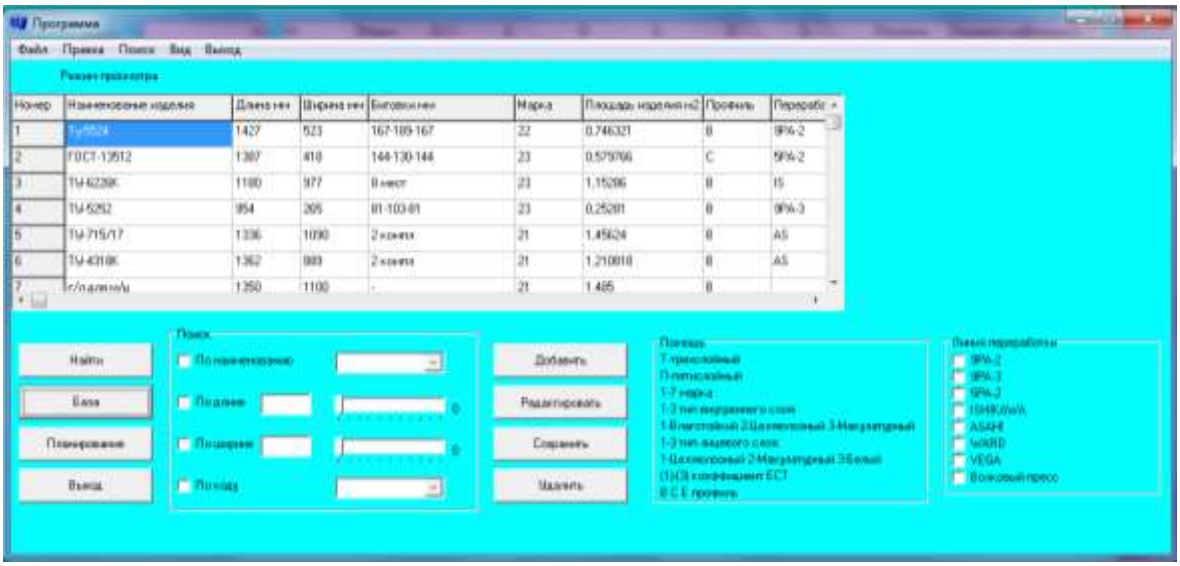

Рисунок 4.1 - Главное окно программы

В открывшееся окне видим поля ввода новых данных. База комбинаций заполняется пользователем, согласно уже используемой на производстве таблице комбинаций сырья, с одним лишь отличием, что в каждой ячейке возможно только одно значение.

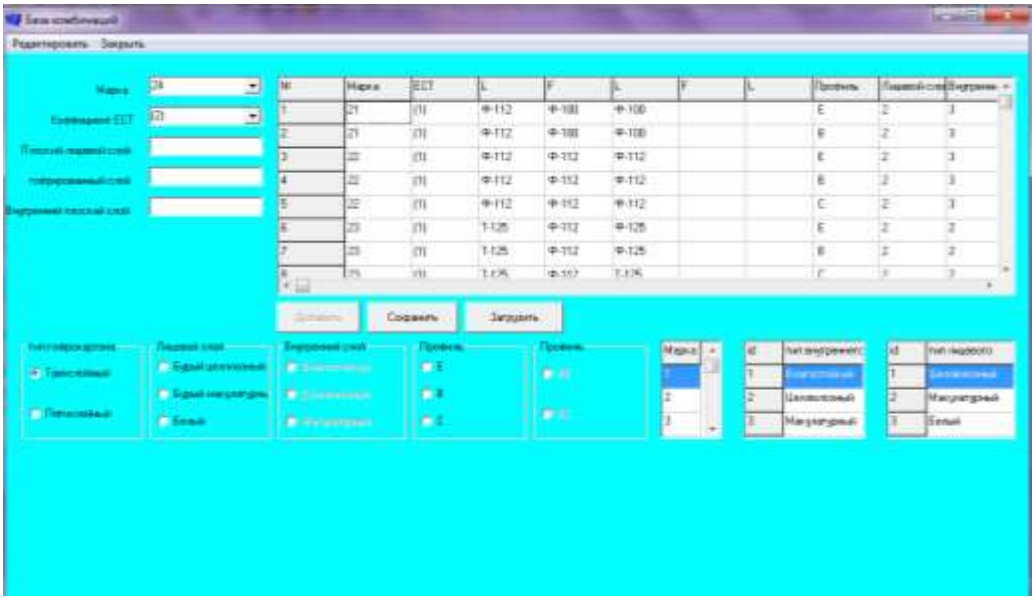

Рисунок 4.2 - База комбинаций

Теперь переходим на главное окно программы, здесь находится база данных изделий. Новое изделие добавляется в базу исходя из информации в технологической карте.

Рисунок 4.3 - Заполнение базы данных изделий

После внесения всех необходимых данных программа готова к работе.

Нажимаем кнопку «Планирование» на главной форме. Открывается окно планировщика (рисунок 4.4)

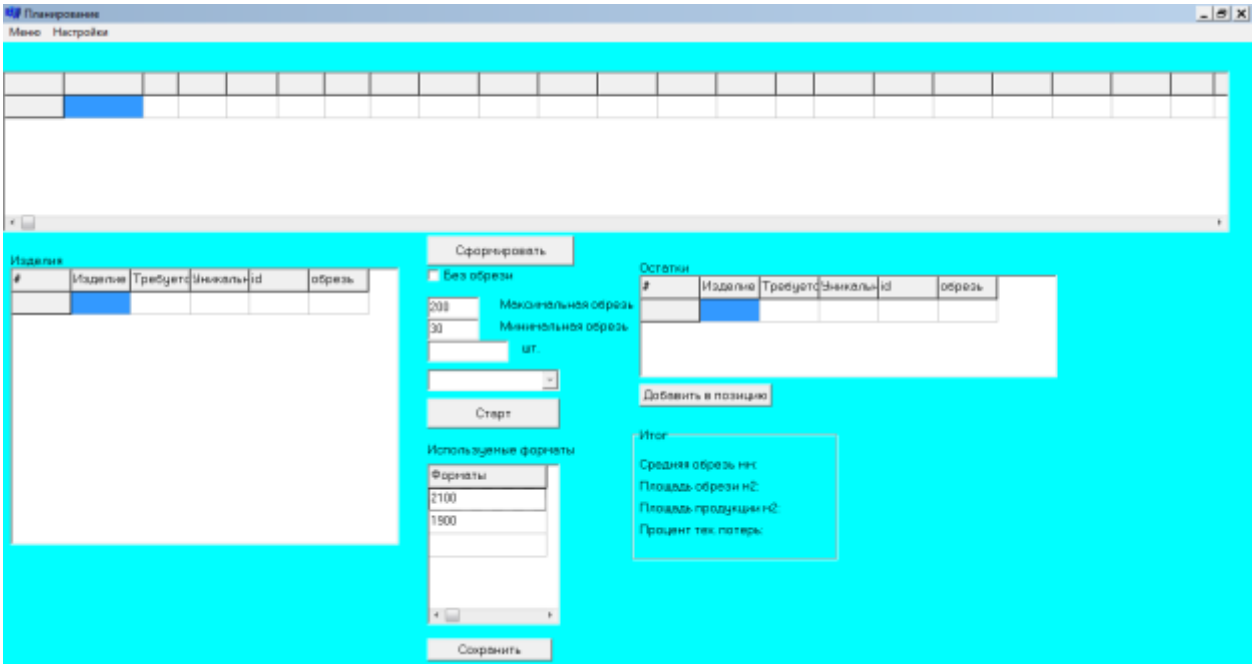

Рисунок 4.4 – Планировщик задания

Сохранение файлов по умолчанию осуществляется в папку: C:\Users\Public\Documents, так как этот каталог один из немногих, куда операционная система Windiws разрешает беспрепятственно сохранять данные. Этот путь изменению не подлежит во избежание сбоев в работе программы.

Программа для своей работы использует три таблицы, соответствующие двум базам данных, и таблице используемых форматов сырья. Планировщик работает с уже сохраненными файлами.

Алгоритм работы с планировщиком будет выглядеть следующим образом:

- Пользователь вводит данные о необходимой обрези, по умолчанию максимальная обрезь – 200 мм, минимальная – 30 мм. Эти значения рекомендованы производителем оборудования и не вызовут проблем при работе линии, однако в редких случаях возможно превышение лимитов, когда, например, сырья нужного формата нет на складе, именно поэтому программа дает возможность пользователю самому выбирать значения.

  Существуют так же заготовки, производимые вообще без обрези, тогда в программе необходимо это указать. Как правило, это заготовки на станки переработки сложной высечки ISHIKAVA и ASAHI. Данные машины способны производить обрезь самостоятельно.
- Пользователь вводит количество заготовок, которых необходимо произвести.
- По умолчанию программой используются 2 формата сырья шириной 1900 мм и 2100 мм, т.к. они являются основными, но существуют и другие форматы, такие как 1600мм, 1650мм, 1700мм, 2300мм, 2350мм и т.д. При необходимости их можно внести в список используемых программой. После нажатия кнопки сохранить, все введенные форматы будут загружаться при каждом запуске программы
- Дальше пользователь вводит название необходимого изделия, и если оно содержится в базе данных, то программа разрешает ввод и изделие, а так же необходимая информация о нем, заносится в таблицу предварительного планирования (Рисунок 4.5).

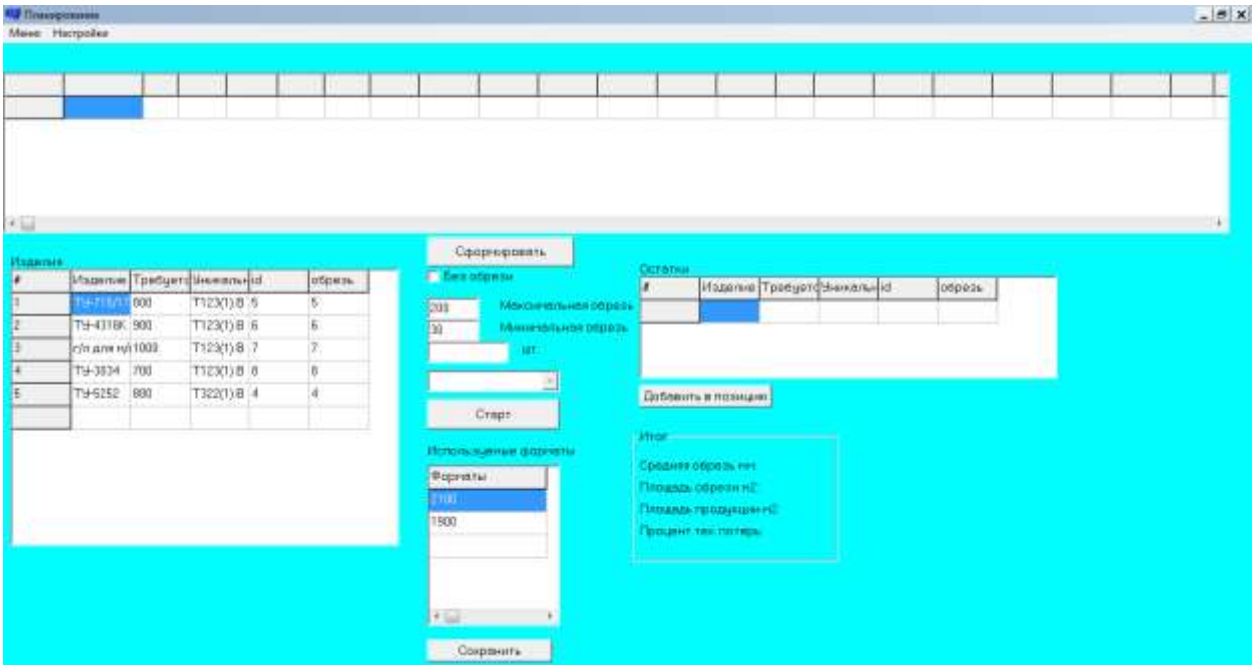

Рисунок 4.5 – Заполнение таблицы предварительного планирования

- После этого нажимается кнопка «Старт» и программа производит раскрой полотна. Формируется предварительное производственное задание (Рисунок 4.6)

