Федеральное агентство по образованию РФ

ГОУ ВПО «ИЖЕВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

Кафедра «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломной работе на тему:

«Конвертирование исходного текста программ для станков с ЧПУ из одной системы программирования в другую»

Дипломник

студент группы 12-19-4з……………………………………….Д.В. Стерхов

Руководитель

вед. инженер - программист

ОАО «Концерн ИЖМАШ»……………………………………В.М. Токарев

Консультант по экономической части
к.э.н., доцент…………………………………………………… И.И. Радыгина

Консультант по безопасности

и экологичности к.т.н., доцент……………………………… А.С. Чаузов

Нормоконтроль…………………………………………………В.П. Соболева

Рецензент

вед. инженер - программист

ОАО «Ижмаш - Станко»………………………………………..И.П. Буторин

Заведующий кафедрой

«Программное обеспечение»

д.т.н, профессор…………………………………………….......А.И. Мурынов

ИЖЕВСК 2008

Реферат

Пояснительная записка к дипломной работе на тему «Конвертирование исходного текста программ для станков с ЧПУ из одной системы программирования в другую» оформлена на 100 листах, содержит 38 рисунков, 19 таблиц.

 Целью данной работы является разработка системы, которая бы повышала коэффициент полезного действия программистов по станкам с числовым и программным управлением (ЧПУ), которое заключается в автоматизированном переводе исходного текста программ станков с ЧПУ в различные системы программирования.

 Для написания соответствующего программного обеспечения были досконально изучены материалы по программированию станков с ЧПУ различных систем, в частности: Маяк, Синумерик, Вм.

 В результате проделанной работы было разработано программное обеспечение для анализа программ их автоматического пересчёта и конвертирования, а так же исправление основных ошибок в тексте, последовательной нумерации кадров, удаление пустых строк и пробелов.

 На сегодняшний день существует большое количество программных продуктов, предоставляющих возможность конвертирования программного текста. Однако эти программы сложны в настройке и эксплуатации, либо стоимость их непомерно велика.

 Разработанное программное обеспечение постоянно находится в разработке, добавляются новые возможности конвертирования, а так же по мере использования программы устраняются чисто человеческие ошибки программиста.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ, СИМВОЛОВ И СПЕЦИАЛЬНЫХ ТЕРМИНОВ С ИХ ОПРЕДЕЛЕНИЯМИ

ЧПУ – станки с числовым и программным управлением

ПО – программное обеспечение

ОС – операционная система

ЭВМ – электронно – вычислительная машина

ВТ – вычислительная техника

# ВВЕДЕНИЕ

Научно-технический прогресс - это непрерывный процесс открытия новых знаний и применения их в общественном производстве, позволяющий по-новому соединять и комбинировать имеющиеся ресурсы в интересах увеличения выпуска высококачественных конечных продуктов при наименьших затратах. От научно-технического потенциала страны во многом зависит и научно-производственный потенциал ее национальных фирм и предприятий, их способность обеспечивать высокий уровень и темпы НТО, их "выживаемость" в условиях конкурентной борьбы. Научно-технический потенциал страны создается как усилиями национальных научно-технических организаций, так и использованием мировых достижений науки и техники. Анализ и оценка этого потенциала позволяет сделать выводы об уровне экономического развития страны и ее отраслей, о степени ее научно-технической самостоятельности, о возможностях ее экономического и научно-технического сотрудничества.

В наше время производсво не может обойтись без станков с ЧПУ. Хотя они дороги и сложны в эксплуатации, но не заменимы в технологии производства на современных предприятиях.

 На производстве очень много станков ЧПУ различных модификаций и возможностей. На нашем производстве используется 3 системы программирования ВМ, МАЯК и Синумерик. Используется общеизвестная программа **Unigraphics –** это графический редактор, в котором отрисовывается деталь, который затем генерирует (текст программы) для системы Синумерик. Программа **Наладчик** регенерирует код из системы Синумерик на МАЯК или ВМ. Ни какой графики в написанной программе нет. Графических редакторов много и в написании очередного нет необходимости.

 В начале программа задумывалась, как программа для работы только в условиях конкретного производства, но по мере роста, как объёма программы, так и её функциональности решено написать документацию к ней и выпустить на просторы Интернета.

Программа называется **Наладчик 1.6.1**, на сегодняшний день это последняя полностью протестированная и работоспособная версия программы.

# 1. РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО КОН ВЕРТИРОВАНИЯ ИСХОДНОГО ТЕКСТА ПРОГРАММ ДЛЯ СТАНКОВ С ЧПУ

## 1.1. Обоснование целесообразности разработки системы автоматического конвертирования текста исходных программ для станков с ЧПУ

###  1.1.1. Назначение системы

 Система автоматического конвертирования предназначена для обработки текстовых программ для станков с ЧПУ их последующего пересчёта в различные системы программирования. В результате обработки полностью или частично исключается правка программы вручную, уменьшается до минимума человеческий фактор (ошибки) в написании программ.

### 1.1.2. Характеристика функциональной структуры системы

 Обработка текста состоит из следующих методов:

- подсистема выбора метода конвертирования. Позволяет задать один из алгоритмов метода пересчёта в зависимости от выбора программиста.

- подсистема оптимизации разработана для удаления пустых строк, лишних пробелов и поочерёдной нумерации кадров в тексте программы, который уменьшает размер исходного текста и увеличивает его читаемость.

- в подсистеме анализа работает алгоритм автоматического исправления ошибок функций захода G41 и G42, что способствует правильной отработке станком заходов, как на радиус, так и на линейную координату, а так же пересчёт радиусов по функциям G2, G3 (круговая интерполяция).

- подсистема конвертирования реализует пересчет координат с заданным алгоритмом обработки.

###  1.1.3. Обоснование цели создания системы

 Задачи конвертирования текста программ для станков с ЧПУ имеют широкий спектр, по крайней мере, на Российских предприятиях. Сейчас очень много различных программ для станков ЧПУ, в том числе и графических. В которых при построении графического образа детали программа выдаёт текст для станков ЧПУ. Однако это ПО не российского производства и рассчитано для станков последнего поколения. Однако, как водица в России, оборудование в основном на предприятиях старое, но вполне рабочее. В связи с этим возникла ситуация когда программист, сгенерировав программный текст в графических редакторах, пересчитывал его вручную на наши отечественные станки старого производства. Это занимало очень много времени и возрастала вероятность чисто «человеческих» ошибок.

 Целью данной работы явилась разработка, внедрение и сопровождение

такой программы, которая могла бы заменить рутинный труд программиста по пересчёту исходного текста и исправления ошибок.

###  1.1.4. Обоснование состава автоматизируемых задач

 Реализация программы «Наладчик» позволит интегрировать в едином интерфейсе все этапы обработки исходного текста:

 - конвертирование исходного текста программ;

- исправление ошибок программирования;

- приведение текста к более читабельному виду;

- уменьшение размера программы.

#### Функциональная схема программы конвертирования «Наладчик»

 Вспомогательные настройки обработки

Основные методы конвертирования

Диалог выбора

Выбор файла для открытия

Диалог выбора

Разделение файла на кол – во частей

Диалог выбора

Обработка заданными методами(методом)

визуальный контроль выполнения

Обработка разделение файла

Сохранение файла

Диалог выбора

Сохранение файлов

Рис. 1

## 1.2. Аналитический обзор

 Текстовый редактор TechEdit 2.1.1 относится к разряду специального ПО. С его помощью можно автоматизировать процесс разработки программ для ЧПУ. C помощью TechEdit можно преобразовать историю команд Autocad (история команд, содержащая координаты выделенных точек) в черновой вариант программы для ЧПУ, а затем с помощью функций TechEdit провести преобразования координат и их значений, чистку программы от ненужных символов и автоматическую нумерацию кадров. TechEdit забирает на себя многие рутинные повторяющиеся операции с текстом программы, хотя конечно, к сожалению, не избавляет разработчика от сложного интеллектуального труда. TNR TechEdit успешно используется на Стахановском вагоностроительном заводе уже в течении 2-х лет.

Рис.1.1

Программа TechEdit 2.1.1 по сути является текстовым редактором и не имеет функций конвертирования. /<http://tnr.kulichki.com/techedit/te.html>/

**Графический редактор T-FLEX**

Программа, предназначенная для создания управляющих программ (УП) на оборудование с числовым программным управлением (ЧПУ). Программа **T-FLEX ЧПУ** поддерживает различные типы систем управления 2D, 2,5D, 3D и 5D и разделена на две независимые системы [T-FLEX ЧПУ 2D](http://www.tflex.ru/products/uprav/chpu2d2.php) и [T-FLEX ЧПУ 3D](http://www.tflex.ru/products/uprav/chpu3d2.php).

 Программа подготовки УП для станков с ЧПУ **T-FLEX ЧПУ** является встраиваемым модулем для САПР **T-FLEX CAD** и фукционирует исключительно совместно с ней. Таким образом, получается полноценное CAD/CAM-решение [/www.tflex.ru/.](http://www.tflex.ru)

Данный подход обеспечивает:

- полную ассоциативность конструкторско-технологических данных (однажды созданная траектория обработки будет автоматически перестраиваться после изменения геометрии детали);

- единство интерфейсов (пользователь, знакомый с принципами работы в **T-FLEX CAD**, без труда овладеет программой подготовки УП для станков с ЧПУ **T-FLEX ЧПУ**);

- всё богатство инструментария конструкторской системы для доработки технологом приходящей информации под свои нужды (ведь не секрет, что геометрию детали приходится каким-либо образом дорабатывать, например, пересчитать геометрию детали в середину поля допуска, а конструкторская система может сделать это автоматически).

Программа подготовки УП для станков с ЧПУ T-FLEX ЧПУ – гибко настраиваемая система, построенная по модульному принципу (к базовому модулю можно подключать любой набор методов обработки).

Программа многофункциональна и сложна в эксплуатации.

## 1.3. Основные требования к системе

### 1.3.1. Основные цели создания системы и критерии эффективности её функционирования

 Создание программы конвертирования текста позволяет автоматизировать процесс пересчёта программ для станков с ЧПУ в различные системы программирования в любой последовательности. Удобный и простой интерфейс, а так же необходимый минимум настроек обеспечивают полное взаимопонимание «человек – компьютер».

### 1.3.2. Функциональное назначение системы

 Автоматизация процесса конвертирования исходного текста программ для станков с ЧПУ подразумевает реализацию в системе определённых средств и функций. Следует выделить ряд функциональных особенностей, которыми обладает программа:

- возможность автоматического и ручного редактирования;

- гибкие настройки программы;

- возможность выбора режима пересчёта;

- расчёт, как относительных, так и абсолютных размеров;

- автоматическое исправление ошибок в тексте;

- выбор цвета и размера шрифта в процессе редактирования;

- возможность печати содержимого поля редактирования;

- корректировка захода инструмента на эквидистанту контура детали;

- режим поиска и замены;

- преобразование к верхнему регистру всего текста программы.

### 1.3.3. Особенности программы и условия её эксплуатации

 При работе с программой «Наладчик» ни каких особых требований не выдвигается. Файлы, обрабатываемые программой, должны быть в текстовом формате. Файлы могут быть любого объёма – это зависит только от оперативной памяти компьютера и поддерживаемой памятью программы блокнот.

### 1.3.4. Требования к функциональной структуре

Построение программы конвертирования исходного текста программ для станков с ЧПУ предполагает модульную структуру. Общий интерфейс и возможность доступа ко всем модулям в составе системы обеспечивает оболочка. Из оболочки программы вызываются следующие модули: подсистема выбора метода конвертирования, подсистема оптимизации, подсистема анализа, подсистема конвертирования.

Подсистема выбора метода конвертирования служит для выбора системы программирования, в которую будет, осуществятся пересчёт. Подсистема располагает графическим интерфейсом выбора. Результатом является выбранная система пересчёта.

Подсистема оптимизации текста программы выполняет функции: удаление пробелов (для уменьшения размера программы), нумерации кадров и удаление пустых строк (для лучшей читабельности). Результатом является оптимизированный программный текст.

Подсистема конвертирования преобразует исходные координаты к заданным. Результатом работы являются преобразованные данные.

Подсистема анализа реализует перерасчёт и проверку правильности заданных координат при заходе фрезы на эквидистанту контура обрабатываемой детали, а так же проверку и при необходимости пересчёт заданной круговой интерполяции (G2, G3). Результатом является полностью готовый и преобразованный программный текст для станков с ЧПУ.

### 1.3.5. Требования к техническому обеспечению

 Программа обработки исходного текста не требует особых ресурсов от компьютера. Программный продукт работает уже не первый год и доведена практически до идеала по скорости обработки и затрачиваемым компьютером ресурсам. Чем выше частота процессора, тем быстрее происходит конвертирование программного текста. Требования приведены в таблице 1.

 Таблица 1

Технические характиристики персонального компьютера

|  |  |
| --- | --- |
| Вид устройства | Характеристика |
| процессор | 486 и выше |
| оперативная память | от 32 мб и выше |
| разрешение экрана | 800х600 и более |
| операционная система | Windows 98, XP, Vista |

### 1.3.6. Требования к информационному обеспечению

Программа конвертирования предназначена только для работы с файлами в текстовом формате. В другом формате работы программа не тестировалась, и могут быть непредсказуемые результаты вплоть до зависания компьютера. Программный продукт ориентирован на опытного человека в области программирования станков с ЧПУ (наладчика либо программиста). Ввод заведомо не верных данных (координат (букв) не предусмотренных в программировании станков) приведёт к игнорированию программой этих значений.

### 1.3.7 Требования к программному обеспечению

Программу целесообразно использовать для функционирования под операционной системой семейства Windows, так как ОС данного класса наиболее широко распространены в современном мире. Платформой для разработки выбрано Delphi7. Эта среда поддерживает алгоритмический язык Object Pascal и обладают при этом возможностями быстрой разработки и проектирования визуальных интерфейсов, что особенно важно при работе с любой информацией.

## 1.4. Основные технические решения проекта системы

### 1.4.1. Решение по комплексу технических средств

Как уже отмечалось в п. 1.3.5 «Требования к техническому обеспечению», для достижения удобного пользователю режима функционирования системы необходимо следующая минимальная конфигурация персонального компьютера: частота процессора 486 МГц, объем оперативной памяти 32 Мб, монитор, поддерживающий разрешение 800х600 точек. Также желательно наличие следующих периферийных технических средств: чёрно – белый лазерный принтер для вывода на печать результатов обработки текста.

### 1.4.2. Описание системы программного обеспечения

Для реализации и функционирования проекта необходимо следующее программное обеспечение: ОС Windows 98, XP, Vista в основе, которой лежит ядро, характеризуемое 32-разрядной вычислительной архитектурой и полностью защищенной моделью памяти, что обеспечивает надежную вычислительную среду.

Разработка программы «Наладчик» и ее подсистем будет вестись с использованием среды для разработки приложений Borland Delphi 7. Среды разработки включают в себя высокопроизводительный 32-битный компилятор, что позволяет оптимизировать создаваемый код. В состав каждой среды включен обширный набор средств, которые повышают производительность труда программистов и сокращают продолжительность цикла разработки. Удобство разработки и эффективность созданных в данных средах разработки программ делают их оптимальным выбором для построения конвертирующей системы, какой является программа «Наладчик».

# 2. РАЗРАБОТКА ЗАДАЧИ «Конвертирование исходного текста программ для станков с ЧПУ»

## 2.1. Описание постановки задачи

### 2.1.1. Характеристика задачи

 Стоит отметить, что текст программы должен быть написан синтаксически правильно. Программа, конечно исправит некоторые не правильно написанные координаты (их значение), но писавший должен быть профессионал в своём деле и допускать, как можно меньше ошибок программирования.

 Для уменьшения размера обрабатываемой текстовой программы были ведены такие функции как: удаление пустых строк, пробелов и для наглядности упорядоченная нумерация кадров.

Таким образом, задача обработки текста программ для станков с ЧПУ может быть разбита на несколько подзадач:

- разработка алгоритмов реализующих конвертирование исходных текстов программ;

- разработка алгоритма исправления ошибок программирования;

- разработка и реализация алгоритмов способствующих уменьшению объёма программы и одновременно увеличение читабельности текста.

### 2.1.2. Входная информация

 Входной информацией для задачи является файл \*.тхт полученный после обработки, либо в графической программе, либо набранный в ручную.

### 2.1.3. Выходная информация

 Выходной информацией задачи является файл \*.тхт обработанный заданным методом обработки.

### 2.1.4. Математическая простановка задачи

 Следует отметить, что вся машинная арифметика основывается на декартовой системе координат, где X, Y, Z – основные программные координаты, I, J, K – дополнительные (при расчёте круговой интерполяции и заходов на эквидистанту обрабатываемого контура). Координата A – поворотное устройство.

X f

Y

Z

Рис. 2.1

#### 2.1.4.1. Математическая постановка задачи ПК – Маяк

**Расчёт радиусов по G2, G3 (круговой интерполяции)**

Эта задача предназначена для пересчёта круговой интерполяции и заходов на эквидистанту контура детали.

Алгоритм расчёта радиусов основан на расстоянии удаления точки центра радиуса от начальной точки радиуса (в относительной системе исчисления) и от ноля детали до центра радиуса (в абсолютной системе исчисления).

На рис. 2.2 изображен эскиз обрабатываемой детали (контура).

 

 Рис. 2.2

Линии a и b отображают разницу расчётов радиусов при программировании системы Вм и Маяк.

Линия со стрелкой ***a*** система Маяк (абсолютная система).

Линия со стрелкой ***b*** система Вм (относительная система).

Формула пересчёта радиусов.

I – значение I координаты в обрабатываемом кадре.

J – значение J координаты в обрабатываемом кадре.

X\_pred – значение координаты X в предыдущем кадре.

Y\_pred – значение координаты Y в предыдущем кадре.

**I = X\_pred + I;**

**J = Y\_pred + J**

Результаты приведены в таблице 2.

Таблица 2

Результаты обработки

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| N1 G0 X0 Y0N1 X10 Y10 F100N1 G42 D30 X17.59 Y18.83N1 G1 X47.59N1 G3 X47.59 Y40.83 I0 J11N1 G1 X17.59 | N1G0X0Y0N2X10.Y10.F100N3G42D30X17.59Y18.83I30.J0N4G1X47.59N5G3X47.59Y40.83I47.59J29.83N6G1X17.59 |

 Как мы видим, нумерация кадров в обработанной программе восстановилась, лишние пробелы удалены.

В кадре N2 расставлены пропущенные точки (таков синтаксис программы обработки для станков с ЧПУ).

В кадре N3 рассчитаны и добавлены недостающие координаты захода.

В кадре N5 пересчитаны координаты I и J в соответствии с заданным методом обработки.

**Расчёт линейных заходов на эквидистанту контура детали по G41 G42** рис 2.3.

 ****

Рис 2.3

Точка захода находится на линейном участке контура.

Находится по формуле:

I – значение I координаты в обрабатываемом кадре.

J – значение J координаты в обрабатываемом кадре.

X\_sled – значение Х в следующем кадре.

X\_nast – значение Х в настоящем кадре.

Y\_sled - значение Y в следующем кадре.

Y\_nast – значение Y в настоящем кадре.

**I = X\_sled – X\_nast;**

**J = Y\_sled – Y\_nast**

Результаты приведены в таблице 3.

 Таблица 3

Результаты обработки

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| N1G0X0Y0N2X40Y10N3G42D30X47.59Y18.83N4G3X47.59Y40.83I0J11N1X17.59N1Y18.83N1X47.59 | N1G0X0Y0N2X40.Y10.N3G42D30X47.59Y18.83I29.83J47.59N4G3X47.59Y40.83I47.59J29.83N5X17.59N6Y18.83N7X47.59 |

Как видно из таблицы кадры №2, 3, 4 изменились в результате обработки

 **Расчёт заходов по радиусу на эквидистанту контура детали по G41 G42**

 

Рис 2.4

Расчёт захода на радиус несколько отличается от захода на линейный размер. Здесь немало важную роль играет вектор направления в зависимости от четверти, на которую направлен вектор рис.2.5.

Формулы:

J\_nast – значение J в обрабатываемом кадре.

J\_sled – значение J в следующем кадре.

I\_nast – значение I в настоящем кадре.

I\_sled – значение I в следующем кадре.

Y\_nast – значение Y в настоящем кадре.

X\_nast – значение X в настоящем кадре.

I и J – служат для определения знака в выражении.

С начала рассчитываем значение (I и J) из следующего кадра.