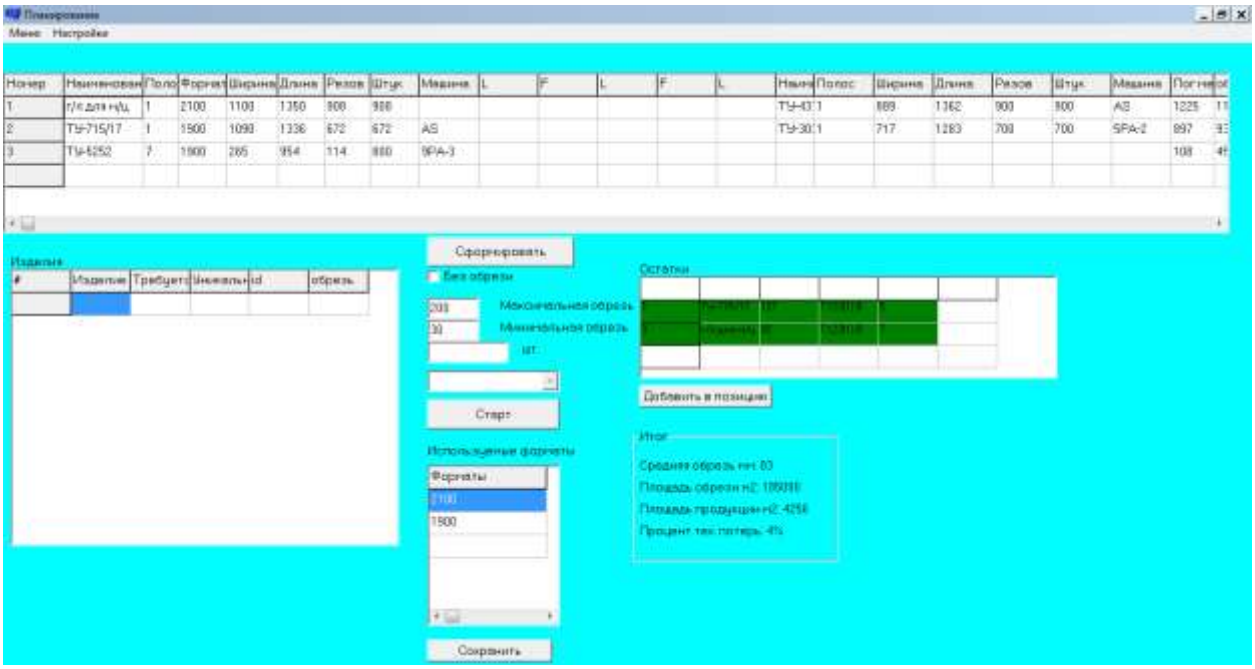

Рисунок 4.6 – Предварительное производственное задание

После формирования предварительного задания, программа выводит в таблицу остатки информацию, об изделиях, которые она не смогла распределить. Следует обратить внимание, что программа выделяет позиции зленным цветом, те, которые уже есть в сменном задании. Строку можно перенести в сменное задание. При этом увеличится длина скроенного с ней заказа.

Последним пунктом работы с программой будет формирование производственного задания. Нажав кнопку сформировать, пользователь получит готовое производственное задание (Рисунок 4.7)

Рисунок 4.7 – Производственное задание

4.2 Статистика

Программа позволяет вести статистику загруженности машин (рисунок 4.8) при выполнении производственного задания.

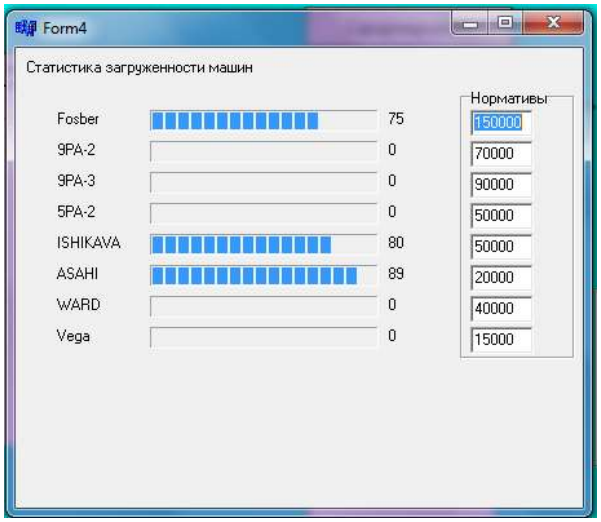

Рисунок 4.8 – Статистика загруженности машин

Статистика помогает заблаговременно получать информацию, анализировать её и принимать решения.

Форма статистика включает в себя поле с названиями всех станков предприятия, шкалу загруженности каждого станка, дополнительный индекс, отражающий загрузку машины в процентах, а так же поле ввода данных нормативов. Значение норматива – это среднегодовая выработка оборудования. Для каждого станка это значение вычисляется индивидуально по формуле (4.1)

$$Т_{ср} = \frac{Т_{год}}{ЭФВ} * 11, \quad (4.1)$$

где ЭФВ – это эффективный фонд рабочего времени в часах.

Работа оборудования невозможна без простоя. Простои бывают плановые и внеплановые. При расчете ЭФВ учитываются только плановые простои, к ним относятся:

- Перерывы на обед и технологические перерывы
- Планово-предупредительные ремонты (ППР)

Внутренним регламентом предприятия предусмотрено, что рабочая смена равна 11 часов, с учетом обеда и технологических перерывов. На ППР отводится 8 часов в неделю. Фабрика работает в двухсменном режиме (день/ночь).Таким образом ЭФВ Предприятия будет равно :

$$ЭФВ = 365 * 2 * 11 - 8 * 52 = 8300 - 416 = 7884 \text{ ч}$$

Так, например, среднегодовая выработка гофроагрегата будет равна:

$$Т_{ср} = \frac{Т_{год}}{ЭФВ} * 11 = \frac{90000000}{7884} * 11 = 125570 \text{ м}^2/\text{см}$$

Так как каждое предприятие стремится к росту производительности, то этот показатель нужно умножить на коэффициент желаемого роста $К_р$. На предприятии «ЮжУралКартон» он принят 1,2. Умножив $Т_{ср}$ на 1,2 получим норматив выработки станка за смену.

$$Н_{г/а} = Т_{ср.г/а} * 1{,}2 \approx 150000 \text{ м}^2/\text{см}$$

Таким образом, отдел планирования получает информацию об отклонениях от плана развития предприятия.

В процессе развития предприятия нормативы будут изменяться, программа позволяет пользователю вводить значения вручную.

Еще одной особенностью производства в пользу ведения статистики, является то, что ПВА 9РА-2, 9РА-3, 5РА-2 имеют схожую конструкцию и в некоторых случаях взаимозаменяемы. Так же бригаду рабочих не загруженных машин возможно привлечь к другим работам, либо вообще отпустить со смены.

Статистика позволяет оценивать работу предприятия в целом. Если производственная мощность предприятия используется недостаточно полно, это приводит к увеличению доли постоянных издержек, росту себестоимости, снижению прибыльности. Поэтому в процессе анализа необходимо установить, какие изменения произошли в производственной мощности предприятия, насколько полно она используется и как это влияет на себестоимость, прибыль, безубыточность и другие показатели.

Производственная мощность составляет материальную основу плана выпуска продукции, поэтому обоснование производственной программы расчетами производственной мощности является основным звеном производственного планирования. Для производственного планирования используют и расчет производственной мощности оборудования станочного типа исходя из эффективного фонда времени станка каждого типа.

4.3 Экономический эффект

На данное время средний показатель технологических отходов на гофроагрегате составляет 4,5%, согласно статистике планового отдела, что не превышает норму, но ведет к увеличению издержек.

Предприятие терпит убытки, ситуацию необходимо менять. На отделе планирования производства лежит большая ответственность. Монотонный ручной труд, сложные расчеты раскроя полотна снижают эффективность планирования. В век цифровых технологий необходимо всю сложную работу перекладывать на машины. Разработанная программа уже способна взять на себя часть задач инженера по планированию. Она может отображать статистику, указывая на слабые места производства в целом, способна раскраивать гофрополотно и формировать сменные задания. В сущности, программа значительно упростит работу планового отдела. Однако,

необходимо проверить на сколько снизится коэффициент технологических потерь на гофроагрегате, ведь это было одной из задач, возлагаемых на приложение.

Для формирования нового сменного задания внесем в программу наиболее часто выпускаемые фабрикой изделия (рисунок 4.9).

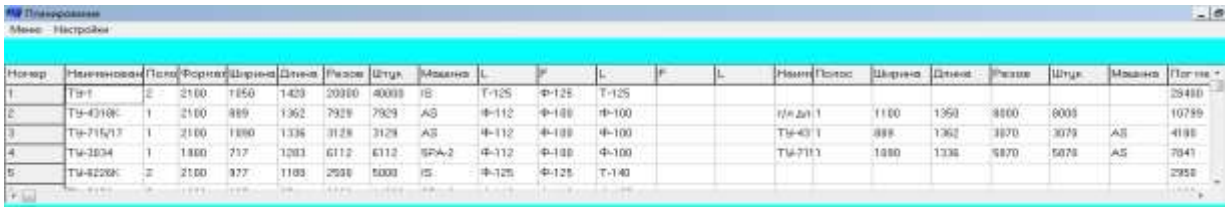

Рисунок 4.9 – Формирование предварительного сменного задания

Показатели, рассчитанные программой, будут выглядеть:

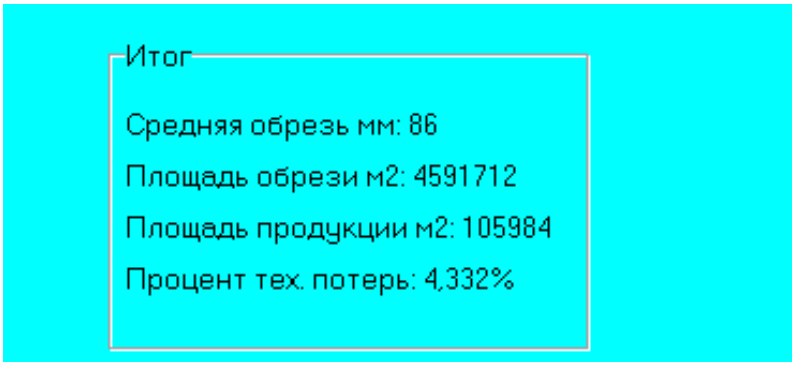

Рисунок 4.10 – Отчет программы о сменном задании

Из отчета(рисунок 4.10) видно, что коэффициент технологических потерь составляет 4,3%, что на 0,2% ниже, чем средний показатель по фабрике.

Среднемесячная норма выработки гофроагрегата составляет 7,5 млн м^2 продукции . При коэффициенте в 4,3% технологические потери (в м^2) в месяц составят:

$$S_{1обр} = 7500000/95{,}7 * 4{,}3 = 336990 \text{ м}^2$$

К сожалению, точные расчеты произвести не удастся, об этом уже упоминалось в пункте 2.3.

Исходя из номенклатуры сырья (Рисунок 3.1), используемого на фабрике, можно высчитать средний граммаж квадратного метра изделия, при условии, что расход сырья разных категорий будет одинаков.

Для плоского лицевого слоя он будет равен:

$L1_{ср}$ =(112+125+140+150+175+200+115+140+125)/9=1282/9=142 г/м^2

Для среднего гофрированного слоя:

$F_{ср}$=(100+112+125+140+150+175+160+200)/8=1162/8=145 г/м^2

Для внутреннего плоского слоя:

$L2_{ср}$=(100+112+125+150+175+200+125+135)/8=1122/8=140 г/м^2

Тогда средний граммаж изделия будет равен:

М=142+145+140=427 г/м^2

Масса технологических потерь в месяц:

$М_{тп}$=427*336990*0,000001=143 т

Стоимость одной тонны сырья в среднем равна 30000 рублей. Тогда убытки будут равны:

30000*143=4290000 рублей

Теперь посчитаем для коэффициента в 4,5%:

$$S_{2обр} = 7500000/9{,}5 * 4{,}5 = 353403 \text{ м}^2$$

Масса технологических потерь в месяц:

$М_{тп}$=427*353403*0,000001=151 т

Стоимость одной тонны сырья в среднем равна 30000 рублей. Тогда убытки будут равны:

30000*151=4527094 рублей

Таким образом, уменьшение издержек предприятия в месяц составит:

4527094-4290000=237094 рубля

Хоть эта цифра и невелика в масштабах финансов предприятия, все же она доказывает эффективность работы программы.

ЗАКЛЮЧЕНИЕ

В данном дипломном проекте были рассмотрены основные проблемы при планировании производственного задания фабрики ООО «ЮжУралКартон». Изучены технологические процессы работы предприятия, а так же характеристики выпускаемой продукции.

Произведены расчеты стоимости технологических потерь на гофроагрегате. Приведены формулы расчета среднемесячной производительности оборудования.

На основании полученных данных была разработана программа планирования производственного задания. В ней учтены все аспекты работы предприятия. Программа позволяет улучшить работу планового отдела, переложить задачи сложных расчетов с человека на компьютер. Она выдает информацию о простоях оборудования, и отображает статистику загруженности машин, а самое главное – производит раскрой полотна, опираясь на минимизацию технологических потерь.

Экономический эффект от внедрения программы состоит в уменьшении количества технологических потерь на 0,2% или экономии предприятием 237094 рублей в месяц.

Программа отвечает всем требованиям, предъявляемым к Windows-приложениям. У пользователя не возникнет проблем с интерпретацией интерфейса. Работа программы проста и понятна, а защита от ввода некорректных данных позволяет избежать ошибок.

Задачи, поставленные в проекте, успешно реализованы. Правильное планирование производства оптимизирует работу всей фабрики, снижает издержки предприятия, позволяет смотреть вперед и увеличивать темпы выпуска продукции.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Архангельский, А.Я. C++Builder: Справочное пособие. Книга 2. Классы и компоненты./ А.Я Архангельский. – М.: БИНОМ-Пресс, 2002. – 528 с.

2 Архангельский, А.Я. Программирование в C++Builder5 (или 6)./ А.Я Архангельский. – М.: ЗАО «Издательство БИНОМ», 2002. – 1152 с.

3 Гладкий, А. М. Экономический анализ: Учебник для вузов / А. Гладкий - http://1-fin.ru/?id=134, 2012. – 77 с.

4 Дейтел, Х.М. Как программировать на Си++./ Х.М. Дейтел, П.Дж.Дейтел – М.: ЗАО БИНОМ,1999. – 1000 с.

5 Казанцев, Р. В. Расчет производственной мощности промышленного предприятия / Р. В Казанцев, // «Справочник экономиста» №3 - https://www.profiz.ru/se/3_2016/prom_moschnost/, 2016. – 70 с.

6 Касаткин, А.И. Профессиональное программирование на языке Си. Управление ресурсами: Справ. пособие./ А.И. Касаткин – ил. (машинно-зависимое программирование на Си под DOS) – Мн.: Выш. шк., 1992. - 432 с.

7 Культин, Н.Б. C++Builderв задачах и примерах./ Н.Б. Культин. – СПб.: БХВ-Петербург, 2005. – 336.с.

8 Подбельский, В.В. Программирование на языке Си. / В.В. Подбельский, С. С. Фомин. – М.: ФиС, 1999. – 600 с.

9 Павловская, Т.А. С/С++. Программирование на языке высокого уровня. / Т.А. Павловская – СПб: Питер, 2003. – 461с.

10 Романчик, В. С. Программирование в C++ BUILDER: учебное пособие для студ. механико-матем. фак. / В. С. Романчик, А.Е. Люлькин. – ISBN 985-485-498-1 – Мн.: БГУ, 2007. –126 с.

11 Романов, Е.Л. Информатика. Основы анализа и проектирования программ. Конспект лекций и методические указания к лабораторным работам по дисциплине "Информатика". / Е.Л. Романов - (" методичка" для 1-го семестра, библиотека НГТУ - 73Р693) – Новосибирск: Изд-во НГТУ, 1999. – 80 с.

12 Романов, Е.Л. Язык Си. Типы данных и управление памятью. Конспект лекций. Тестовые вопросы и задания к лабораторным работам. / Е.Л. Романов – ("методичка" для 1,2-го семестра, библиотека НГТУ - 51Р693) – Новосибирск, Изд-во НГТУ, 2000. – 62 с.

13 СТО ЮУрГУ 04 – 2008: Курсовое и дипломное проектирование: Общие требования к оформлению – Издательство «ЮУрГУ», 2008. – 56 с.

14 Технология гофрокартона: учебное пособие. / А.С. Смолин, В.И. Комаров, В.К. Дубовый, В.И. Белоглазов – СПб: Издательство СПбГТУРП 2014. – 146 с.

15 Технологический регламент на изготовление гофрированного картона и гофротары на ООО «Фабрика Южуралкартон» / Разработал А.С. Новиков. – Коркино, 2010 г. – 38 с.

16 Топп ,У. Структуры данных в Си++. / У. Топп, У. Форд. – М.: ЗАО БИНОМ, 1999. – 800 с.

17 Шуп, Т. Решение инженерных задач на ЭВМ: Практическое руководство.Пер. с англ. / Т. Шуп – М.: Мир, 1982. – 238 с.

18 Шилдт, Г. Самоучитель С++: Пер.с англ. / Г. Шилдт – 3-е изд. – СПб.: БХВ-Петербург, 2006. –688.с.

ПРИЛОЖЕНИЕ А.

Исходный код программы

Листинг файла ProgPlan_1.~cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "ProgPlan_1.h"
#include "Unit4.h"
#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int i1=1;
bool flag=false;
TStringList *Table1 = new TStringList;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormActivate(TObject *Sender)
{
char *k="C:\\Users\\Public\\Documents\\Table3.txt";
    WIN32_FIND_DATA FindFileData;
    HANDLE h;
    h=FindFirstFile(k, &FindFileData);
    if (h!=INVALID_HANDLE_VALUE)
 {
Table1->LoadFromFile("C:\\Users\\Public\\Documents\\Table1.txt");
StringGrid1->RowCount = Table1->Count;
  for(int i = 0 ; i<StringGrid1->RowCount ; i++)
    {
```

```
      StringGrid1->Rows[i]->DelimitedText = Table1->Strings[i] ;
    }
  }
   StringGrid1->ColWidths[0]=50;
   StringGrid1->ColWidths[1]=160;
   StringGrid1->ColWidths[4]=140;
   StringGrid1->ColWidths[6]=115;
   StringGrid1->ColWidths[8]=185;
   StringGrid1->Cells[0][0]="Номер";
   StringGrid1->Cells[1][0]="Наименование изделия";
   StringGrid1->Cells[2][0]="Длина мм";
   StringGrid1->Cells[3][0]="Ширина мм";
   StringGrid1->Cells[4][0]="Биговки мм";
   StringGrid1->Cells[5][0]="Марка";
   StringGrid1->Cells[6][0]="Площадь изделия м2";
   StringGrid1->Cells[7][0]="Профиль";
   StringGrid1->Cells[8][0]="Перерабатывающая линия";
   StringGrid1->Cells[9][0]="Уникальный код";
   StringGrid1->Cells[10][0]="id качества";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Edit2->Visible=true;
Edit3->Visible=true;
Edit4->Visible=true;
Edit5->Visible=true;
ComboBox1->Visible=true;
Label1->Visible=true;
Label2->Visible=true;
Label3->Visible=true;
Label4->Visible=true;
Label5->Visible=true;
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
```

```
Button3->Visible=true;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
if (Edit2->Text.IsEmpty()|| Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty()
||
ComboBox1->Text.IsEmpty())
 {
  ShowMessage("Не все поля заполнены");
 }
 else
 {
 int j=StringGrid1->RowCount-1;
 if (CheckBox1->Checked==true)
  {
   StringGrid1->Cells[8][j]="9PA-2";
  }
   if (CheckBox2->Checked==true)
  {
   StringGrid1->Cells[8][j]="9PA-3";
  }
   if (CheckBox3->Checked==true)
  {
   StringGrid1->Cells[8][j]="5PA-2";
  }
   if (CheckBox4->Checked==true)
  {
   StringGrid1->Cells[8][j]="IS";
  }
   if (CheckBox5->Checked==true)
  {
   StringGrid1->Cells[8][j]="AS";
  }
   if (CheckBox6->Checked==true)
  {
```

```
  StringGrid1->Cells[8][j]="W";
 }
  if (CheckBox7->Checked==true)
 {
  StringGrid1->Cells[8][j]="V";
 }
  if (CheckBox8->Checked==true)
 {
  StringGrid1->Cells[8][j]="В.П.";
 }
AnsiString prof=ComboBox1->Text, mar=ComboBox1->Text;
int a=StrToInt(Edit3->Text);
int b=StrToInt(Edit4->Text);
float m=((float)a*(float)b)/1000000;
Form1->StringGrid1->Cells[0][j]=j;
Form1->StringGrid1->Cells[1][j]=Edit2->Text;
Form1->StringGrid1->Cells[2][j]=Edit3->Text;
Form1->StringGrid1->Cells[3][j]=Edit4->Text;
Form1->StringGrid1->Cells[4][j]=Edit5->Text;
Form1->StringGrid1->Cells[6][j]=m;
Form1->StringGrid1->Cells[7][j]=prof[9];
Form1->StringGrid1->Cells[5][j]=mar[2];
Form1->StringGrid1->Cells[9][j]=ComboBox1->Text;
Form1->StringGrid1->Cells[9][j]=ComboBox1->ItemIndex;
Button2->Visible=true;
Button3->Visible=false;
StringGrid1->RowCount=StringGrid1->RowCount+1;
 Edit2->Visible=false;
 Edit3->Visible=false;
 Edit4->Visible=false;
 Edit5->Visible=false;
 ComboBox1->Visible=false;
 Label1->Visible=false;
 Label2->Visible=false;
 Label3->Visible=false;
 Label4->Visible=false;
 Label5->Visible=false;
```

```
        }
        CheckBox1->Checked=false;CheckBox2->Checked=false;CheckBox3-
>Checked=false;
        CheckBox4->Checked=false;CheckBox5->Checked=false;CheckBox6-
>Checked=false;
        CheckBox7->Checked=false;CheckBox8->Checked=false;
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::N2Click(TObject *Sender)
        {
        Form4->Show();
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::N8Click(TObject *Sender)
        {
        Form1->Close();
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::Button5Click(TObject *Sender)
        {
        Form2->Show();
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::FormCreate(TObject *Sender)
        {
        Form1->Caption ="Программа";
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::Button6Click(TObject *Sender)
        {
        Form1->Close();
        }
        //---------------------------------------------------------------------
        void __fastcall TForm1::Button4Click(TObject *Sender)
        {
        Table1->Clear();
        for(int n = 0 ; n<StringGrid1->RowCount ; n++)
```

```
  {
  Table1->Add(StringGrid1->Rows[n]->DelimitedText) ;
  }
Table1->SaveToFile("C:\\Users\\Public\\Documents\\Table1.txt") ;
 }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form3->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
flag=true;
Label15->Caption="Выберите строку";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Enter(TObject *Sender)
{
    for (int i=1; i<Form2->StringGrid1->RowCount; i++)
 {
ComboBox1->Items->Add(Form2->StringGrid1->Cells[11][i]);
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit3KeyPress(TObject *Sender, char &Key)
{
  if (Key==13)
  {
   Edit4->SetFocus();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2KeyPress(TObject *Sender, char &Key)
{
  if (Key==13)
  {
```

```
   Edit3->SetFocus();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit4KeyPress(TObject *Sender, char &Key)
{
  if (Key==13)
  {
   Edit5->SetFocus();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit5KeyPress(TObject *Sender, char &Key)
{
  if (Key==13)
  {
   ComboBox1->SetFocus();
  }
}
//---------------------------------------------------------------------------
bool f=true;
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
 SpeedButton1->AllowAllUp=true;
 if (f)
 {
 SpeedButton1->GroupIndex=1;
 SpeedButton1->Down=true;
 StringGrid1->Options=StringGrid1->Options<<goEditing;
 Label15->Caption="Режим редактирования";
 f=false;
 }
 else
 {
  SpeedButton1->GroupIndex=0;
  SpeedButton1->Down=false;
  StringGrid1->Options=StringGrid1->Options>>goEditing;
```

```
 Label15->Caption="Режим просмотра";
 f=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid1SelectCell(TObject *Sender, int ACol,
     int ARow, bool &CanSelect)
{
 if (flag)
 {
  for (int i=ARow;i<StringGrid1->RowCount-1;i++)
   {
    for (int j=1; j<StringGrid1->ColCount;j++)
    {
     StringGrid1->Cells[j][i]=StringGrid1->Cells[j][i+1];
     StringGrid1->Cells[0][i]=i;
    }
   }
   StringGrid1->RowCount=StringGrid1->RowCount-1;
   StringGrid1->Rows[StringGrid1->RowCount]->Clear();
 }
 flag=false;
 Label15->Caption="Режим просмотра";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid1Enter(TObject *Sender)
{
 for (int i=1;i<StringGrid1->RowCount-1;i++)
 {
 AnsiString p=StringGrid1->Cells[9][i];
 int a=StrToInt(StringGrid1->Cells[2][i]);
 int b=StrToInt((StringGrid1->Cells[3][i]));
 StringGrid1->Cells[6][i]=((float)a*(float)b)/1000000;
 StringGrid1->Cells[7][i]=p[9];
 StringGrid1->Cells[5][i]=p[2];
//                          StringGrid1->Cells[5][i]=Form2->StringGrid1-
>Cells[1][StrToInt(Form1->StringGrid1->Cells[10][i])];
```

```
 }

}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox9Click(TObject *Sender)
{
if(CheckBox9->Checked==true)
 {
  ComboBox2->Enabled=true;
  CheckBox10->Enabled=false;
  CheckBox11->Enabled=false;
  CheckBox12->Enabled=false;
  for (int i=1; i<StringGrid1->RowCount-1;i++)
  {
   ComboBox2->Items->Add(StringGrid1->Cells[1][i]);
  }
 }
 else
 {
 ComboBox2->Enabled=false;
 CheckBox10->Enabled=true;
 CheckBox11->Enabled=true;
 CheckBox12->Enabled=true;
 ComboBox2->Items->Clear();
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox12Click(TObject *Sender)
{
 if(CheckBox12->Checked==true)
 {
  ComboBox3->Enabled=true;
  CheckBox9->Enabled=false;
  for (int i=1; i<StringGrid1->RowCount-1;i++)
  {
   ComboBox3->Items->Add(StringGrid1->Cells[9][i]);
  }
```

```
 }
 else
 {
 ComboBox3->Enabled=false;
 CheckBox9->Enabled=true;
 ComboBox3->Items->Clear();
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox10Click(TObject *Sender)
{
if(CheckBox10->Checked==true)
 {
  Edit1->Enabled=true;
  TrackBar1->Enabled=true;
  CheckBox9->Enabled=false;
 }
 else
 {
  Edit1->Enabled=false;
  TrackBar1->Enabled=false;
  CheckBox9->Enabled=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox11Click(TObject *Sender)
{
if(CheckBox11->Checked==true)
 {
  Edit6->Enabled=true;
  TrackBar2->Enabled=true;
  CheckBox9->Enabled=false;
 }
 else
 {
  Edit6->Enabled=false;
  TrackBar2->Enabled=false;
```