**Логическое решение задачи**

Если G3, то проверяем значение заданных координат:

**Если I\_sled > 0 и J\_sled >= 0, то I = 1, J = -1;**

**Если I\_sled < 0 и J\_sled <= 0, то I = -1, J = 1;**

**Если I\_sled < 0 и J\_sled >= 0, то I = 1, J = 1;**

**Если I\_sled > 0 и J\_sled <= 0, то I = -1, J = -1**

Если G2, то проверяем значение заданных координат:

**Если I\_sled > 0 и J\_sled >= 0, то I = -1, J = 1;**

**Если I\_sled < 0 и J\_sled <= 0, то I = 1, J = -1;**

**Если I\_sled < 0 и J\_sled >= 0, то I = -1, J = -1;**

**Если I\_sled > 0 и J\_sled <= 0, то I = 1, J = 1**

 Следующее операция описана формулами:

**I\_nast = Y\_nast + J\_sled**

**J\_nast = X\_nast + I\_sled**

 Заключительные вычисления:

**если I = -1 и J > 0, то J\_nast:= -J\_nast;**

**если I = 1 и J < 0, то J\_nast:= -J\_nast;**

**если J = -1 и I > 0, то I\_nast:= -I\_nast;**

**если J = 1 и I < 0, то I\_nast:= -I\_nast**



Рис 2.5

## 2.2. Описание алгоритма метода выбора обработки

### 2.2.1. Назначение и характеристика алгоритма

Алгоритм описывает метод выбора обработки пользователем в зависимости от назначения конечного результата.

### 2.2.2. Используемая информация

В алгоритме используется графическая оболочка метода выбора обработки.

### 2.2.3. Результаты решения

В результате реализации алгоритма программа настраивается для обработки заданным методом.

### 2.2.4. Алгоритм решения

*1. Если Form2.PC\_MAYAK.Checked, то переход к п. 2*

 *2. pc\_mayak(bufer);*

 *3. Если Form2.PC\_VM.Checked, то переход к п. 4*

 *4. pc\_vm(bufer);*

 *5. Если Form2.VM\_MAYAK.Checked, то переход к п. 6*

 *6. vm\_mayak(bufer);*

 *7. Если Form2.MAYAK\_VM.Checked, то переход к п.8*

 *8. mayak\_vm(bufer);*

 *9. Bufer:= '';*

 *10. Inc(m);*

 *11. конец*

### 2.2.5. Список условных обозначений

Условные обозначения, используемые в описании алгоритма, приведены в табл. 4.

Таблица 4

Условные обозначения

|  |  |
| --- | --- |
| Обозначение | Расшифровка |
| bufer | Строка значение процедуры  |
| pc\_mayak(bufer); | Процедура пересчёта заданным методом |
| pc\_vm(bufer); | Процедура пересчёта заданным методом |
| vm\_mayak(bufer); | Процедура пересчёта заданным методом |
| mayak\_vm(bufer); | Процедура пересчёта заданным методом |
| Form2.MAYAK\_VM.Checked | Свойства переключателя |
| Bufer  | Переменная хранения строки |
| Inc() | Функция увеличения |

#### Схема алгоритма метода выбора обработки

Начало

If DelProbel

DelProbel(bufer);

pc\_mayak(bufer);

pc\_vm(bufer);

nomerkadra(bufer);

vm\_mayak(bufer);

Mayak\_vm(bufer);

If pc\_mayak

If pc\_vm

If nomerkadra

If vm\_mayak

If Mayak\_vm

*Bufer:= ''; Inc(m);*

Конец файла

Конец

Рис 2.6

## 2.3 Описание алгоритма выбора вспомогательных методов обработки

### 2.3.1. Назначение и характеристика алгоритма

Алгоритм описывает дополнительные (вспомогательные) методы выбора обработки данных.В результате выбора подключаются дополнительные возможности (удаление пробелов, пустых строк и последовательная нумерация кадров).

### 2.3.2. Используемая информация

В алгоритме используется графическая оболочка метода выбора обработки.

### 2.3.3. Результаты решения

В результате реализации алгоритма программа настраивается для обработки дополнительными методами.

### 2.3.4. Алгоритм решения

*1. Если Form1.Probel.Checked, переходим к п. 2*

 *2. DelProbel(bufer);*

 *3. Если Form1.NomerKadra.Checked, то переходим к п. 4*

 *4. nomerkadra(bufer);*

 *5. Если Form1.PustoStr.Checked, то переход к п. 6*

 *6. pustostroka;*

 *7. Bufer:= '';*

 *8. Inc(m);*

 *9. конец*

 Таблица 5

Список условных обозначений

|  |  |
| --- | --- |
| Название | Описание |
| Bufer | Переменная хранения строки |
| Inc() | Функция увеличение значения |
| DelProbel(bufer); | Функция удаления пробелов  |
| nomerkadra(bufer); | Функция восстановления нумерации кадров |
| pustostroka; | Функция удаления пустых строк |

## 2.4. Описание алгоритма конвертирования

### 2.4.1. Назначение и характеристика алгоритма

Алгоритм описывает конвертирование исходного текста программы для станков с ЧПУ. В результате программа пересчитывает все исходные координаты из данной системы программирования в заданную.

### 2.4.2. Алгоритм решения

 *1. Если (pos('%', kadr) <> 0) or (pos('$', kadr) <> 0), то переход к п. 28*

 *2. Если pos('X',kadr) <> 0, то переход к п. 3 иначе к п. 5*

 *3. NaytiKoordinatu(kadr,'X', vyh);*

 *4. PreobrazovanieMayak(X\_nast, kadr);*

 *5. uslovie:= FindVstavki('X', kadr);*

 *6. DobavitTochku(s);*

 *7. insert(Concat('X', s), kadr, uslovie);*

 *8. Если (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0), то пер к п. 9*

 *9. Если pos('X', kadr) = 0, то переход к п. 10*

 *10. uslovie:= FindVstavki('X', kadr);*

 *11. Insert('X' + s, kadr, uslovie);*

 *12. Если (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0), то переход к п. 13 иначе к п. 16*

 *13. Naytikoordinatu(Stroka.Strings[Succ(m)], 'I', st);*

 *14. Naytikoordinatu(Stroka.Strings[Succ(m)], 'J', st);*

 *15. Podhod\_G41\_G42\_radius;*

 *16. Podhod\_G41\_G42\_line;*

 *17. Если pos('I', kadr) <> 0, то переход к п. 18*

 *18. NaytiKoordinatu(kadr, 'I', st);*

 *19. PreobrazovanieMayak(X\_sled, kadr);*

 *20. uslovie:= pos('I',kadr);*

 *21. Если uslovie <> 0, то переход к п. 22*

 *22. NaytiKoordinatu(kadr,'I',vyh);*

 *23. Если oshibka = 1, то переход к п. 24*

 *24. I:= X\_pred + I;*

 *25. preobrazovanieMAYAK(I,kadr);*

 *26. X\_pred:= X\_nast;*

 *27. Y\_pred:= Y\_nast;*

 *28. конец*

 Таблица 6

Список условных обозначений

|  |  |
| --- | --- |
| Обозначение | Расшифровка |
| Kadr | Текущая строка обработки |
| NaytiKoordinatu(kadr,'X', vyh); | Функция нахождения значения координаты |
| vyh | Выходное значение процедуры |
| PreobrazovanieMayak(X\_nast, kadr); | Функция преобразования координаты  |
| X\_nast и X\_pred | Числовая переменная настоящего и предыдущего значения Х |
| uslovie | Числовая переменная |
| FindVstavki('X', kadr); | Функция поиска вставки значения заданной координаты в строке |
| DobavitTochku(s); | Функция вставляет точку в строку, если её нет |
| insert(Concat('X', s), kadr, uslovie); | Функция вставляет и сцепляет строки |
| (pos('I', Stroka.Strings[Succ(m)]) | Функция поиска и проверки заданной позиции в строке |
| Podhod\_G41\_G42\_radius | Функция расчёта радиусов на заходах |
| Podhod\_G41\_G42\_line; | Функция расчёта линейных заходов |
| I и J | Координаты круговой интерполяции в программе |
| Y\_nast и Y\_pred | Числовая переменная настоящего и предыдущего значения Y |

#### Схема описание алгоритма конвертирования

if (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0)

Начало

*pos('%', kadr) <> 0) or (pos('$', kadr) <> 0)*

 *NaytiKoordinatu(kadr,'X', vyh);*

pos('X',kadr) <> 0

If oshibka <> 1

insert(Concat('X', s), kadr, uslovie);

pos('Y',kadr) <> 0

PreobrazovanieMayak(X\_nast, kadr);

 *NaytiKoordinatu(kadr,'Y', vyh);*

If oshibka <> 1

PreobrazovanieMayak(Y\_nast, kadr);

insert(Concat('Y', s), kadr, uslovie);

if (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0)

Insert('X' + s, kadr, uslovie);

Insert('Y' + s, kadr, uslovie);

Podhod\_G41\_G42\_radius;

Podhod\_G41\_G42\_line;

NaytiKoordinatu(kadr,'I',vyh);

preobrazovanieMAYAK(I,kadr);

NaytiKoordinatu(kadr,'J',vyh);

preobrazovanieMAYAK(J,kadr);

X\_pred:= X\_nast;

Y\_pred:= Y\_nast;

Конец

Конец

Рис. 2.7

## 2.5 Описание программы «Наладчик»

### 2.5.1. Водная часть

Программа «Наладчик» применяется для автоматизации процесса пересчёта исходного текста (кода) программ для станков с ЧПУ. При этом программа использует человеко-машинный интерфейс: человек выбирает, чем оперировать и параметры обработки данных, а программа выполняет все необходимые расчеты.

Для начала работы следует загрузить текстовый файл (программный код для станка с ЧПУ), и задать параметры его обработки. После этого программа производит пересчёт координат из данной в заданную систему программирования.

Программа имеет обозначение «Наладчик», разработана в среде программирования Borland Delphi 7.

### 2.5.2. Функциональное назначение

 Программа выполняет конвертирование (пересчёт) исходных координат в заданную систему ЧПУ. Программа использует алгоритмы, описанные в подразделах 2.2-2.4.

Для функционирования программы необходим следующий комплекс технических средств:

1. процессор IBM совместимый 486 и выше;
2. подсистема «Наладчик» занимает около 5 Мб оперативной памяти. Дополнительно к этому программа выделяет дополнительную память под текстовые файлы, загруженные в её оболочку;
3. для размещения ОС Windows XP и прикладных задач на жестком диске необходимо минимально 1,5 Гб свободного пространства. На диске программа занимает 625 килобайт памяти. Дополнительное место на диске занимают текстовые файлы используемые программой, а также файлы создаваемые программой. С учетом этого на диске необходимо выделить достаточное пространство для всех файлов (от 10 Мб и выше);
4. видеоподсистема с объемом памяти 32 Мб;
5. цветной SVGA монитор с диагональю не менее 15’’ и разрешением не ниже 800x600 (рекомендуется 1024x768);
6. русифицированная клавиатура;
7. манипулятор «мышь».