```
  CheckBox9->Enabled=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
{
  Button8->Visible=false;
  Button9->Visible=true;
  StringGrid2->ColWidths[0]=50;
  StringGrid2->ColWidths[1]=160;
  StringGrid2->ColWidths[4]=140;
  StringGrid2->ColWidths[6]=115;
  StringGrid2->ColWidths[8]=185;
  StringGrid2->Cells[0][0]="Номер";
  StringGrid2->Cells[1][0]="Наименование изделия";
  StringGrid2->Cells[2][0]="Длина мм";
  StringGrid2->Cells[3][0]="Ширина мм";
  StringGrid2->Cells[4][0]="Биговки мм";
  StringGrid2->Cells[5][0]="Марка";
  StringGrid2->Cells[6][0]="Площадь изделия м2";
  StringGrid2->Cells[7][0]="Профиль";
  StringGrid2->Cells[8][0]="Перерабатывающая линия";
  StringGrid2->Cells[9][0]="Уникальный код";
  StringGrid2->Cells[10][0]="id качества";
 if (CheckBox9->Checked==true)
 {
  for (int i=1;i<StringGrid1->RowCount-1;i++)
  {
   if (StringGrid1->Cells[1][i]==ComboBox2->Text)
   {
    StringGrid2->Visible=true;
    StringGrid1->Visible=false;
    for (int j=0;j<StringGrid1->ColCount;j++)
    {
    StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
    }
```

```
            StringGrid2->RowCount=StringGrid2->RowCount+1;
           }
          }
         }
        if (CheckBox10->Checked==true && CheckBox11->Checked==false &&
CheckBox12->Checked==false)
        {
         int a=TrackBar1->Position;
         for (int i=1;i<StringGrid1->RowCount-1;i++)
          {
           for(int k=0;k<=a;k++)
           {
            if (StrToInt(StringGrid1->Cells[2][i])==(StrToInt(Edit1->Text)+k))
            {
             StringGrid2->Visible=true;
             StringGrid1->Visible=false;
             for (int j=0;j<StringGrid1->ColCount;j++)
             {
             StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
              }
             StringGrid2->RowCount=StringGrid2->RowCount+1;
            }
           }
          }
         }
        if (CheckBox10->Checked==true && CheckBox11->Checked==true
&&CheckBox12->Checked==false)
        {
         int a=TrackBar1->Position;
         int b=TrackBar2->Position;
         for (int i=1;i<StringGrid1->RowCount-1;i++)
          {
           for(int k=0;k<=a;k++)
           {
            for(int x=0;x<=b;x++)
            {
```

```
            if (StrToInt(StringGrid1->Cells[2][i])==(StrToInt(Edit1->Text)+k) &&
            StrToInt(StringGrid1->Cells[3][i])==(StrToInt(Edit6->Text)+x))
            {
             StringGrid2->Visible=true;
             StringGrid1->Visible=false;
             for (int j=0;j<StringGrid1->ColCount;j++)
             {
             StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
             }
             StringGrid2->RowCount=StringGrid2->RowCount+1;
            }
           }
          }
         }
        if (CheckBox10->Checked==true && CheckBox11->Checked==true &&
CheckBox12->Checked==true)
        {
         int a=TrackBar1->Position;
         int b=TrackBar2->Position;
         for (int i=1;i<StringGrid1->RowCount-1;i++)
          {
           for(int k=0;k<=a;k++)
           {
            for(int x=0;x<=b;x++)
            {
            if (StrToInt(StringGrid1->Cells[2][i])==(StrToInt(Edit1->Text)+k) &&
            StrToInt(StringGrid1->Cells[3][i])==(StrToInt(Edit6->Text)+x) &&
            StringGrid1->Cells[9][i]==ComboBox3->Text)
            {
             StringGrid2->Visible=true;
             StringGrid1->Visible=false;
             for (int j=0;j<StringGrid1->ColCount;j++)
             {
             StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
```

```
            }
            StringGrid2->RowCount=StringGrid2->RowCount+1;
           }
          }
         }
        }
       }
      if (CheckBox10->Checked==true && CheckBox11->Checked==false &&
CheckBox12->Checked==true)
      {
       int a=TrackBar1->Position;
       for (int i=1;i<StringGrid1->RowCount-1;i++)
        {
         for(int k=0;k<=a;k++)
         {
          if (StrToInt(StringGrid1->Cells[2][i])==(StrToInt(Edit1->Text)+k)&&
          StringGrid1->Cells[9][i]==ComboBox3->Text)
          {
           StringGrid2->Visible=true;
           StringGrid1->Visible=false;
           for (int j=0;j<StringGrid1->ColCount;j++)
           {
           StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
            }
            StringGrid2->RowCount=StringGrid2->RowCount+1;
           }
          }
         }
        }
       if (CheckBox10->Checked==false && CheckBox11->Checked==true
&& CheckBox12->Checked==true)
      {
       int b=TrackBar2->Position;
       for (int i=1;i<StringGrid1->RowCount-1;i++)
        {
         for(int k=0;k<=b;k++)
```

```
            {
             if (StrToInt(StringGrid1->Cells[3][i])==(StrToInt(Edit6->Text)+k)&&
             StringGrid1->Cells[9][i]==ComboBox3->Text)
             {
              StringGrid2->Visible=true;
              StringGrid1->Visible=false;
              for (int j=0;j<StringGrid1->ColCount;j++)
              {
              StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
               }
              StringGrid2->RowCount=StringGrid2->RowCount+1;
             }
            }
           }
           }
           if (CheckBox10->Checked==false && CheckBox11->Checked==true
&& CheckBox12->Checked==false)
          {
           int b=TrackBar2->Position;
           for (int i=1;i<StringGrid1->RowCount-1;i++)
            {
             for(int k=0;k<=b;k++)
             {
              if (StrToInt(StringGrid1->Cells[3][i])==(StrToInt(Edit1->Text)+k))
              {
               StringGrid2->Visible=true;
               StringGrid1->Visible=false;
               for (int j=0;j<StringGrid1->ColCount;j++)
               {
               StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
                }
               StringGrid2->RowCount=StringGrid2->RowCount+1;
              }
             }
            }
```

```
  }
 else {ShowMessage("Нет данных"); }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
Label16->Caption=TrackBar1->Position;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TrackBar2Change(TObject *Sender)
{
Label17->Caption=TrackBar2->Position;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button9Click(TObject *Sender)
{
  Button8->Visible=true;
  Button9->Visible=false;
  StringGrid1->Visible=true;
  StringGrid2->Visible=false;
  StringGrid2->RowCount=2;
  StringGrid2->Rows[1]->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N9Click(TObject *Sender)
{
 if(N9->Checked!=true)
 {
 N9->Checked=true;
 N10->Checked=false;
 N11->Checked=false;
  for(int i=1;i<StringGrid1->RowCount-1;i++)
  {
   if (StringGrid1->Cells[7][i]=="B")
   {
     StringGrid2->Visible=true;
     StringGrid1->Visible=false;
```

```
            for (int j=0;j<StringGrid1->ColCount;j++)
             {
             StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
             }
             StringGrid2->RowCount=StringGrid2->RowCount+1;
            }
           }
          }
          else
          {
           N9->Checked=false;
           StringGrid2->Visible=false;
           StringGrid1->Visible=true;
           StringGrid2->RowCount=2;
           StringGrid2->Rows[1]->Clear();
          }
         }
         //---------------------------------------------------------------------------
         void __fastcall TForm1::N10Click(TObject *Sender)
         {
         if(N10->Checked!=true)
          {
          N9->Checked=false;
          N10->Checked=true;
          N11->Checked=false;
           for(int i=1;i<StringGrid1->RowCount-1;i++)
           {
            if (StringGrid1->Cells[7][i]=="C")
            {
              StringGrid2->Visible=true;
              StringGrid1->Visible=false;
              for (int j=0;j<StringGrid1->ColCount;j++)
               {
               StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
               }
```

```
            StringGrid2->RowCount=StringGrid2->RowCount+1;
           }
          }
          }
         else
         {
          N10->Checked=false;
          StringGrid2->Visible=false;
          StringGrid1->Visible=true;
          StringGrid2->RowCount=2;
          StringGrid2->Rows[1]->Clear();
         }
        }
        //---------------------------------------------------------------------------
        void __fastcall TForm1::N11Click(TObject *Sender)
        {
        if(N11->Checked!=true)
         {
         N9->Checked=false;
         N10->Checked=false;
         N11->Checked=true;
          for(int i=1;i<StringGrid1->RowCount-1;i++)
          {
           if (StringGrid1->Cells[7][i]=="E")
           {
            StringGrid2->Visible=true;
            StringGrid1->Visible=false;
            for (int j=0;j<StringGrid1->ColCount;j++)
             {
             StringGrid2->Cells[j][StringGrid2->RowCount-1]=StringGrid1-
>Cells[j][i];
             }
             StringGrid2->RowCount=StringGrid2->RowCount+1;
            }
          }
          }
         else
```

```
 {
  N11->Checked=false;
  StringGrid2->Visible=false;
  StringGrid1->Visible=true;
  StringGrid2->RowCount=2;
  StringGrid2->Rows[1]->Clear();
 }
}
//---------------------------------------------------------------------------
```

Листинг файла Unit2.~cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "ProgPlan_1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
TStringList *Table2 = new TStringList;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Label1Click(TObject *Sender)
{
Form2->Label1->Caption=Form1->StringGrid1->Cells[1][1];
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormActivate(TObject *Sender)
{
Form2->StringGrid1->SetFocus();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm2::RadioGroup1Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex==0)
 {
  Label6->Visible=false;
  Label7->Visible=false;
  Edit5->Visible=false;
  Edit5->Clear();
  Edit6->Clear();
  Edit6->Visible=false;
  RadioGroup5->Enabled=false;
  RadioGroup4->Enabled=true;
 }
if (RadioGroup1->ItemIndex==1)
 {
  Label6->Visible=true;
  Label7->Visible=true;
  Edit5->Visible=true;
  Edit6->Visible=true;
  RadioGroup4->Enabled=false;
  RadioGroup5->Enabled=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
 Table2->Clear();
for(int n = 0 ; n<StringGrid1->RowCount ; n++)
  {
  Table2->Add(StringGrid1->Rows[n]->DelimitedText) ;
  }
Table2->SaveToFile("C:\\Users\\Public\\Documents\\Table2.txt") ;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button3Click(TObject *Sender)
{
Table2->LoadFromFile("C:\\Users\\Public\\Documents\\Table2.txt");
```

```
StringGrid1->RowCount = Table2->Count;
 for(int i = 0 ; i<StringGrid1->RowCount ; i++)
  {
   StringGrid1->Rows[i]->DelimitedText = Table2->Strings[i] ;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
if (ComboBox1->Text.IsEmpty()|| Edit2->Text.IsEmpty() || Edit3->Text.IsEmpty() ||
Edit4->Text.IsEmpty() || ComboBox2->Text.IsEmpty())
 {
 ShowMessage("Не все поля заполнены");
 }
 else
 {
  int i=StringGrid1->RowCount-1;
  StringGrid1->Cells[0][i]=i;
  StringGrid1->Cells[1][i]=ComboBox1->Text;
  StringGrid1->Cells[2][i]=ComboBox2->Text;
  StringGrid1->Cells[3][i]=Edit2->Text;
  StringGrid1->Cells[4][i]=Edit3->Text;
  StringGrid1->Cells[5][i]=Edit4->Text;
  StringGrid1->Cells[6][i]=Edit5->Text;
  StringGrid1->Cells[7][i]=Edit6->Text;
  if (RadioGroup1->ItemIndex==0)
  {
   if (RadioGroup4->ItemIndex==0)
   {
    StringGrid1->Cells[8][i]=" E ";
   }
   if (RadioGroup4->ItemIndex==1)
   {
    StringGrid1->Cells[8][i]=" B ";
   }
   if (RadioGroup4->ItemIndex==2)
```

```
 {
  StringGrid1->Cells[8][i]=" C ";
 }
}
else
{
 if (RadioGroup5->ItemIndex==0)
 {
 StringGrid1->Cells[8][i]=" BE ";
 }
 if (RadioGroup5->ItemIndex==1)
 {
 StringGrid1->Cells[8][i]=" BC ";
 }
}
if (RadioGroup2->ItemIndex==0)
 {
  StringGrid1->Cells[9][i]=1;
 }
if (RadioGroup2->ItemIndex==1)
 {
  StringGrid1->Cells[9][i]=2;
 }
 if (RadioGroup2->ItemIndex==2)
 {
  StringGrid1->Cells[9][i]=3;
 }
 if (RadioGroup3->ItemIndex==0)
 {
  StringGrid1->Cells[10][i]=1;
 }
 if (RadioGroup3->ItemIndex==1)
 {
  StringGrid1->Cells[10][i]=2;
 }
 if (RadioGroup3->ItemIndex==2)
 {
```

```
 StringGrid1->Cells[10][i]=3;
 }
 if(RadioGroup1->ItemIndex==0)
 {
 StringGrid1->Cells[12][i]=("Т");
 }
 if(RadioGroup1->ItemIndex==1)
 {
 StringGrid1->Cells[12][i]=("П");
 }
 StringGrid1->Cells[11][i]=StringGrid1->Cells[12][i]+
 IntToStr(StrToInt(StringGrid1->Cells[1][i])-20)+
 StringGrid1->Cells[9][i]+StringGrid1->Cells[10][i]+StringGrid1-
>Cells[2][i]+
 StringGrid1->Cells[8][i];
 StringGrid1->RowCount=StringGrid1->RowCount+1;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormCreate(TObject *Sender)
{
StringGrid2->ColWidths[0]=40;
for (int i=1; i<4; i++)
 {
 ComboBox2->Items->Add("("+IntToStr(i)+")");
 }
 StringGrid2->Cells[0][0]="Марка";
 for (int i=1; i<28; i++)
 {
 StringGrid2->Cells[0][i]=i;
 StringGrid2->RowCount=StringGrid2->RowCount+1;
 }
 for (int i2=28;i2<35;i2++)
 {
 StringGrid2->Cells[0][i2]=i2+3;
 StringGrid2->RowCount=StringGrid2->RowCount+1;
 }
```

```
for (int i=21; i<StringGrid2->RowCount; i++)
 {
  ComboBox1->Items->Add(StringGrid2->Cells[0][i]);
 }
StringGrid4->Cells[0][0]=("id");
StringGrid4->ColWidths[0]=40;
StringGrid4->ColWidths[1]=200;
StringGrid4->Cells[1][0]=("тип лицевого");
StringGrid4->Cells[0][1]=("1");
StringGrid4->Cells[0][2]=("2");
StringGrid4->Cells[0][3]=("3");
StringGrid4->Cells[1][1]=("Целлюлозный");
StringGrid4->Cells[1][2]=("Макулатурный");
StringGrid4->Cells[1][3]=("Белый");
StringGrid3->Cells[0][0]=("id");
StringGrid3->ColWidths[0]=40;
StringGrid3->ColWidths[1]=200;
StringGrid3->Cells[1][0]=("тип внутреннего");
StringGrid3->Cells[0][1]=("1");
StringGrid3->Cells[0][2]=("2");
StringGrid3->Cells[0][3]=("3");
StringGrid3->Cells[1][1]=("Влагостойкий");
StringGrid3->Cells[1][2]=("Целлюлозный");
StringGrid3->Cells[1][3]=("Макулатурный");
StringGrid1->Cells[0][0]=("№");
StringGrid1->Cells[1][0]=("Марка");
StringGrid1->Cells[2][0]=("ECT");
StringGrid1->Cells[3][0]=("L");
StringGrid1->Cells[4][0]=("F");
StringGrid1->Cells[5][0]=("L");
StringGrid1->Cells[6][0]=("F");
StringGrid1->Cells[7][0]=("L");
StringGrid1->Cells[8][0]=("Профиль");
StringGrid1->Cells[9][0]=("Лицевой слой");
StringGrid1->Cells[10][0]=("Внутренний слой");
StringGrid1->Cells[11][0]=("Уникальный код");
StringGrid1->Cells[12][0]=("Тип");
```

```
Edit5->Visible=false;
Edit6->Visible=false;
Label6->Visible=false;
Label7->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::N1Click(TObject *Sender)
{
StringGrid1->Options = StringGrid1->Options << goEditing;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::RadioGroup3Click(TObject *Sender)
{
Button1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::RadioGroup2Click(TObject *Sender)
{
RadioGroup3->Enabled=true;
}
//---------------------------------------------------------------------------
```

Листинг файла Unit3.~cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
#include "ProgPlan_1.h"
#include "Unit2.h"
#include "Unit4.h"
#include "Unit5.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
TStringList *Table3 = new TStringList;
int n=1;//переменная для таблицы изделий
```

```
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormCreate(TObject *Sender)
{
StringGrid2->Cells[0][0]="#";
StringGrid2->Cells[1][0]="Изделие";
StringGrid2->Cells[2][0]="Требуется шт.";
StringGrid2->Cells[3][0]="Уникальный код";
StringGrid2->Cells[4][0]="id";
StringGrid2->Cells[5][0]="обрезь";
StringGrid5->Cells[0][0]="#";
StringGrid5->Cells[1][0]="Изделие";
StringGrid5->Cells[2][0]="Требуется шт.";
StringGrid5->Cells[3][0]="Уникальный код";
StringGrid5->Cells[4][0]="id";
StringGrid5->Cells[5][0]="обрезь";
char *k="C:\\Users\\Public\\Documents\\Table3.txt";
    WIN32_FIND_DATA FindFileData;
    HANDLE h;
    h=FindFirstFile(k, &FindFileData);
    if (h!=INVALID_HANDLE_VALUE)
 {
Table3->LoadFromFile("C:\\Users\\Public\\Documents\\Table3.txt");
StringGrid1->RowCount = Table3->Count;
  for(int i = 0 ; i<StringGrid1->RowCount ; i++)
    {
      StringGrid1->Rows[i]->DelimitedText = Table3->Strings[i] ;
    }
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::StringGrid1SelectCell(TObject *Sender, int ACol,
```

```
    int ARow, bool &CanSelect)
{
if (StringGrid1->Cells[ACol][ARow].IsEmpty()==true)
 {
  StringGrid1->Options=StringGrid1->Options<<goEditing;
  StringGrid1->RowCount=StringGrid1->RowCount+1;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button1Click(TObject *Sender)
{
  Table3->Clear();
for(int n = 0 ; n<StringGrid1->RowCount ; n++)
  {
  Table3->Add(StringGrid1->Rows[n]->DelimitedText) ;
  }
Table3->SaveToFile("C:\\Users\\Public\\Documents\\Table3.txt") ;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::N1Click(TObject *Sender)
{
//StringGrid1->
}
//---------------------------------------------------------------------------
void __fastcall TForm3::ComboBox1Click(TObject *Sender)
{
ComboBox1->Items->Add(Form1->StringGrid1->Cells[0][0]);
}
//---------------------------------------------------------------------------
void __fastcall TForm3::ComboBox1KeyPress(TObject *Sender, char
&Key)
{
if (Key==13)
 {
  int pos;
  for (int i=1; i<Form1->StringGrid1->RowCount; i++)
  {
```

```
  if (ComboBox1->Text==Form1->StringGrid1->Cells[1][i])
  {
  pos=StrToInt(Form1->StringGrid1->Cells[0][i]);
  StringGrid2->Cells[0][n]=IntToStr(n);
  StringGrid2->Cells[1][n]=Form1->StringGrid1->Cells[1][pos];
  StringGrid2->Cells[2][n]=Edit1->Text;
  StringGrid2->Cells[3][n]=Form1->StringGrid1->Cells[9][pos];
  StringGrid2->Cells[4][n]=pos;
  StringGrid2->Cells[5][n]=Form1->StringGrid1->Cells[0][i];
  if (CheckBox1->Checked==true)
   {
   StringGrid2->Cells[5][n]=0;
   }
  StringGrid2->RowCount=StringGrid2->RowCount+1;
  n=n+1;
  Edit1->Clear();
  ComboBox1->Clear();
  ComboBox1->Enabled=false;
   }
  }
 }
 CheckBox1->Checked=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Edit1KeyPress(TObject *Sender, char &Key)
{
if (Key==13)
 {
 if (Edit1->Text.IsEmpty())
  {
   ShowMessage("Ошибка ввода, повторите");
  }
 else
  {
 ComboBox1->Enabled=true;
 ComboBox1->SetFocus();
```

```
 for (int i=1; i<Form1->StringGrid1->RowCount; i++)
   {
   ComboBox1->Items->Add(Form1->StringGrid1->Cells[1][i]);
   }
  }
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button2Click(TObject *Sender)
{
if (StringGrid2->RowCount<3)
 {
  ShowMessage("Данные не введенны");
 }
int del1=0;
for (int q=1;q<StringGrid2->RowCount-1;q++)
 {
  if (!(StringGrid2->Cells[0][q].IsEmpty()))
  {
   if (!(StringGrid2->Cells[5][q].IsEmpty()))
    {
    if (StrToInt(StringGrid2->Cells[5][q])==0)
     {
     int   shir=StrToInt(Form1->StringGrid1->Cells[3][StrToInt(StringGrid2-
>Cells[4][q])]);
     for(int w=1;w<StringGrid1->RowCount-1;w++)
     {
     int format=StrToInt(StringGrid1->Cells[0][w]);
     if (format%shir==0)
      {
      int st5=StringGrid5->RowCount-1;
      StringGrid3->Cells[0][st5]=st5;
      StringGrid3->Cells[1][st5]=StringGrid2->Cells[1][q];
      StringGrid3->Cells[2][st5]=format/shir;
      StringGrid3->Cells[3][st5]=format;
      StringGrid3->Cells[4][st5]=shir;
```

```
            StringGrid3->Cells[5][st5]=Form1->StringGrid1-
>Cells[2][StrToInt(Form3->StringGrid2->Cells[4][q])]; //L
            StringGrid3->Cells[6][st5]=StrToInt(StringGrid2-
>Cells[2][q])/(format/shir);//Rez
            StringGrid3->Cells[7][st5]=StringGrid2->Cells[2][q]; //Unit
            StringGrid3->Cells[8][st5]=Form1->StringGrid1-
>Cells[8][StrToInt(Form3->StringGrid2->Cells[4][q])]; //Car
            StringGrid3->Cells[9][st5]=Form2->StringGrid1-
>Cells[3][StrToInt(Form1->StringGrid1->Cells[10][StrToInt(StringGrid2-
>Cells[4][q])])]; //L
            StringGrid3->Cells[10][st5]=Form2->StringGrid1-
>Cells[4][StrToInt(Form1->StringGrid1->Cells[10][StrToInt(StringGrid2-
>Cells[4][q])])]; //F
            StringGrid3->Cells[11][st5]=Form2->StringGrid1-
>Cells[5][StrToInt(Form1->StringGrid1->Cells[10][StrToInt(StringGrid2-
>Cells[4][q])])]; //L
            StringGrid3->Cells[12][st5]=Form2->StringGrid1-
>Cells[6][StrToInt(Form1->StringGrid1->Cells[10][StrToInt(StringGrid2-
>Cells[4][q])])]; //F
            StringGrid3->Cells[13][st5]=Form2->StringGrid1-
>Cells[7][StrToInt(Form1->StringGrid1->Cells[10][StrToInt(StringGrid2-
>Cells[4][q])])]; //L
            StringGrid3->Cells[21][st5]=StrToInt(Form1->StringGrid1-
>Cells[2][StrToInt(StringGrid2->Cells[4][q])])*StrToInt(StringGrid2-
>Cells[2][q])/(format/shir);//m
            StringGrid3->Cells[22][st5]=format%shir;//O
            StringGrid3->Cells[23][st5]=StringGrid2->Cells[3][q];
            StringGrid3->Cells[24][st5]=StringGrid2->Cells[4][q];
            del1=del1+1;
            StringGrid3->RowCount=StringGrid3->RowCount+1;
            for (int w1=q;w1<StringGrid2->RowCount-1;w1++)
             {
             for (int w2=0;w2<StringGrid2->ColCount;w2++)
             StringGrid2->Cells[w2][w1]=StringGrid2->Cells[w2][w1+1];
             }
            }
           }
```

```
            }
           }
          }
         }
        StringGrid2->RowCount=StringGrid2->RowCount-del1;
        //---------------------------------------------------------------------без обрези
        int min=StrToInt(Edit3->Text);
        int max=StrToInt(Edit2->Text);
        int st3=1, st2=1;
        bool zapis=false, chek1=false;
        StringGrid4->Cells[0][0]="номер";
        StringGrid4->Cells[1][0]="id";
        StringGrid4->Cells[2][0]="требуется";
        StringGrid4->Cells[3][0]="длина заказа";
        StringGrid4->Cells[4][0]="id2";
        while(StringGrid2->RowCount>2 && !(StringGrid2->Cells[0][1].IsEmpty()))  //while глав
        {

        StringGrid4->RowCount=3;
        StringGrid4->Rows[2]->Clear();
        StringGrid4->Rows[1]->Clear();
        int j=2, del=0;
         for (int i=2;i<StringGrid2->RowCount-1; i++)
         {
          if (StringGrid2->Cells[3][1]==StringGrid2->Cells[3][i])          //if ==
          {
            if (chek1==false)
            {
              StringGrid4->Cells[0][1]=StringGrid2->Cells[0][1];
              StringGrid4->Cells[1][1]=StringGrid2->Cells[4][1];
              StringGrid4->Cells[2][1]=StringGrid2->Cells[2][1];
              StringGrid4->Cells[3][1]=StrToInt(StringGrid2->Cells[2][1]*StrToInt(Form1->
              StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][1])]));
              n=n-1;
              StringGrid4->Cells[0][j]=StringGrid2->Cells[0][i];
```