### 2.5.3. Описание информации

Вся входная и выходная информация – это текстовые файлы, написанные программистом вручную либо сгенерированные специальной графической программой обработки.

### 2.5.4. Используемые подпрограммы

В программе использованы следующие подпрограммы:

1. NaytiKoordinatu – нахождения значения заданной координаты;
2. DobavitTochku – добавление необходимой точки;
3. Podhod\_G41\_G42\_radius – расчёт радиусов при заходе инструмента на эквидистанту контура обрабатываемой детали;
4. preobrazovanieVM - преобразует строку, удаляет и вставляет подстроку (система ВМ);
5. preobrazovanieMAYAK - Преобразует строку, удаляет и вставляет подстроку (система Маяк);
6. Podhod\_G41\_G42\_line – расчёт линейных заходов на эквидистанту контура обрабатываемой детали;
7. FindVstavki – поиск места вставки заданной координаты;
8. DelProbel – удаление всех пробелов;
9. Pustostroka – удаление всех пустых строк;
10. Nomerkadra – восстановление последовательной нумерации кадров;
11. PC\_MAYAK – пересчёт координат из системы Синумерик в систему Маяк;
12. VM\_MAYAK – пересчёт координат из системы Вм в систему Маяк;
13. MAYAK\_VM – пересчёт координат из системы Маяк в систему Вм;
14. PC\_VM – пересчёт координат из системы Синумерик в систему Вм.

### 2.5.5. Описание логики

Описание логики программы «Наладчик» приведено на рис. 2.8.

Логическая схема

начало

Открытие

файла

Редактор не пуст и метод задан

Обработка заданными методом

Выход

конец

Выбор основного метода обработки

Выбор дополнительных методов обработки

Рис. 2.8

## 2.6. Описание контрольного примера

### 2.6.1. Назначение

Контрольный пример предназначен для проверки корректности работы программы.

### 2.6.2. Исходные данные

В качестве исходных данных использовался текстовый файл программы обработки.

### 2.6.3. Результаты расчета

В результате обработки (конвертирования) должна получиться полностью работоспособная программа для станков с ЧПУ. Обработка подразумевает использование всех методов конвертирования. А так же с использованием завед7омо неверных координат, с целью их автоматического преобразования программой, и создание исключительных ситуаций для обработки и их устранения.

### 2.6.4. Результаты испытания

В результате обработки данных контрольного примера получены текстовые файлы для различных систем программирования. Они были проверены и просчитаны вручную и полностью соответствуют выбранной системе конвертирования.

# 3. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## 3.1. Обоснование необходимости разработки системы конвертирования текста программ для станков с ЧПУ

Система конвертирования является самостоятельной программой предназначенной для полного анализа, пересчёта и исправления ошибок в текстах программ для станков с ЧПУ. Разрабатываемая система служит на производстве в течение уже нескольких лет и зарекомендовала себя с лучшей стороны.

- программа позволяет исправлять ошибки, полученные в результате не правильных действий программиста;

 - практически исключает правку программы вручную;

 - уменьшает объём программного текста.

Таким образом, разработка системы позволит, помимо выполнения основной задачи, сэкономить на объеме памяти устройств хранения информации и снизить нагрузку и требования по скорости разработки и внедрения программ для станков с ЧПУ.

## 3.2. Планирование разработки

Планирование разработки программного обеспечения позволяет правильно организовать рабочий процесс, распределить обязанности, обозначить перечень необходимых работ и выполнить его в установленные сроки.

Расчет трудоемкости разработки задачи приведен в табл.7

Таблица 7

Расчет трудоемкости разработки задачи

|  |  |  |
| --- | --- | --- |
| Наименование этапа разработки | Исполнители | Трудоемкость работ, чел-ч |
| Должность | Кол-во |
| 1. Изучение постановки задачи и методов решения
 | инженер-программист |  1 | 84 |
| 1. Разработка алгоритмов
 | инженер-программист | 1 |  168 |
| 1. Реализация и отладка алгоритмов
 | инженер-программист |  1 | 168 |
| 1. Документирование системы
 | инженер-программист |  1 | 84 |
| ИТОГО |  |  1 | 504 |

## 3.3. Расчет расходов на создание системы

Для определения величины расходов на создание системы, используем метод прямого счета. Расчет расходов осуществляется по следующим статьям:

1. расходы на материалы;
2. расходы на оплату труда исполнителей;
3. социальные налоги;
4. расходы на содержание ВТ;
5. накладные расходы;
6. прочие расходы.

В статью “Расходы на материалы” включают стоимость основных и вспомогательных материалов, покупных изделий, а также затраты энергии и запчастей необходимых для создания системы. Расходы по статье приведены в табл. 8

Таблица 8

Расходы на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование материала | Количество | Цена ед, руб | Расходы |
| Бумага писчая | 1 пачка (500л) | 100.00 | 100.00 |
| Диск перезаписываемый CD-RW | 1 шт. | 40.00 | 40.00 |
| Картридж для принтера Canon BJC-1000 | 1 шт. | 150.00 | 150.00 |
| Прочие канцелярские товары |  |  | 50.00 |
| ИТОГО |  |  | 340.00 |

Расчет расходов по оплате труда исполнителей.

Оклад инженера-программиста на период разработки (Пм=3 месяца) составляет в среднем ЗПм=4500.00 рублей в месяц. Тогда расходы на оплату труда:

 **Рот**тариф=**ЗПм**\***Пм** (3.1)

Роттариф=4500.00\*3 =13500.00 руб.

С учетом уральского коэффициента (15%) общие расходы на оплату труда исполнителей составят:

 **Рот**=**Зп**+**Кур** (3.2)

Кур=13500.00\*0.15= 2025.00 руб.

Рот=13500.00+2025.00 = 15525.00 руб.

Отчисления в фонды резерва отпусков (10%) от расходов на оплату труда исполнителей:

 **РРезО=Рот\*0.1** (3.3)

РРезО=15525.00 \*0.1=1552.50 руб.

Статья “Социальные налоги” составляет 26,2 % от оплаты труда и предназначена для перечисления во внебюджетные государственные фонды:

1. пенсионный (20%);
2. обязательного медицинского страхования (3,1%);
3. социального страхования (2,9%);
4. обязательное страхование от несчастных случаев (0,2%).

Сумма по статье составляет:

 **Рсоц нал=(Рот+РРезО)**\***26.2/100** (3.4)

Рсоц нал=(15525.00 +1552.50)\*26.2/100=4474.31 руб.

Статья “Расходы на содержание ВТ” включает расходы, связанные с арендой ВТ и обслуживанием ВТ. Эти расходы включают в себя амортизационные отчисления, процент за аренду (комиссионные), затраты на запчасти, стоимость электроэнергии (если они не включены в материальные расходы), а также другие расходы, связанные с содержанием ВТ.

Стоимость одного машинного часа рассчитывается по формуле:

 **Ач = Сисп / (Чм \* Кч),** (3.5)

где Ач - аренда за час использования;

Сисп - общая стоимость использования оборудования (рассчитывается по формуле);

Чм - число месяцев в году;

Кч - количество рабочих часов в месяце.

 **Сисп=Акомп+Апо+ЗПобсл+Сзч+Сэл**, (3.6)

где Акомп – амортизация компьютера и принтера за год эксплуатации;

АПО – амортизация программного обеспечения;

ЗПобсл – расходы обслуживающего персонала за год эксплуатации;

Сзч – стоимость запчастей для компьютера за год эксплуатации;

Сэл – стоимость израсходованной электроэнергии за год эксплуатации.

 **Акомп=Скомп/Спи,** (3.7)

где Скомп – стоимость компьютера и принтера;

Спи – срок полезного использования (5 лет).

Акомп=34000/5=6800 руб.

 **А­по=Спо/Спи,** (3.8)

где Спо – стоимость программного обеспечения;

Спи – срок полезного использования (5 лет).

Апо=10000/5=2000 руб.

Затраты на потребляемую от сети электроэнергию определяются по формуле:

 **Сэл=N\*Тэксп\*Сэ,** (3.9)

где N – потребляемая мощность, кВт.; (220\*(0.24+1.5+4)/√2=0.893 кВт)

Тэксп – продолжительность эксплуатации за год; (1920 ч.)

Cэ – стоимость 1 кВт.-ч. силовой энергии. (1.26 р./кВт)

Сэл=0.893\*1920\*1.26=2160.35 руб.

Стоимость годового использования составляет:

Сисп=6800+2000+2000+1000+2160.35 =13960.35 руб.

Соответственно аренда за час использования:

Ач=13960.35/(12\*160)=7.27 руб.

ВТ использовалась на всех этапах разработки системы (см. табл. 3.1) Общее время эксплуатации ВТ составила Тисп=504 часа. Следовательно, сумма расходов на ВТ, т.е. на аренду составляет:

 **Рвт=А=Тисп\*Ач** (3.10)

Рвт=504\*7.27 =3664.08 руб.

Статья “Накладные расходы” включает в себя затраты, связанные с содержанием управленческого персонала, на предприятии составляет до 130% основной заработной платы:

 **Нр=Рот\*1.30** (3.11)

Нр=15525.00\*1.30=20182.5 руб.

Статья “Прочие расходы” включает в себя расходы, неучтенные в предыдущих статьях (до 10% от основной заработной платы):

 **Пр=Рот\*0.10** (3.12)

Пр=15525.00\*0.10=1552.5 руб.

Табл. 9 отражает затраты по статьям и долю этих затрат в общей сумме. Долевое соотношение затрат представлено на рисунке 3.1

Таблица 9

Смета затрат на разработку системы

|  |  |
| --- | --- |
| Статьи затрат | Сумма затрат, руб. |
| Расходы на материалы | 340 |
| Расходы на оплату труда исполнителей | 17080 |
| Социальные налоги | 4470 |
| Расходы на содержание ВТ | 3660 |
| Накладные расходы | 20190 |
| Прочие расходы | 1550 |
| **Итого** | 47290 |

Структура затрат на разработку


 Рис 3.1

# 4. БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ ПРОЕКТА

Компьютеры в настоящее время широко используются во всех сферах человеческой деятельности. Они стали основным рабочим инструментом человека в его ежедневной деятельности. В связи с этим необходимо уделять внимание охране труда при работе с ПК. Несоблюдение санитарно-гигиенических правил и норм при работе с компьютером может повлечь за собой развитие ряда заболеваний.

Охрана труда имеет не только социальный, но и экономический аспект. В результате улучшения условий труда наблюдается повышение производительности труда, снижение непроизводительных затрат времени, увеличение фонда рабочего времени, экономия расходов на оплату больничных и компенсации за работу в неблагоприятных условиях труда.

В данном разделе дипломного проекта освещаются основные вопросы техники безопасности и экологии труда.

## 4.11 Анализ вредных и опасных производственных факторов

**Постоянное напряжение глаз**

Работа с компьютером характеризуется высокой напряженностью зрительной работы. Постоянное напряжение глаз может привести к снижению остроты зрения. Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600…700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов. Также для снижения утомляемости рекомендуется делать 15-минутные перерывы в работе за компьютером в течение каждого часа.

**Влияние электростатических и электромагнитных полей**

Точное действие электромагнитных полей не изучено, однако исследования показали, что излучение осложняет сердечнососудистые заболевания, отрицательно влияет на развитие беременности, снижает иммунитет. Учитывая, что программист проводит много времени с компьютером, воздействие электромагнитных полей может отрицательно отразиться на его здоровье.

Допустимые значения параметров неионизирую­щих электромагнитных излучений от монитора компьютера представлены в табл. 4.1.

Максимальный уровень рентгеновского излучения на рабочем месте оператора ком­пьютера обычно не превышает 10 мкбэр/ч, а интенсивность ультрафиолетового и ин­фра­красного излучений от экрана монитора лежит в пределах 10…100 мВт/м2 /18/.

Таблица 4.1

Допустимые значения параметров неионизирующих электро­магнитных излучений (в соответствии с СанПиН 2.2.2.542-96)

|  |  |
| --- | --- |
| Наименование параметра | Допустимые значения |
| Напряженность электриче­ской составляющей электромагнитногополя на расстоянии 50см от поверхно­сти видеомонитора | 10В/м |
| Напряженность магнитной составляющей электромагнитногополя на расстоянии 50см от поверхности ви­деомонитора | 0,3А/м |
| Напряженность электростатического поля не должна превышать:для взрослых пользователейдля детей дошкольных учреждений и учащихсясредних специальных и высших учебных заведений | 20кВ/м15кВ/м |

Для снижения воздействия этих видов излучения рекомендуется применять мониторы с пониженным уровнем излучения (MPR-II, TCO-92, TCO-99), устанавливать защитные экраны, а также соблюдать регламентированные режимы труда и отдыха.

**Длительное неизменное положение тела**

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой ЭВМ. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы человека-оператора.

**Шум**

Шум ухудшает условия труда, оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражитель­ность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. В табл. 4.2 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности/21/.

Таблица 4.2

Предельные уровни звука, дБ, на рабочих местах

|  |  |
| --- | --- |
| Категориянапряженности труда | Категория тяжести труда |
| I. Легкая | II. Средняя | III. Тяжелая | IV. Очень тяжелая |
| I. Мало напряженный | 80 | 80 | 75 | 75 |
| II. Умеренно напряженный | 70 | 70 | 65 | 65 |
| III. Напряженный | 60 | 60 | - | - |
| IV. Очень напряженный | 50 | 50 | - | - |

Уровень шума на рабочем месте инженеров-программистов и операторов видеоматериалов не должен превышать 50дБА, а в залах обработки информации на вычислительных машинах **-** 65дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами.

## 4.2. Техника безопасности при работе с компьютером

**Принцип защиты расстоянием**

Основную опасность при работе с компьютером представляет электромагнитное излучение. Наиболее сильно оно проявляется в зоне до 30 см от экрана, а также от задней и боковых поверхностей. Поэтому монитор необходимо располагать так, чтобы глаза находились на расстоянии не ближе 60-70 см от экрана. Между боковыми поверхностями соседних компьютеров должно быть не менее 1,2м. При расположении рядами: между тыльными сторонами компьютеров должно быть не менее 2м. Рекомендуется располагать компьютеры вдоль стены.

**Принцип защиты временем**

По санитарным правилам и нормам СанПиН 2.2.2.542-96 «Гигиенические требования к видеодисплейным терминалам, персональным электронным вычислительным машинам и организации работы» /18/ виды трудовой деятельности подразделяются на три группы:

* группа А - работа по считыванию информации с экрана;
* группа Б - работа по вводу информации;
* группа В - творческая работа в режиме диалога с ЭВМ.

Деятельность разработчика программного обеспечения относится к группе В. При выполнении работ, относящихся к разным видам трудовой деятельности, за основную работу с ЭВМ принимают такую, которая занимает не менее 50% времени в течение рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливаются три категории тяжести и напряженности работ с ЭВМ и видеотерминалом (табл. 4.3).

 Таблица 4.3

Категории тяжести и напряженности работ

|  |  |  |
| --- | --- | --- |
| Категория работы с ПК | Уровень нагрузки за рабочую смену при работе с ПК (max) | Суммарное время регламентированных перерывов, мин. |
| Группа А, кол-во знаков | Группа Б, кол-во знаков | Группа В,кол-во часов | При 8-ми часовой смене | При 12-ти часовой смене |
| I | 20 000 | 15 000 | 2 | 30 | 70 |
| II | 40 000 | 30 000 | 4 | 50 | 90 |
| III | 60 000 | 40 000 | 6 | 70 | 120 |

При 8-ми часовой рабочей смене и работе на ПК регламентированные перерывы следует устанавливать:

1. для I-ой категории работ через 2 часа от начала рабочей смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;
2. для II-ой категории работ через 2 часа от начала рабочей смены и через 1,5 - 2 часа после обеденного перерыва продолжительностью 15 минут каждый или 10 минут через каждый час работы;
3. для III-ой категории работ через 1,5 - 2 часа от начала рабочей смены и через 1,5 - 2 часа после обеденного перерыва продолжительностью 20 минут каждый или 15 минут через каждый час работы.

При 12-ти часовой смене и работе на ПК регламентированные перерывы следует устанавливать: в первые 8 часов работы аналогично перерывам при 8-ми часовом рабочей смене, а в течении последующих 4-х часов работы не зависимо от категории и вида работ 15 минут через каждый час.

## 4.3. Электробезопасность при работе с компьютером

Электрические установки, к которым относится практически все оборудование ЭВМ, представляют для человека большую потенциальную опасность, так как в процессе эксплуатации или проведении профилактических работ человек может коснуться частей, находящихся под напряжением.

Любое воздействие тока может привести к электрической травме, то есть к повреждению организма, вызванному действием электрического тока или электрической дуги /19/.

При рассмотрении вопроса обеспечения электробезопасности разработчика необходимо выделить три основных фактора:

1. электроустановки рабочего места программиста;
2. вспомогательное электрооборудование;
3. окружающая среда помещения.

К электроустройствам рабочего места относятся: компьютер, видеомонитор, принтер. К вспомогательному оборудованию относятся лампы местного освещения, вентиляторы и другие электрические приборы. Электрооборудование, перечисленное выше, относится к установкам напряжением до 1000 В, исключение составляют лишь дисплей, электронно-лучевые трубки, которых имеют напряжение в несколько киловольт.

Окружающая среда помещений, в которых работает программист, воздействует на электрическую изоляцию приборов и устройств, электрическое сопротивление тела человека и может создавать условия для поражения электрическим током.

Помещения, оборудованные вычислительной техникой, как правило, относятся к категории помещений без повышенной опасности так как:

1. относительная влажность воздуха не превышает 75%;
2. нет токопроводящей пыли;
3. температура не превышает длительное время 30 °С;
4. отсутствует возможность одновременного прикосновения человека с имеющими соединение с землей металлическими конструкциями;
5. отсутствие доступа к токоведущим частям оборудования;
6. нет токопроводящих полов.

Таким образом, для предотвращения электротравматизма пользователя, необходимо соблюдать требования безопасности, как при работе с обычной бытовой техникой.

## 4.4. Требования к освещению рабочего места и его расчет

### 4.4.1. Требования к освещению рабочего места

К современному освещению помещений, где работают с вычислительной техникой, предъявляют высокие требования как гигиенического, так и технического характера. Правильно спроектированное и выполненное освещение обеспечивает высокий уровень работоспособности, оказывает положительное психологическое воздействие, способствует повышению производительности труда. Условия деятельности пользователя в системе "человек-машина" связаны с явным преобладанием зрительной информации - до 90% общего объема.

В помещениях с компьютерной техникой применяется совмещенная система освещения/17/. К таким системам предъявляют следующие требования:

1. соответствие уровня освещенности рабочих мест характеру выполняемых зрительных работ;
2. достаточно равномерное распределение яркости на рабочих поверхностях и в окружающем пространстве;
3. отсутствие резких теней, прямой и отраженной блеклости;
4. постоянство освещенности во времени;
5. оптимальная направленность излучаемого осветительными приборами светового потока;
6. долговечность, экономичность, электро- и пожаробезопасность, эстетичность, удобство и простота эксплуатации.

Для искусственного освещения помещений с вычислительной техникой следует использовать главным образом люминесцентные лампы, у которых высокая световая отдача (до 75 лм/Вт и более); продолжительный срок службы (до 10 000 ч), малая яркость светящейся поверхности, близкий к естественному спектр излучения, что обеспечивает хорошую цветопередачу. Наиболее приемлемыми являются люминесцентные лампы белого света и тепло-белого света мощностью 20, 40, 80 Вт.

Для исключения засветки экранов дисплеев прямым световым потоком, светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора и стене с окнами. Такое расположение светильников позволяет производить их последовательное включение по мере необходимости и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

Рекомендуемая освещенность для работы с экраном дисплея составляет 150 лк, а при работе с экраном в сочетании с работой над документами - 300 лк. Рекомендуемая яркость в поле зрения программиста должна лежать в пределах 1:5-1:10.

### 4.4.2. Расчет освещенности рабочего места

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Процесс работы программиста в таких условиях, когда естественное освещение недостаточно или отсутствует. Исходя из этого, рассчитаем параметры искусственного освещения.

Расчет освещения производится для комнаты площадью 42 м2: длина – 7 м, ширина - 6 м, высота - 3 м. Равномерное освещение горизонтальной рабочей поверхности достигается при определенных отношениях расстояния между центрами светильников L, м (L=1,75\*Н) к высоте их подвеса над рабочей поверхностью Н, м (в расчетах Н=Н).