```
                StringGrid4->Cells[1][j]=StringGrid2->Cells[4][i];
                StringGrid4->Cells[2][j]=StringGrid2->Cells[2][i];
                StringGrid4->Cells[3][j]=StrToInt(StringGrid2-
>Cells[2][i]*StrToInt(Form1->
                StringGrid1->Cells[2][StrToInt(StringGrid2->Cells[4][i])]));
                StringGrid4->RowCount=StringGrid4->RowCount+1;
                j=j+1;
                n=n-1;
                del=del+2;
                chek1=true;
               }  // if chek1
              else
               {
               StringGrid4->Cells[0][j]=StringGrid2->Cells[0][i];
               StringGrid4->Cells[1][j]=StringGrid2->Cells[4][i];
               StringGrid4->Cells[2][j]=StringGrid2->Cells[2][i];
               StringGrid4->Cells[3][j]=StrToInt(StringGrid2-
>Cells[2][i]*StrToInt(Form1->
               StringGrid1->Cells[2][StrToInt(StringGrid2->Cells[4][i])]));
               StringGrid4->RowCount=StringGrid4->RowCount+1;
               j=j+1;
               n=n-1;
               del=del+1;
              }//else
             }// if1
            } // for1
           chek1=false;
           //-------------------------------------------------------  нашли совпадения
           if (StringGrid4->RowCount>2 && !(StringGrid4->Cells[0][2].IsEmpty()))
//if глав
           {
            int kolvo=StringGrid2->RowCount;
            int iz1=1, iz2, iz3;
            while(StringGrid2->RowCount>(kolvo-del))                    //while
            {
             iz2=StrToInt(StringGrid4->Cells[0][iz1]);
             for (int b=1;b<StringGrid2->RowCount-1; b++)
```

```
            {
             if (iz2==StrToInt(StringGrid2->Cells[0][b]))
             {
              iz3=b;
             }
            }
            for(int l=iz3;l<StringGrid2->RowCount-1;l++)
             {
              for (int l2=0;l2<StringGrid2->ColCount; l2++)
              {
               StringGrid2->Cells[l2][l]=StringGrid2->Cells[l2][l+1];
              }
             }
            StringGrid2->RowCount=StringGrid2->RowCount-1;
            n=n-1;
            iz1=iz1+1;
           } // while kolvo-del '
          //---------------------------------------------------------------------------
           int dlina=StrToInt(StringGrid4->Cells[3][1]);
              for (int i0=2; i0<StringGrid4->RowCount-1; i0++)
              {
               if (dlina<StrToInt(StringGrid4->Cells[3][i0]))
               {
              dlina=StrToInt(StringGrid4->Cells[3][i0]);
              StringGrid4->Cells[0][StringGrid4->RowCount]=StringGrid4-
>Cells[0][1];
              StringGrid4->Cells[1][StringGrid4->RowCount]=StringGrid4-
>Cells[1][1];
              StringGrid4->Cells[2][StringGrid4->RowCount]=StringGrid4-
>Cells[2][1];
              StringGrid4->Cells[3][StringGrid4->RowCount]=StringGrid4-
>Cells[3][1];
              StringGrid4->Cells[0][1]=StringGrid4->Cells[0][i0];
              StringGrid4->Cells[1][1]=StringGrid4->Cells[1][i0];
              StringGrid4->Cells[2][1]=StringGrid4->Cells[2][i0];
              StringGrid4->Cells[3][1]=StringGrid4->Cells[3][i0];
```

```
            StringGrid4->Cells[0][i0]=StringGrid4->Cells[0][StringGrid4-
>RowCount];
            StringGrid4->Cells[1][i0]=StringGrid4->Cells[1][StringGrid4-
>RowCount];
            StringGrid4->Cells[2][i0]=StringGrid4->Cells[2][StringGrid4-
>RowCount];
            StringGrid4->Cells[3][i0]=StringGrid4->Cells[3][StringGrid4-
>RowCount];
            StringGrid4->Rows[StringGrid4->RowCount]->Clear();
             }
            }
          //---------------------------------------------------------------------------
           for (int i4=2; i4<StringGrid4->RowCount-1; i4++)
            {
             for (int ii=1; ii<StringGrid1->RowCount-1;ii++)
             {
             if(StrToInt(StringGrid4->Cells[2][i4])!=0)
             {
             int                                  dlina1=StrToInt(Form1->StringGrid1-
>Cells[2][StrToInt(StringGrid4
             ->Cells[1][1])])*StrToInt(StringGrid4->Cells[2][1]);
             int                                  dlina2=StrToInt(Form1->StringGrid1-
>Cells[2][StrToInt(StringGrid4
             ->Cells[1][i4])])*StrToInt(StringGrid4->Cells[2][i4]);
             int format=StrToInt(StringGrid1->Cells[0][ii]);
             int                                 shirina1=StrToInt(Form1->StringGrid1-
>Cells[3][StrToInt(StringGrid4
             ->Cells[1][1])]);
             int                                 shirina2=StrToInt(Form1->StringGrid1-
>Cells[3][StrToInt(StringGrid4
             ->Cells[1][i4])]);
              for (int it1=1;it1<8;it1++)
               {
                for (int it2=1;it2<8;it2++)
                {
```

```
            if (format%((it1)*shirina1+(it2)*shirina2)<max && format%((it1)*shirina1+(it2)*shirina2)>min && StrToInt(StringGrid4->Cells[2][i4])!=0)
             {
            st3=StringGrid3->RowCount-1;
             int id=StrToInt(StringGrid4->Cells[1][1]);
             int id2=StrToInt(Form1->StringGrid1->Cells[10][id]);
            StringGrid3->Cells[0][st3]=st3;
            StringGrid3->Cells[1][st3]=Form1->StringGrid1->Cells[1][StrToInt(StringGrid4->Cells[1][1])];
            StringGrid3->Cells[2][st3]=it1;          //полос
            StringGrid3->Cells[3][st3]=format;
            StringGrid3->Cells[4][st3]=shirina1;
            StringGrid3->Cells[5][st3]=Form1->StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][1])];
            StringGrid3->Cells[6][st3]=StrToInt(StringGrid4->Cells[2][1])/it1; //резов
            StringGrid3->Cells[7][st3]=dlina2/StrToInt(Form1->StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][1])]);     //штук
            StringGrid3->Cells[8][st3]=Form1->StringGrid1->Cells[8][id];// car
            StringGrid3->Cells[9][st3]=Form2->StringGrid1->Cells[3][id2];
            StringGrid3->Cells[10][st3]=Form2->StringGrid1->Cells[4][id2];
            StringGrid3->Cells[11][st3]=Form2->StringGrid1->Cells[5][id2];
            StringGrid3->Cells[12][st3]=Form2->StringGrid1->Cells[6][id2];
            StringGrid3->Cells[13][st3]=Form2->StringGrid1->Cells[7][id2];
            StringGrid3->Cells[14][st3]=Form1->StringGrid1->Cells[1][StrToInt(StringGrid4->Cells[1][i4])];
            StringGrid3->Cells[15][st3]=it2; // полос
            StringGrid3->Cells[16][st3]=shirina2;
            StringGrid3->Cells[17][st3]=Form1->StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][i4])];
            StringGrid3->Cells[18][st3]=dlina2/StrToInt(Form1->StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][i4])])/it2; //резов
            StringGrid3->Cells[19][st3]=dlina2/StrToInt(Form1->StringGrid1->Cells[2][StrToInt(StringGrid4->Cells[1][i4])]); //штук
            StringGrid3->Cells[20][st3]=Form1->StringGrid1->Cells[8][StrToInt(StringGrid4->Cells[1][i4])]; //car
```

```
                StringGrid3->Cells[21][st3]=dlina2/1000; //метров
                StringGrid3->Cells[22][st3]=format%(shirina1+shirina2);
                StringGrid3->Cells[23][st3]=Form1->StringGrid1-
>Cells[9][StrToInt(StringGrid4->Cells[1][i4])];
                StringGrid3->Cells[24][st3]=StringGrid4->Cells[4][i4];
                StringGrid3->RowCount=StringGrid3->RowCount+1;
                StringGrid4->Cells[2][1]=(dlina1-dlina2)/
                StrToInt(Form1->StringGrid1->Cells[2][StrToInt(StringGrid4-
>Cells[1][1])]);
                StringGrid4->Cells[3][1]=dlina1-dlina2;
                StringGrid4->Cells[2][i4]=0;
                StringGrid4->Cells[3][i4]=0;
                int dlina=StrToInt(StringGrid4->Cells[3][1]);
              for (int i0=2; i0<StringGrid4->RowCount-1; i0++)
              {
               if (dlina<StrToInt(StringGrid4->Cells[3][i0]))
               {
                dlina=StrToInt(StringGrid4->Cells[3][i0]);
                StringGrid4->Cells[0][StringGrid4->RowCount]=StringGrid4-
>Cells[0][1];
                StringGrid4->Cells[1][StringGrid4->RowCount]=StringGrid4-
>Cells[1][1];
                StringGrid4->Cells[2][StringGrid4->RowCount]=StringGrid4-
>Cells[2][1];
                StringGrid4->Cells[3][StringGrid4->RowCount]=StringGrid4-
>Cells[3][1];
                StringGrid4->Cells[0][1]=StringGrid4->Cells[0][i0];
                StringGrid4->Cells[1][1]=StringGrid4->Cells[1][i0];
                StringGrid4->Cells[2][1]=StringGrid4->Cells[2][i0];
                StringGrid4->Cells[3][1]=StringGrid4->Cells[3][i0];
                StringGrid4->Cells[0][i0]=StringGrid4->Cells[0][StringGrid4-
>RowCount];
                StringGrid4->Cells[1][i0]=StringGrid4->Cells[1][StringGrid4-
>RowCount];
                StringGrid4->Cells[2][i0]=StringGrid4->Cells[2][StringGrid4-
>RowCount];
```

```
                StringGrid4->Cells[3][i0]=StringGrid4->Cells[3][StringGrid4-
>RowCount];
                StringGrid4->Rows[StringGrid4->RowCount]->Clear();
                }
               }
              } // if delenie
             }  //for it2
            }  // for it1
           }  //if >0
          }  //for ii
         }  //for i4
          for (int d=1;d<=StringGrid4->RowCount;d++)
          {
           if(!(StringGrid4->Cells[2][d].IsEmpty()))
           {
            if(StrToInt(StringGrid4->Cells[2][d])>0)
            {
            int st5=StringGrid5->RowCount-1;
            StringGrid5->Cells[0][st5]=StringGrid4->Cells[0][d];
            StringGrid5->Cells[2][st5]=StringGrid4->Cells[2][d];
            StringGrid5->Cells[3][st5]=Form1->StringGrid1-
>Cells[9][StrToInt(StringGrid4->Cells[1][d])];
            StringGrid5->Cells[1][st5]=Form1->StringGrid1-
>Cells[1][StrToInt(StringGrid4->Cells[1][d])];
            StringGrid5->Cells[4][st5]=StringGrid4->Cells[1][d];
            StringGrid5->RowCount=StringGrid5->RowCount+1;
            }
           }
          }
         } //if главный
         //---------------------------------------------------------------------
         else
          {
          int  shir  =StrToInt(Form1->StringGrid1->Cells[3][StrToInt(StringGrid2-
>Cells[4][1])]);

          for (int i=1; i<StringGrid1->RowCount-1; i++)
```

```
            {
            int format=StrToInt(StringGrid1->Cells[0][i]);
            if (format%shir<max && format%shir>min && zapis==false)
             {
              int id=StrToInt(StringGrid2->Cells[4][1]);
              int id2=StrToInt(Form1->StringGrid1->Cells[10][id]);
              st3=StringGrid3->RowCount-1;
              StringGrid3->Cells[0][st3]=st3;
              StringGrid3->Cells[1][st3]=StringGrid2->Cells[1][1];
              StringGrid3->Cells[2][st3]=format/shir;
              StringGrid3->Cells[3][st3]=format;
              StringGrid3->Cells[4][st3]=shir;
              StringGrid3->Cells[5][st3]=Form1->StringGrid1->Cells[2][id];
              StringGrid3->Cells[6][st3]=StrToInt(StringGrid2-
>Cells[2][1])/(format/shir);
              StringGrid3->Cells[7][st3]=StringGrid2->Cells[2][1];
              StringGrid3->Cells[8][st3]=Form1->StringGrid1->Cells[8][id];
              StringGrid3->Cells[9][st3]=Form2->StringGrid1->Cells[4][id2];
              StringGrid3->Cells[10][st3]=Form2->StringGrid1->Cells[4][id2];
              StringGrid3->Cells[11][st3]=Form2->StringGrid1->Cells[5][id2];
              StringGrid3->Cells[12][st3]=Form2->StringGrid1->Cells[6][id2];
              StringGrid3->Cells[13][st3]=Form2->StringGrid1->Cells[7][id2];
              StringGrid3->Cells[21][st3]=StrToInt(Form1->StringGrid1-
>Cells[2][id])*StrToInt(StringGrid2->Cells[2][1])/(format/shir);
              StringGrid3->Cells[22][st3]=format%shir;
              StringGrid3->Cells[23][st3]=StringGrid2->Cells[3][1];
              StringGrid3->RowCount=StringGrid3->RowCount+1;
              for(int l=1;l<StringGrid2->RowCount;l++)            //ispr
              {
               for (int l2=0;l2<StringGrid2->ColCount; l2++)
               {
                StringGrid2->Cells[l2][l]=StringGrid2->Cells[l2][l+1];
               }
              }
              StringGrid2->RowCount=StringGrid2->RowCount-1;
              zapis =true;
              n=n-1;
```