L = 1,75\*Н = 1,75\*3 = 5,25 м.

Число светильников с люминесцентными лампами (ЛЛ)

, (4.1)

где S – площадь помещения, м;

 М – расстояние между параллельными рядами, м.

М0,6 Н

М0,6\*3=1,8 принимаем М=2;

;

Для достижения равномерной горизонтальной освещенности светильники с ЛЛ рекомендуется располагать сплошными рядами, параллельно стенам с окнами или длинным сторонам помещения.

Для расчета общего равномерного освещения горизонтальной рабочей поверхности используют метод светового потока, учитывающий световой поток, отраженный от потолка и стен.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

**, (4.2)

где **Fp** - рассчитываемый световой поток, Лм;

 **Еn** - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет **Е** = 300 Лк при газоразрядных лампах;

 **S** - площадь освещаемого помещения ( в нашем случае **S** = 42 м2 );

 **Z** - отношение средней освещенности к минимальной (обычно принимается равным 1.1-1.2 , пусть **Z** = 1.1);

 **К** - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение определяется по таблице коэффициентов запаса для различных помещений и в нашем случае **К** = 1.5);

 **η** - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен (Рс) и потолка (Рп)), значение коэффициентов Рс и Рп определим по таблице зависимостей коэффициентов отражения от характера поверхности: Рс=30%, Рп=50%. Значение **η** определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

**, (4.3)

где **S** - площадь помещения, **S** = 42 м2;

 **h** - расчетная высота подвеса, **h** = 3 м;

 **A** - ширина помещения, **А** = 6 м;

 **В** - длина помещения, **В** =7 м.

Подставив значения получим:



Зная индекс помещения **I**, **Рс** и **Рп**, по таблице находим **η** = 0.3

Подставим все значения в формулу для определения светового потока **F**:

 Лм

Учитывая, что в светильнике по 4 лампы, световой поток одной лампы будет равен . Для освещения выбираем люминесцентные лампы типа ЛБ20, световой поток которых **Fл** = 1180 Лм.

#### Рассчитываем потребляемую мощность осветительной установки в соответствии с формулой:

 , (4.4)

где *р –* мощность лампы, Вт;

 *N –* число светильников, шт. *N*=4;

 *n –* число ламп в светильнике; для ЛЛ *n=*4.

P=20\*4\*4=320 Вт=0,32 кВт.

Рассчитаем действительное освещение.

В рабочем помещении имеется 4 светильников, в каждом из которых используется по 4 лампы ЛДЦ20. Рассчитаем световой поток:

Ф= 820\*4\*4 = 3280 лк;

Рассчитаем значение нормированной освещенности:

Е = (Ф\*N\*η)/(S\*Z\*K) = 227 лк ≤ 300 лк.

Полученное значение нормированной освещенности оказалось меньше нормированной минимальной освещенности. Рекомендуется заменить используемые лампы, на лампы с большим световым потоком.

## 4.5. Параметры микроклимата, загрязнение воздуха рабочей зоны вредными веществами

Под метеорологическими условиями понимают сочетание температуры, относительной влажности, скорости движения и запыленности воздуха. Перечисленные параметры оказывают огромное влияние на функциональную деятельность человека, его самочувствие и здоровье и на надежность средств вычислительной техники. Эти микроклиматические параметры влияют как каждый в отдельности, так и в различных сочетаниях.

Температура воздуха является одним из основных параметров, характеризующих тепловое состояние микроклимата. Суммарное тепловыделение в помещении поступает от следующих источников:

1. ЭВМ;
2. вспомогательного оборудования;
3. приборов освещения;
4. людей;
5. внешних источников.

Наибольшее количество теплоты выделяют ЭВМ и вспомогательное оборудование. Средняя величина тепловыделения от компьютеров колеблется до 100 Вт/м2 . Тепловыделения от приборов освещения также велики. Удельная величина их составляет 35 Вт/м2.При этом, чем больше уровень освещенности, тем выше удельные величины тепловыделений. Количество теплоты от обслуживающего персонала незначительно. Оно зависит от числа работающих в помещении, интенсивности работы, выполняемой человеком.

К внешним источникам поступления теплоты относят теплоту, поступающую через окна от солнечной радиации, приток теплоты через непрозрачные ограждения конструкций. Интенсивность этих источников зависит от расположения здания, ориентации по частям света, цветовой гаммы и прочее/15/.

С целью создания нормальных условий труда программиста установлены оптимальные и допустимые значения всех параметров микроклимата (табл. 4.4.).

Таблица 4.4

Параметры микроклимата производственных помещений

| Параметры | Значения параметров |
| --- | --- |
| оптимальные | допустимые |
| Температура | 20-22 °С | 17-22 °С |
| Относительная влажность | 40-60 % | до 75% |
| Скорость движения воздуха | 0,1 м/с | не более 0,3 м/с |

В целях поддержания температуры и влажности воздуха в помещении можно использовать системы отопления, вентиляции и кондиционирования воздуха.

На исследуемом рабочем месте температура воздуха, влажность и скорость движения воздуха держится в рамках оптимальных параметров. Вредные вещества в воздухе рабочей зоны не превышают предельной допустимой концентрации.

## 4.6. Пожаробезопасность при работе с компьютером

Помещение, в котором установлено рабочее место инженера-программиста, относится к категории Д по взрывопожароопасности, так как не содержит горючих веществ, но лишь негорючие вещества и материалы в холодном состоянии.

Пожары в помещении, в котором находится ЭВМ, представляют особую опасность, так как сопряжены с большими материальными потерями. Площадь помещения, в котором ведется проектирование, невелика и составляет 8 м2. Как известно пожар может возникнуть при взаимодействии горючих веществ, окисления и источников зажигания. В помещении присутствуют все три основные фактора, необходимые для возникновения пожара. Горючими компонентами являются: строительные материалы для акустической и эстетической отделки помещений, двери, полы, бумага, изоляция кабелей и др.

Противопожарная защита - это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

Источниками зажигания в помещении, содержащем ЭВМ, могут быть электронные схемы от ЭВМ, приборы, применяемые для технического обслуживания, устройства электропитания, где в результате различных нарушений образуются перегретые элементы, электрические искры и дуги, способные вызвать загорания горючих материалов.

В современных ЭВМ очень высока плотность размещения элементов электронных схем. В непосредственной близости друг от друга располагаются соединительные провода, кабели. При протекании по ним электрического тока выделяется значительное количество теплоты. При этом возможно оплавление изоляции. Для отвода избыточной теплоты от ЭВМ служат системы вентиляции и кондиционирования воздуха. При постоянном действии эти системы представляют собой дополнительную пожарную опасность.

Одной из наиболее важных задач пожарной защиты является защита строительных помещений от разрушений и обеспечение их достаточной прочности в условиях воздействия высоких температур при пожаре. Учитывая высокую стоимость электронного оборудования, а также категорию его пожарной опасности, здания, в которых предусмотрено размещение ЭВМ, должны быть 1 и 2 степени огнестойкости.

К средствам тушения пожара, предназначенных для локализации небольших возгораний, относятся пожарные стволы, внутренние пожарные водопроводы, огнетушители, сухой песок, асбестовые одеяла и т. п.

В соответствии с “Типовыми правилами пожарной безопасности для промышленных предприятий” залы ЭВМ, помещения для внешних запоминающих устройств, подготовки данных, сервисной аппаратуры, архивов, копировально-множительного оборудования и т.п. необходимо оборудовать дымовыми пожарными извещателями. Так как в этих помещениях при возгорании различных пластмассовых, изоляционных материалов и бумажных изделий выделяется значительное количество дыма и мало теплоты.

Помещение, в котором производится разработка данного проекта, необходимо оборудовать средствами оповещения о пожаре, а также средствами для тушения пожара.

Электрические установки представляют для человека большую потенциальную опасность, так как в процессе эксплуатации или проведении профилактических работ человек может коснуться частей, находящихся под напряжением. Специфическая опасность электроустановок: токоведущие проводники, корпуса стоек ЭВМ и прочего оборудования, оказавшегося под напряжением в результате повреждения (пробоя) изоляции, не подают каких-либо сигналов, которые предупреждают человека об опасности. Реакция человека на электрический ток возникает лишь при протекании последнего через тело человека. Разрядные токи статического электричества чаще всего возникают при прикосновении к любому из элементов ЭВМ. Такие разряды опасности для человека не представляют, но кроме неприятных ощущений они могут привести к выходу из строя ЭВМ. Для снижения величины возникающих зарядов статического электричества покрытие технологических полов следует выполнить из однослойного поливинилхлоридного антистатического линолеума.

В ходе дипломного работы были выявлены вредные факторы, влияющие на работоспособность инженера-программиста. Для обеспечения безопасных условий труда инженера-программиста были выдвинуты требования к электро- и пожаробезопасности на рабочем месте. Также был произведен расчет общего освещения, в результате которого была выявлена недостаточность освещения. Предлагаемые меры: замена ламп на лампы с большим световым потоком.

# ЗАКЛЮЧЕНИЕ

 В ходе дипломной работы было разработано ПО, позволяющее решать задачу автоматизированного пересчёта программного кода для станков с ЧПУ с использованием различных систем программирования. Программа была написана на языке высокого уровня Object Pascal в среде разработки Borland Delphi 7.

 В процессе разработки были досконально исследованы системы программирования станков с ЧПУ, такие как (Вм, Маяк, Sinumerik). Результаты тестирования показали что, программа верно пересчитывает программный код в различные системы программирования при этом устраняет ошибки допущенные программистом, а так же имеет много вспомогательных настроек. Что значительно упрощает человеко-машинное взаимодействие и повышает время разработки программного кода и ввод его в эксплуатацию.

# СПИСОК ЛИТЕРАТУРЫ

1. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. - М.: Наука, 1979. – 720с.
2. Никита Культин Delphi 6. Программирование на Object Pascal. – СПб.: БХВ – Петербург, 2002. – 528 с.: ил.
3. <http://tnr.kulichki.com/techedit/te.html> - TechEdit 2.1.1 относится к разряду специального ПО. С его помощью можно автоматизировать процесс разработки программ для ЧПУ.
4. Гофман В. Э., Хомоненко А. Д. Delphi 6. – СПБ.: БХВ – Петербург, 2002. – 1152 с.: ил
5. ГОСТ 12.0.002-80. Система стандартов безопасности труда. Термины и определения. – М.: ИПК Издательство стандартов, 2002
6. СанПиН 2.2.4. 548 – 96. Гигиенические требования к микроклимату производственных помещений. – М.: Госкомсанэпиднадзор, 1996.
7. ГОСТ 12.1.006–84. ССБТ. Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля - М.: Издательство стандартов, 1985.
8. СанПиН 2.2.2. 542-96. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы. – М.: Госкомсанэпиднадзор, 1996.
9. ГОСТ 12.1.038-83. Система стандартов безопасности труда. Электробезопасность. Термины и определения. – М.: Издательство стандартов, 1983.
10. ГОСТ 12.1.004-91. Система стандартов безопасности труда. Пожарная безопасность. Общие положения. – М.: Издательство стандартов, 1991.
11. ГОСТ 12.1.003–83. ССБТ. Шум. Общие требования безопасности (с изменениями по И-1-III-89).
12. Почерняев С.В., Килин И.В., Сенилов М.А. Методические указания по дипломному проектированию. – Ижевск: Издательство ИжГТУ, 1994.
13. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Издательство стандартов, 1991
14. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам. – М.: Издательство стандартов, 1988
15. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1988
16. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1988
17. ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1988
18. ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению. – М.: Издательство стандартов, 1988

# ПРИЛОЖЕНИЕ 1

# ТЕКСТ ПРОГРАММЫ

**Файл Project1.dpr**

program Project1;

uses

 Forms,

 Unit1 in 'Unit1.pas' {Form1},

 Unit2 in 'Unit2.pas' {Form2},

 Unit3 in 'Unit3.pas' {AboutBox},

 Unit5 in 'Unit5.pas' {Form5},

 Unit6 in 'Unit6.pas' {Form6};

{$R \*.res}

begin

 Application.Initialize;

 Application.Title := 'Наладчик 1.6.1';

 Application.CreateForm(TForm1, Form1);

 Application.CreateForm(TForm5, Form5);

 Application.CreateForm(TForm6, Form6);

 Form1.Label3.Caption:= 'Всего строк';

 Application.CreateForm(TForm2, Form2);

 Application.CreateForm(TAboutBox, AboutBox);

 Application.Run;

end.

**Файл Unit1.pas**

unit Unit1;

interface

uses

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

 Dialogs, StdCtrls, Menus, ExtCtrls, ToolWin, ComCtrls, ImgList, Printers,

 DBCtrls, Registry, IniFiles;

type

 TForm1 = class(TForm)

 memo1: TMemo;

 MainMenu1: TMainMenu;

 OpenDialog1: TOpenDialog;

 FontDialog1: TFontDialog;

 PrinterSetupDialog1: TPrinterSetupDialog;

 PrintDialog1: TPrintDialog;

 fayl: TMenuItem;

 Nastroyki: TMenuItem;

 fond: TMenuItem;

 Printery: TMenuItem;

 ToolBar1: TToolBar;

 ToolButton1: TToolButton;

 ImageList1: TImageList;

 Parametry: TMenuItem;

 ToolButton2: TToolButton;

 SaveDialog1: TSaveDialog;

 ToolButton3: TToolButton;

 Label1: TLabel;

 Help: TMenuItem;

 Versia: TMenuItem;

 Vyhod: TMenuItem;

 Save: TMenuItem;

 Open: TMenuItem;

 Print: TMenuItem;

 ToolButton4: TToolButton;

 ToolButton6: TToolButton;

 NomerKadra: TCheckBox;

 Probel: TCheckBox;

 Pustostr: TCheckBox;

 PanelSostoyaniya: TStatusBar;

 Indikator: TProgressBar;

 ToolButton9: TToolButton;

 ReplaceDialog1: TReplaceDialog;

 Cancel: TButton;

 Obrabotka: TButton;

 Label2: TLabel;

 Label3: TLabel;

 Panel1: TPanel;

 Panel2: TPanel;

 ToolButton5: TToolButton;

 ToolButton8: TToolButton;

 procedure ButPrinterClick(Sender: TObject);

 procedure ButPechatClick(Sender: TObject);

 procedure fondClick(Sender: TObject);

 procedure PrinteryClick(Sender: TObject);

 procedure PechatClick(Sender: TObject);

 procedure ButParametrClick(Sender: TObject);

 procedure FormResize(Sender: TObject);

 procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

 procedure FormCreate(Sender: TObject);

 procedure ToolButton1Click(Sender: TObject);

 procedure ParametryClick(Sender: TObject);

 procedure ToolButton2Click(Sender: TObject);

 procedure ToolButton3Click(Sender: TObject);

 procedure VersiaClick(Sender: TObject);

 procedure VyhodClick(Sender: TObject);

 procedure SaveClick(Sender: TObject);

 procedure OpenClick(Sender: TObject);

 procedure PrintClick(Sender: TObject);

 procedure CancelClick(Sender: TObject);

 procedure ToolButton6Click(Sender: TObject);

 procedure ToolButton4Click(Sender: TObject);

 procedure ToolButton9Click(Sender: TObject);

 procedure ReplaceDialog1Replace(Sender: TObject);

 procedure ObrabotkaClick(Sender: TObject);

 procedure memo1KeyDown(Sender: TObject; var Key: Word;

 Shift: TShiftState);

 procedure ToolButton5Click(Sender: TObject);

 procedure ToolButton8Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

const

 st = ' Параметры: '; // текст в строке состояния + в модуле 2

 Code\_Begin = 100;

var

 Form1: TForm1;

 Code\_My, nach, oshibka, m, uslovie: Integer;

 Code\_Reg: longint;

 proverka\_sushesvovania\_ini: byte;

 reestra, reestrc, reestrb, reestrMy\_Code, reestrReg\_Code: integer;

 reestrversia: string[20];

 nach\_period, Hac: integer;

 Stroka: TStrings;

 memo1mod: boolean;

 I, J, I\_sled, J\_sled: real;

 X\_nast, X\_pred, X\_sled, Y\_nast, Y\_pred, Y\_sled: real;

 Z, A, Uvmmayak, K : real;

implementation

uses Unit2, Unit3, Unit5;

{$R \*.dfm}

procedure TForm1.ButPrinterClick(Sender: TObject);

begin

 PrinterSetupDialog1.Execute

end;

procedure TForm1.ButPechatClick(Sender: TObject);

begin

 PrintDialog1.Execute

end;

procedure TForm1.FondClick(Sender: TObject);

begin

 if FontDialog1.Execute then

 Memo1.Font:= FontDialog1.Font;

end;

procedure TForm1.PrinteryClick(Sender: TObject);

begin

 PrinterSetupDialog1.Execute

end;

procedure TForm1.PechatClick(Sender: TObject);

begin

 ToolButton3Click(Sender);

end;

procedure TForm1.ButParametrClick(Sender: TObject);

begin

 Form2.ShowModal;

end;

Procedure TForm1.FormResize( Sender: TObject); // Изменение размера формы

begin

 Memo1.Left:= 10; // Положение слева

 Memo1.Top:= 60; // Положение сверху

 Memo1.Width:= Form1.ClientWidth-130; // Ширина

 Memo1.Height:= Form1.ClientHeight-100; // Высота

 NomerKadra.Left:= Form1.ClientWidth-15-NomerKadra.Width;

 Probel.Left:= Form1.ClientWidth-15-Probel.Width;

 Pustostr.Left:= Form1.ClientWidth-15-Pustostr.Width;

 Cancel.Left:= Form1.ClientWidth-20-Cancel.Width;

 Obrabotka.Left:= Form1.ClientWidth-20-Obrabotka.Width;

 Label2.Left:= Form1.ClientWidth-40-Cancel.Width;

 Label3.Left:= Form1.ClientWidth-40-Cancel.Width;

 Panel1.Left:= Form1.ClientWidth-18-Panel1.Width;

 Panel2.Left:= Form1.ClientWidth-18-Panel2.Width;

end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

 if Memo1mod then // Если поле Мемо1 изменилось

 begin

 CanClose:= MessageDlg(' Содержимое редактора изменилось '#10#13' Хотите сохранить данные?', mtConfirmation,[mbYes, mbNo], 0) = mrYes;

 if CanClose then ToolButton2Click(Sender);

 CanClose:= true;

 end;

end;

procedure TForm1.FormCreate(Sender: TObject);

const

 Reg\_Code = 1234567890;

 My\_Code = 469103;

var

 RegIni: TRegistry;

 Ini: TIniFile;

 data: TDatetime;

begin

 DateSeparator:= '.';

 DecimalSeparator:= '.';

 ShortDateFormat:= 'dd/mm/yyyy';

 memo1mod:= false;

 reestrMy\_Code:= 0;

 reestrReg\_Code:= 0;

 data:= Date; // получаем сегодняшнюю дату

 nach\_period:= Trunc(data);

 Ini:= TIniFile.Create('Nal.ini');

 proverka\_sushesvovania\_ini:= Ini.ReadInteger('START', 'CountStart', 0);

 if proverka\_sushesvovania\_ini = 0 then

 begin

 ini.WriteInteger('START', 'CountStart', 1);

 ini.WriteDate('DateBegin', 'DateBegin', data); //в формате дд.мм.гггг

 ini.WriteDate('DateEnd', 'DateEnd', data + 30); //в формате дд.мм.гггг

 ini.WriteString('Versia', 'Versia', Application.Title);

 ini.WriteInteger('CodeBegin', 'CodeBegin', Code\_Begin);

 ini.WriteInteger('Hac','Hac', nach\_period);

 end

 else

 begin

 Code\_My:= ini.ReadInteger('Code\_My', 'Code\_My', 0);

 Code\_Reg:= ini.ReadInteger('Code\_Reg', 'Code\_Reg', 0);

 Hac:= Ini.ReadInteger('Hac','Hac',0);

 end;

 Ini.Free;

 Form1.Caption:= Application.Title;

 Form1.memo1.Clear;

 Form1.PanelSostoyaniya.Panels[0].Text:= st + ' Не обрабатывать ';

 Form1.Panel1.Caption:= '0';

 Form1.Panel2.Caption:= '0';

 RegIni:= TRegistry.Create;

 RegIni.RootKey:= HKEY\_LOCAL\_MACHINE;

 RegIni.OpenKey('Software\Naladchik', true);

 if not RegIni.KeyExists('a') then // Если нет этого ключа то прописываем.

 begin

 RegIni.CreateKey('a');

 RegIni.WriteInteger('a', 100);

 RegIni.WriteString('Versia', Application.Title);

 RegIni.WriteInteger('b', nach\_period); // в формате 00000

 RegIni.WriteInteger('c', nach\_period + 30); // в формате 00000

 RegIni.CloseKey;