```
   }  //id delenie

  }  //for
if (zapis==false)
 {
  st2=StringGrid5->RowCount-1;
  for (int i=0;i<StringGrid2->ColCount; i++)
   {
   StringGrid5->Cells[i][st2]=StringGrid2->Cells[i][1];
   }
  for(int l=1;l<StringGrid2->RowCount-1;l++)            //ispr
  {
   for (int l2=0;l2<StringGrid2->ColCount; l2++)
    {
    StringGrid2->Cells[l2][l]=StringGrid2->Cells[l2][l+1];
    }
   }
  StringGrid2->RowCount=StringGrid2->RowCount-1;
  StringGrid5->RowCount=StringGrid5->RowCount+1;
  n=n-1;
 }
} //else1
zapis=false;
} //while1
  ShowMessage("The end");
  StringGrid3->Cells[0][0]="Номер";
  StringGrid3->Cells[1][0]="Наименование";
  StringGrid3->Cells[2][0]="Полос";
  StringGrid3->Cells[3][0]="Формат";
  StringGrid3->Cells[4][0]="Ширина";
  StringGrid3->Cells[5][0]="Длина";
  StringGrid3->Cells[6][0]="Резов";
  StringGrid3->Cells[7][0]="Штук";
  StringGrid3->Cells[8][0]="Машина";
  StringGrid3->Cells[9][0]="L";
  StringGrid3->Cells[10][0]="F";
  StringGrid3->Cells[11][0]="L";
```

```
          StringGrid3->Cells[12][0]="F";
          StringGrid3->Cells[13][0]="L";
          StringGrid3->Cells[14][0]="Наименование";
          StringGrid3->Cells[15][0]="Полос";
          StringGrid3->Cells[16][0]="Ширина";
          StringGrid3->Cells[17][0]="Длина";
          StringGrid3->Cells[18][0]="Резов";
          StringGrid3->Cells[19][0]="Штук";
          StringGrid3->Cells[20][0]="Машина";
          StringGrid3->Cells[21][0]="Пог метры";
          StringGrid3->Cells[22][0]="обрезь";
          StringGrid3->Cells[23][0]="Код";
          StringGrid3->Cells[24][0]="id";
         //---------------------------------------------------------------------------
         for (int j=1;j<StringGrid5->RowCount-1;j++)                //контрольная
проверка
         {
         int shir =StrToInt(Form1->StringGrid1->Cells[3][StrToInt(StringGrid5-
>Cells[4][j])]);

          for (int i=1; i<StringGrid1->RowCount-1; i++)
          {
          int format=StrToInt(StringGrid1->Cells[0][i]);
          if (format%shir<max && format%shir>min)
           {
            int id=StrToInt(StringGrid5->Cells[4][j]);
            int id2=StrToInt(Form1->StringGrid1->Cells[10][id]);
            st3=StringGrid3->RowCount-1;
            StringGrid3->Cells[0][st3]=st3;
            StringGrid3->Cells[1][st3]=StringGrid5->Cells[1][j];
            StringGrid3->Cells[2][st3]=format/shir;
            StringGrid3->Cells[3][st3]=format;
            StringGrid3->Cells[4][st3]=shir;
            StringGrid3->Cells[5][st3]=Form1->StringGrid1->Cells[2][id];
            StringGrid3->Cells[6][st3]=StrToInt(StringGrid5-
>Cells[2][j])/format/shir;
            StringGrid3->Cells[7][st3]=StringGrid5->Cells[2][j];
```

```
            StringGrid3->Cells[8][st3]=Form1->StringGrid1->Cells[8][id];
//машина
            StringGrid3->Cells[9][st3]=Form2->StringGrid1->Cells[3][id2];
            StringGrid3->Cells[10][st3]=Form2->StringGrid1->Cells[4][id2];
            StringGrid3->Cells[11][st3]=Form2->StringGrid1->Cells[5][id2];
            StringGrid3->Cells[12][st3]=Form2->StringGrid1->Cells[6][id2];
            StringGrid3->Cells[13][st3]=Form2->StringGrid1->Cells[7][id2];
            StringGrid3->Cells[21][st3]=StrToInt(Form1->StringGrid1-
>Cells[2][id])*StrToInt(StringGrid5->Cells[2][j])/format/shir;
            StringGrid3->Cells[22][st3]=format%shir;
            StringGrid3->Cells[23][st3]=StringGrid5->Cells[3][j];
            StringGrid3->Cells[24][st3]=StringGrid5->Cells[4][j];;
            StringGrid3->RowCount=StringGrid3->RowCount+1;
            StringGrid5->Cells[2][j]=0;
            //break;
           }
          }
         }
         int del=0;
         for(int i1=1; i1<StringGrid5->RowCount-1;i1++)
         {
          if(StrToInt(StringGrid5->Cells[2][i1])==0)
          {
           del=del+1;
           for (int m=0;m<StringGrid5->ColCount; m++)
           {
            StringGrid5->Cells[m][StringGrid5->RowCount]=StringGrid5-
>Cells[m][i1];
           }
           for(int m1=0; m1<StringGrid5->ColCount; m1++)
           {
           StringGrid5->Cells[m1][i1]=StringGrid5->Cells[m1][i1+1];
           }
          }
          }
         StringGrid5->RowCount=StringGrid5->RowCount-del;
         StringGrid2->Rows[1]->Clear();
```

```
 StringGrid3->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::StringGrid5DrawCell(TObject *Sender, int ACol,
     int ARow, TRect &Rect, TGridDrawState State)
{
if (StringGrid5->RowCount>2 && ARow!=StringGrid5->RowCount-1
&& ACol!=StringGrid5->ColCount-1 && ARow!=0)
 {
 if(ARow!=0 && ARow!=StringGrid5->RowCount-1 &&
ACol!=StringGrid5->ColCount-1)
  {
  if (!(StringGrid5->Cells[ACol][ARow].IsEmpty()))
  {
  StringGrid5->Canvas->Brush->Color=clRed;
  StringGrid5->Canvas->FillRect(Rect);
  StringGrid5->Canvas->TextOut(Rect.Left, Rect.Top, StringGrid5-
>Cells[ACol][ARow]);
  }
  }
  for (int j=1; j<StringGrid3->RowCount-1; j++)
  {
  if (StringGrid5->Cells[1][ARow]==StringGrid3->Cells[1][j] ||
StringGrid5->Cells[1][ARow]==StringGrid3->Cells[14][j])
  {
  StringGrid5->Canvas->Brush->Color=clGreen;
  StringGrid5->Canvas->FillRect(Rect);
  StringGrid5->Canvas->TextOut(Rect.Left, Rect.Top, StringGrid5-
>Cells[ACol][ARow]);
  }
  }
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::StringGrid3DragOver(TObject *Sender,
     TObject *Source, int X, int Y, TDragState State, bool &Accept)
{
```

```
    Accept = Source == StringGrid5;
  }
  //---------------------------------------------------------------------------
  void __fastcall TForm3::StringGrid3DragDrop(TObject *Sender,
       TObject *Source, int X, int Y)
  {
   int ACol=1, ARow, del=0;
   StringGrid3->MouseToCell( X, Y, ACol, ARow);
   for (int m=1;m<StringGrid5->RowCount-1;m++)
    {
     if (StringGrid3->Cells[1][ARow]==StringGrid5->Cells[1][m])
     {
      StringGrid3->Cells[1][ARow] = StringGrid5->Cells[1][m];
      StringGrid3->Cells[7][ARow]=StrToInt(StringGrid3-
>Cells[7][ARow])+
      StrToInt(StringGrid5->Cells[2][m]);
      StringGrid3->Cells[6][ARow]=StrToInt(StringGrid3-
>Cells[7][ARow])/StrToInt(StringGrid3->Cells[2][ARow]);
      StringGrid3->Cells[21][ARow]=StrToInt(StringGrid3-
>Cells[5][ARow])*StrToInt(StringGrid3->Cells[6][ARow]);//m
      StringGrid3->Cells[18][ARow]=StrToInt(StringGrid3-
>Cells[21][ARow])/(StrToInt(StringGrid3->Cells[17][ARow])); //rez
      StringGrid3->Cells[19][ARow]=StrToInt(StringGrid3-
>Cells[18][ARow])*StrToInt(StringGrid3->Cells[15][ARow]);// unit
      StringGrid3->Cells[21][ARow]=StrToInt(StringGrid3-
>Cells[21][ARow])/1000;
      del=del+1;
      for (int i=m; i<StringGrid5->RowCount-1;i++)
      {
       for(int j=0;j<StringGrid5->ColCount;j++)
       {
        StringGrid5->Cells[j][i]=StringGrid5->Cells[j][i+1];
       }
      }
     }
     if (StringGrid3->Cells[1][ARow]==StringGrid5->Cells[1][m])
     {
```

```
            StringGrid3->Cells[1][ARow] = StringGrid5->Cells[1][m];
            StringGrid3->Cells[7][ARow]=StrToInt(StringGrid3-
>Cells[7][ARow])+
            StrToInt(StringGrid5->Cells[2][m]);
            StringGrid3->Cells[6][ARow]=StrToInt(StringGrid3-
>Cells[7][ARow])/StrToInt(StringGrid3->Cells[2][ARow]);
            StringGrid3->Cells[21][ARow]=StrToInt(StringGrid3-
>Cells[5][ARow])*StrToInt(StringGrid3->Cells[6][ARow]);//m
            StringGrid3->Cells[18][ARow]=StrToInt(StringGrid3-
>Cells[21][ARow])/(StrToInt(StringGrid3->Cells[17][ARow])); //rez
            StringGrid3->Cells[19][ARow]=StrToInt(StringGrid3-
>Cells[18][ARow])*StrToInt(StringGrid3->Cells[15][ARow]);// unit
            StringGrid3->Cells[21][ARow]=StrToInt(StringGrid3-
>Cells[21][ARow])/1000;
            del=del+1;
            for (int i=m; i<StringGrid5->RowCount-1;i++)
            {
             for(int j=0;j<StringGrid5->ColCount;j++)
             {
              StringGrid5->Cells[j][i]=StringGrid5->Cells[j][i+1];
             }
            }
          }
         StringGrid5->RowCount=StringGrid5->RowCount-del;
        }
        }
        //---------------------------------------------------------------------
        void __fastcall TForm3::CheckBox1Click(TObject *Sender)
        {
        if (CheckBox1->Checked==true)
         {
         Edit2->Enabled=false;
         Edit3->Enabled=false;
         }
         else
         {
         Edit2->Enabled=true;
```

```
 Edit3->Enabled=true;
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::StringGrid3Enter(TObject *Sender)
{
int sum=0,L=0, S=0, kol=StringGrid3->RowCount-2;
if (kol!=0)
{
 for (int i=1;i<StringGrid3->RowCount-1;i++ )
 {
  StringGrid3->Cells[6][i]=StrToInt(StringGrid3-
>Cells[7][i])/StrToInt(StringGrid3->Cells[2][i]);
  StringGrid3->Cells[21][i]=StrToInt(StringGrid3-
>Cells[5][i])*StrToInt(StringGrid3->Cells[6][i])/1000;
  sum=sum+StrToInt(StringGrid3->Cells[22][i]);
  L=L+StrToInt(StringGrid3->Cells[21][i]);
  S=S+((StrToInt(StringGrid3->Cells[3][i])-StrToInt(StringGrid3-
>Cells[22][i]))*StrToInt(StringGrid3->Cells[21][i]))/1000;
 }
 if (sum==0)
 {
  Label7->Caption=Label7->Caption+" "+0; // средняя обрезь
  Label8->Caption=Label8->Caption+" "+0;// площадь обрези
  Label9->Caption=Label9->Caption+" "+S;
  Label10->Caption=Label10->Caption+" "+0;
 }
 else
{
Label7->Caption=Label7->Caption+" "+sum/kol; // средняя обрезь
Label8->Caption=Label8->Caption+" "+sum/kol*L;// площадь обрези
Label9->Caption=Label9->Caption+" "+S;
Label10->Caption=Label10->Caption+" "+(sum/kol*L)/(S*10)+"%";
}
 }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm3::N4Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button4Click(TObject *Sender)
{
if (StringGrid3->RowCount>2)
 {
  int st1=1;
  Form5->Show();
  for (int i=1;i<StringGrid3->RowCount-2;i++)
   {
    for(int i1=2;i1<StringGrid3->RowCount-1;i1++)
     {
      if (StringGrid3->Cells[3][i]==StringGrid3->Cells[3][i1] &&
      StringGrid3->Cells[23][i]==StringGrid3->Cells[23][i1])
      {
      if (StringGrid3->Cells[24][i]!="OK")
      {
       Form5->StringGrid1->RowCount=Form5->StringGrid1-
>RowCount+1;
       AnsiString p=StringGrid3->Cells[23][i];
       Form5->StringGrid1->Cells[0][st1]=p[9];
       Form5->StringGrid1->Cells[2][st1]=StringGrid3->Cells[3][i];
      Form5->StringGrid1->Cells[4][st1]=StringGrid3->Cells[9][i];
      Form5->StringGrid1->Cells[6][st1]=StringGrid3->Cells[10][i];
      Form5->StringGrid1->Cells[8][st1]=StringGrid3->Cells[11][i];
      Form5->StringGrid1->Cells[10][st1]=StringGrid3->Cells[12][i];
      Form5->StringGrid1->Cells[12][st1]=StringGrid3->Cells[13][i];
      st1=st1+1;
      Form5->StringGrid1->Cells[0][st1]=StringGrid3->Cells[1][i];
      Form5->StringGrid1->Cells[1][st1]=StringGrid3->Cells[2][i];
      Form5->StringGrid1->Cells[2][st1]=StringGrid3->Cells[4][i];
      Form5->StringGrid1->Cells[3][st1]=StringGrid3->Cells[5][i];
      for (int j=1;j<Form1->StringGrid1->RowCount;j++)
       {
```

```
                if          (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid3-
>Cells[1][i])
                {
                 Form5->StringGrid1->Cells[4][st1]=Form1->StringGrid1-
>Cells[4][j];
                }
               }
              Form5->StringGrid1->Cells[5][st1]=StringGrid3->Cells[6][i];
              Form5->StringGrid1->Cells[6][st1]=StringGrid3->Cells[7][i];
              Form5->StringGrid1->Cells[7][st1]=StringGrid3->Cells[8][i];
              Form5->StringGrid1->Cells[9][st1]=StringGrid3->Cells[14][i];//name
              Form5->StringGrid1->Cells[10][st1]=StringGrid3->Cells[15][i];
              Form5->StringGrid1->Cells[11][st1]=StringGrid3->Cells[16][i];
              Form5->StringGrid1->Cells[12][st1]=StringGrid3->Cells[17][i];
              for (int j=1;j<Form1->StringGrid1->RowCount;j++)
               {
                if          (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid1-
>Cells[14][i])
                {
                 Form5->StringGrid1->Cells[13][st1]=Form1->StringGrid1-
>Cells[4][j];
                }
               }
              Form5->StringGrid1->Cells[14][st1]=StringGrid3->Cells[18][i];
              Form5->StringGrid1->Cells[15][st1]=StringGrid3->Cells[19][i];
              Form5->StringGrid1->Cells[16][st1]=StringGrid3->Cells[20][i];
              Form5->StringGrid1->Cells[18][st1]=StringGrid3->Cells[21][i];
              Form5->StringGrid1->Cells[19][st1]=StringGrid3->Cells[22][i];
              StringGrid3->Cells[24][i]="OK";
              st1=st1+1;
             }
        //---------------------------------------------------------------------------
              if(StringGrid3->Cells[24][i1]!="OK")
              {
              Form5->StringGrid1->RowCount=Form5->StringGrid1-
>RowCount+1;
              Form5->StringGrid1->Cells[0][st1]=StringGrid3->Cells[1][i1];
```

```
Form5->StringGrid1->Cells[1][st1]=StringGrid3->Cells[2][i1];
Form5->StringGrid1->Cells[2][st1]=StringGrid3->Cells[4][i1];
Form5->StringGrid1->Cells[3][st1]=StringGrid3->Cells[5][i1];
for (int j=1;j<Form1->StringGrid1->RowCount;j++)
 {
  if (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid3->Cells[1][i1])
  {
   Form5->StringGrid1->Cells[4][st1]=Form1->StringGrid1->Cells[4][j];
  }
 }
Form5->StringGrid1->Cells[5][st1]=StringGrid3->Cells[6][i1];
Form5->StringGrid1->Cells[6][st1]=StringGrid3->Cells[7][i1];
Form5->StringGrid1->Cells[7][st1]=StringGrid3->Cells[8][i1];
Form5->StringGrid1->Cells[9][st1]=StringGrid3->Cells[14][i1];//name
Form5->StringGrid1->Cells[10][st1]=StringGrid3->Cells[15][i1];
Form5->StringGrid1->Cells[11][st1]=StringGrid3->Cells[16][i1];
Form5->StringGrid1->Cells[12][st1]=StringGrid3->Cells[17][i1];
for (int j=1;j<Form1->StringGrid1->RowCount;j++)
 {
  if (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid1->Cells[14][i1])
  {
   Form5->StringGrid1->Cells[13][st1]=Form1->StringGrid1->Cells[4][j];
  }
 }
Form5->StringGrid1->Cells[14][st1]=StringGrid3->Cells[18][i1];
Form5->StringGrid1->Cells[15][st1]=StringGrid3->Cells[19][i1];
Form5->StringGrid1->Cells[16][st1]=StringGrid3->Cells[20][i1];
Form5->StringGrid1->Cells[18][st1]=StringGrid3->Cells[21][i1];
Form5->StringGrid1->Cells[19][st1]=StringGrid3->Cells[22][i1];
StringGrid3->Cells[24][i1]="OK";
st1=st1+1;
}
```

```
        } //if
       }
      }
       for(int i=1;i<StringGrid3->RowCount-1;i++)
        {
         if (StringGrid3->Cells[24][i]!="OK")
         {
         AnsiString p=StringGrid3->Cells[23][i];
         Form5->StringGrid1->Cells[0][st1]=p[9];
         Form5->StringGrid1->Cells[2][st1]=StringGrid3->Cells[3][i];
         Form5->StringGrid1->Cells[4][st1]=StringGrid3->Cells[9][i];
         Form5->StringGrid1->Cells[6][st1]=StringGrid3->Cells[10][i];
         Form5->StringGrid1->Cells[8][st1]=StringGrid3->Cells[11][i];
         Form5->StringGrid1->Cells[10][st1]=StringGrid3->Cells[12][i];
         Form5->StringGrid1->Cells[12][st1]=StringGrid3->Cells[13][i];
         st1=st1+1;
         Form5->StringGrid1->Cells[0][st1]=StringGrid3->Cells[1][i];
         Form5->StringGrid1->Cells[1][st1]=StringGrid3->Cells[2][i];
         Form5->StringGrid1->Cells[2][st1]=StringGrid3->Cells[4][i];
         Form5->StringGrid1->Cells[3][st1]=StringGrid3->Cells[5][i];
         for (int j=1;j<Form1->StringGrid1->RowCount;j++)
          {
           if          (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid3-
>Cells[1][i])
            {
             Form5->StringGrid1->Cells[4][st1]=Form1->StringGrid1-
>Cells[4][j];
            }
          }
         Form5->StringGrid1->Cells[5][st1]=StringGrid3->Cells[6][i];
         Form5->StringGrid1->Cells[6][st1]=StringGrid3->Cells[7][i];
         Form5->StringGrid1->Cells[7][st1]=StringGrid3->Cells[8][i];
         Form5->StringGrid1->Cells[9][st1]=StringGrid3->Cells[14][i];//name
         Form5->StringGrid1->Cells[10][st1]=StringGrid3->Cells[15][i];
         Form5->StringGrid1->Cells[11][st1]=StringGrid3->Cells[16][i];
         Form5->StringGrid1->Cells[12][st1]=StringGrid3->Cells[17][i];
         for (int j=1;j<Form1->StringGrid1->RowCount;j++)
```

```
                {
                 if          (Form1->StringGrid1->Cells[1][j]==Form3->StringGrid1-
>Cells[14][i])
                 {
                 Form5->StringGrid1->Cells[13][st1]=Form1->StringGrid1-
>Cells[4][j];
                 }
                }
               Form5->StringGrid1->Cells[14][st1]=StringGrid3->Cells[18][i];
               Form5->StringGrid1->Cells[15][st1]=StringGrid3->Cells[19][i];
               Form5->StringGrid1->Cells[16][st1]=StringGrid3->Cells[20][i];
               Form5->StringGrid1->Cells[18][st1]=StringGrid3->Cells[21][i];
               Form5->StringGrid1->Cells[19][st1]=StringGrid3->Cells[22][i];
               Form5->StringGrid1->RowCount=Form5->StringGrid1-
>RowCount+2;
                st1=st1+1;
              } //if
             }//for
            } //if1
           Form5->StringGrid1->Cells[0][0]="Наименование";
           Form5->StringGrid1->Cells[1][0]="Полос";
           Form5->StringGrid1->Cells[2][0]="Ширина";
           Form5->StringGrid1->Cells[3][0]="Длина";
           Form5->StringGrid1->Cells[4][0]="Биговки";
           Form5->StringGrid1->Cells[5][0]="Резов";
           Form5->StringGrid1->Cells[6][0]="Штук";
           Form5->StringGrid1->Cells[7][0]="Машина";
           Form5->StringGrid1->Cells[8][0]="В стопе";
           Form5->StringGrid1->Cells[9][0]="Наименование";
           Form5->StringGrid1->Cells[10][0]="Полос";
           Form5->StringGrid1->Cells[11][0]="Ширина";
           Form5->StringGrid1->Cells[12][0]="Длина";
           Form5->StringGrid1->Cells[13][0]="Биговки";
           Form5->StringGrid1->Cells[14][0]="Резов";
           Form5->StringGrid1->Cells[15][0]="Штук";
           Form5->StringGrid1->Cells[16][0]="Машина";
           Form5->StringGrid1->Cells[18][0]="Пог.м";
```

```
 Form5->StringGrid1->Cells[19][0]="Обрезь";
}
//---------------------------------------------------------------------------
```

Листинг файла Unit4.~cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit3.h"
#include "ProgPlan_1.h"
#include <FileCtrl.hpp>
#include <string.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
TStringList *hand = new TStringList ;
String Path;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormActivate(TObject *Sender)
{
int sum=0,L=0, S=0, IS=0, P2=0,P3=0,P5=0, AS=0,W=0,V=0;
int m1=8,m2=7;

 for (int i=1;i<Form3->StringGrid3->RowCount-1;i++ )
 {
  for (int m=0;m<2;m++)
   {
  if (Form3->StringGrid3->Cells[m1][i]=="IS")
  {
   IS=IS+StrToInt(Form3->StringGrid3->Cells[m2][i]);
```

```
            }
            if (Form3->StringGrid3->Cells[m1][i]=="9PA-2")
            {
             P2=P2+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
              if (Form3->StringGrid3->Cells[m1][i]=="9PA-3")
            {
             P3=P3+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
             if (Form3->StringGrid3->Cells[m1][i]=="5PA-2")
            {
             P5=P5+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
             if (Form3->StringGrid3->Cells[m1][i]=="AS")
            {
             AS=AS+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
             if (Form3->StringGrid3->Cells[m1][i]=="W")
            {
             W=W+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
             if (Form3->StringGrid3->Cells[m1][i]=="V")
            {
             V=V+StrToInt(Form3->StringGrid3->Cells[m2][i]);
            }
            m1=20;
            m2=19;
           }
          }
           for (int i1=1;i1<Form3->StringGrid3->RowCount-1;i1++ )
           {
            Form3->StringGrid3->Cells[6][i1]=StrToInt(Form3->StringGrid3-
>Cells[7][i1])/StrToInt(Form3->StringGrid3->Cells[2][i1]);
            Form3->StringGrid3->Cells[21][i1]=StrToInt(Form3->StringGrid3-
>Cells[5][i1])*StrToInt(Form3->StringGrid3->Cells[6][i1])/1000;
            sum=sum+StrToInt(Form3->StringGrid3->Cells[22][i1]);
            L=L+StrToInt(Form3->StringGrid3->Cells[21][i1]);
```

```
        S=S+((StrToInt(Form3->StringGrid3->Cells[3][i1])-StrToInt(Form3-
>StringGrid3->Cells[22][i1]))*StrToInt(Form3->StringGrid3-
>Cells[21][i1]))/1000;
        }
      ProgressBar1->Position=S*100/StrToInt(Edit1->Text);
      Label10->Caption=S*100/StrToInt(Edit1->Text);
      ProgressBar5->Position=IS*100/StrToInt(Edit5->Text);
      Label14->Caption=IS*100/StrToInt(Edit5->Text);
      ProgressBar2->Position=P2*100/StrToInt(Edit2->Text);
      Label11->Caption=P2*100/StrToInt(Edit2->Text);
      ProgressBar3->Position=P3*100/StrToInt(Edit3->Text);
      Label12->Caption=P3*100/StrToInt(Edit3->Text);
      ProgressBar4->Position=P5*100/StrToInt(Edit4->Text);
      Label13->Caption=P5*100/StrToInt(Edit4->Text);
      ProgressBar6->Position=AS*100/StrToInt(Edit6->Text);
      Label15->Caption=AS*100/StrToInt(Edit6->Text);
      ProgressBar7->Position=W*100/StrToInt(Edit7->Text);
      Label16->Caption=W*100/StrToInt(Edit7->Text);
      ProgressBar8->Position=V*100/StrToInt(Edit8->Text);
      Label17->Caption=V*100/StrToInt(Edit8->Text);
      m1=8;
      }
      //---------------------------------------------------------------------------
```