 RegIni.Free;

 end;

 RegIni:= TRegistry.Create;

 RegIni.RootKey:= HKEY\_LOCAL\_MACHINE;

 RegIni.OpenKey('Software\Naladchik', false);

 reestra:= RegIni.ReadInteger('a');

 reestrversia:= RegIni.ReadString('Versia');

 reestrb:= RegIni.ReadInteger('b');

 reestrc:= RegIni.ReadInteger('c');

 if proverka\_sushesvovania\_ini = 2 then

 begin

 reestrMy\_Code:= RegIni.ReadInteger('d');

 reestrReg\_Code:= RegIni.ReadInteger('e');

 end;

 RegIni.CloseKey;

 RegIni.Free;

 if Hac > nach\_period then nach\_period:= Hac // защита от перевода даты

 else Hac:= nach\_period;

 Ini:= TIniFile.Create('Nal.ini');

 ini.WriteInteger('Hac', 'Hac', Hac);

 ini.Free;

 // Code\_My и Code\_Reg из ini файла

 if (Code\_My = My\_Code) and (Code\_Reg = Reg\_Code) and (reestrMy\_Code = My\_Code) and (reestrReg\_Code = Reg\_Code)

 then Form1.Caption:= Application.Title + '(Зарегистрированная версия)'

 else

 begin

 if reestrc > nach\_period then

 Form1.Caption:= Application.Title + ' (Осталось '+IntToStr(reestrc - nach\_period)+' дней)';

 if reestrc <= nach\_period then

 begin

 ShowMessage('Демонстрационный период закончен' + #13 + 'За регистрацией обратитесь к разработчику'+#13'dimasoft1@mail.ru');

 Halt(1);

 end;

 end;

end;

procedure TForm1.ToolButton1Click(Sender: TObject);

var

 kol: integer;

begin

 Indikator.Position:= 0;

 Form1.PanelSostoyaniya.Panels[1].Text:= '';

 Memo1.Clear;

 if OpenDialog1.Execute then

 begin

 Memo1.Lines.LoadFromFile(OpenDialog1.FileName);

 label1.Caption:= OpenDialog1.FileName;

 kol:= Memo1.Lines.Count;

 Panel2.Caption:= '0';

 Panel1.Caption:= IntToStr(kol - 1);

 end;

 Form2.PC\_VM.Enabled:= true;

 Form2.PC\_MAYAK.Enabled:= true;

 Form2.VM\_MAYAK.Enabled:= true;

 Form2.MAYAK\_VM.Enabled:= true;

 Form2.FormActivate(Sender);

end;

procedure TForm1.ParametryClick(Sender: TObject);

begin

 Form2.ShowModal;

end;

procedure TForm1.ToolButton2Click(Sender: TObject);

begin

 if SaveDialog1.Execute then

 begin

 Memo1.Lines.SaveToFile(SaveDialog1.FileName);

 Memo1.Modified:= false;

 end;

end;

procedure TForm1.ToolButton3Click(Sender: TObject); // Печать

var

 line: System.TextFile;

 i: integer;

begin

 if PrintDialog1.Execute then

 begin

 AssignPrn(line);

 rewrite(line);

 Printer.Canvas.Font:= Memo1.Font;

 for i:= 0 to memo1.Lines.Count-1 do writeln(line,' ', Memo1.Lines[i]);

 System.CloseFile(line);

 end;

end;

procedure TForm1.VersiaClick(Sender: TObject);

begin

 AboutBox.ShowModal;

end;

procedure TForm1.VyhodClick(Sender: TObject);

begin

 Form1.Close;

end;

procedure TForm1.SaveClick(Sender: TObject);

begin

 ToolButton2Click(Sender);

end;

procedure TForm1.OpenClick(Sender: TObject);

begin

 ToolButton1Click(Sender);

end;

procedure TForm1.PrintClick(Sender: TObject);

begin

 ToolButton3Click(Sender);

end;

procedure TForm1.CancelClick(Sender: TObject);

begin

 Form1.Close;

end;

procedure NaytiKoordinatu(kadr: string; koordinata: char; var poluchenaya: string);

var

 p : integer;

 poz\_10: string[100];

 per : string[10];

 r: real;

begin

 per:=' ';

 kadr:= Concat(kadr, ' ');

 nach:= pos(koordinata,kadr);

 nach:= Succ(nach);

 poz\_10:= copy(kadr, nach, 100);

 val(poz\_10,r,oshibka);

 for p:=1 to oshibka - 1 do

 per[p]:= poz\_10[p];

 if poz\_10 = ' ' then oshibka:= 1;

 poluchenaya:= TrimRight(per);

end;

function DobavitTochku( f: string): string; // Добавляет точку, если её нет

begin

 if pos('.', f) = 0

 then f:= Concat(f, '.');

 Result:= f;

end;

procedure Podhod\_G41\_G42\_radius;

begin // Х и Y уже найдены

 if pos('G3', Stroka.Strings[Succ(m)]) <> 0 then

 begin

 if (I > 0) and (J >= 0) then

 begin

 I\_sled:= 1;

 J\_sled:= -1;

 exit;

 end;

 if (I < 0) and (J <= 0) then

 begin

 I\_sled:= -1;

 J\_sled:= 1;

 exit;

 end;

 if (I <= 0) and (J > 0) then

 begin

 I\_sled:= 1;

 J\_sled:= 1;

 exit;

 end;

 if (I >= 0) and (J < 0) then

 begin

 I\_sled:= -1;

 J\_sled:= -1;

 exit;

 end;

 end;

 if pos('G2', Stroka.Strings[Succ(m)]) <> 0 then

 begin

 if (I >= 0) and (J > 0) then

 begin

 I\_sled:= -1;

 J\_sled:= 1;

 exit;

 end;

 if (I <= 0) and (J < 0) then

 begin

 I\_sled:= 1;

 J\_sled:= -1;

 exit;

 end;

 if (I < 0) and (J >= 0) then

 begin

 I\_sled:= -1;

 J\_sled:= -1;

 exit;

 end;

 if (I > 0) and (J <= 0) then

 begin

 I\_sled:= 1;

 J\_sled:= 1;

 exit;

 end;

 end;

end;

 // Преобразует строку, удаляет и вставляет подстроку ВМ

function preobrazovanieVM( vhod\_v\_preobrazovanie: real ;var kadr: string): string;

var

 stroka: string[10];

begin

 delete(kadr,nach,oshibka-1);

 stroka:= FloatToStr(vhod\_v\_preobrazovanie); // Преобразуем в число строку

 insert(stroka, kadr, nach);

end;

 // Преобразует строку, удаляет и вставляет подстроку МАЯК

function preobrazovanieMAYAK( vhod\_v\_preobrazovanie: real ;var kadr: string): string;

var

 stroka: string[10];

begin

 delete(kadr,nach,oshibka-1);

 if vhod\_v\_preobrazovanie <> 0 then

 begin

 stroka:= FloatToStrF(vhod\_v\_preobrazovanie, fffixed, 12, 3); // Преобразуем в число строку

 vhod\_v\_preobrazovanie:= StrToFloat(stroka); // Был глюк при конвертации

 stroka:= FloatToStr(vhod\_v\_preobrazovanie); // Сейчас вроде работает

 stroka:= DobavitTochku(stroka); // Добавляет точку, если её нет

 end

 else stroka:= '0'; // вставляем без точки

 insert(stroka, kadr, nach);

end;

procedure Podhod\_G41\_G42\_line;

var

 str: String;

begin

 str:= stroka.Strings[Succ(m)];

 if Pos('X', str) <> 0 then

 begin

 NaytiKoordinatu(str, 'X', str);

 X\_sled:= StrToFloat(str);

 end

 else

 begin

 if (Form2.PC\_VM.Checked) or (Form2.MAYAK\_VM.Checked) then

 X\_sled:= X\_nast / 1000

 else

 X\_sled:= X\_nast;

 end;

 str:= stroka.Strings[Succ(m)];

 if Pos('Y', str) <> 0 then

 begin

 NaytiKoordinatu(str, 'Y', str);

 Y\_sled:= StrToFloat(str);

 end

 else

 begin

 if (Form2.PC\_VM.Checked) or (Form2.MAYAK\_VM.Checked) then

 Y\_sled:= Y\_pred / 1000

 else

 Y\_sled:= Y\_nast;

 end;

 if (Form2.PC\_VM.Checked) or (Form2.MAYAK\_VM.Checked) then

 begin

 X\_sled:= X\_sled \* 1000;

 Y\_sled:= Y\_sled \* 1000;

 X\_sled:= X\_sled - X\_nast;

 Y\_sled:= Y\_sled - Y\_nast;

 end

 else

 begin

 X\_sled:= X\_sled - X\_nast;

 Y\_sled:= Y\_sled - Y\_nast;

 end;

end;

function FindVstavki(koordinata: char; kadr: string): integer;

begin

 case koordinata of

 'X': begin

 if pos('Y', kadr) <> 0 then Result:= pos('Y', kadr)

 else if pos('I', kadr) <> 0 then Result:= pos('I', kadr)

 else if pos('J', kadr) <> 0 then Result:= pos('J', kadr)

 else if pos('F', kadr) <> 0 then Result:= pos('F', kadr)

 else if pos('M', kadr) <> 0 then Result:= pos('M', kadr)

 else Result:= Length(kadr) + 1;

 end;

 'Y': begin

 if pos('I', kadr) <> 0 then Result:= pos('I', kadr)

 else if pos('J', kadr) <> 0 then Result:= pos('J', kadr)

 else if pos('F', kadr) <> 0 then Result:= pos('F', kadr)

 else if pos('M', kadr) <> 0 then Result:= pos('M', kadr)

 else Result:= Length(kadr) + 1;

 end;

 'I': begin

 if pos('J', kadr) <> 0 then Result:= pos('J', kadr)

 else if pos('F', kadr) <> 0 then Result:= pos('F', kadr)

 else if pos('M', kadr) <> 0 then Result:= pos('M', kadr)

 else Result:= Length(kadr) + 1;

 end;

 'J': begin

 if pos('F', kadr) <> 0 then Result:= pos('F', kadr)

 else if pos('M', kadr) <> 0 then Result:= pos('M', kadr)

 else Result:= Length(kadr) + 1;

 end;

 else

 Result:= Length(kadr) + 1;

 end;

end;

procedure TForm1.ObrabotkaClick(Sender: TObject);

label

 metka;

var

 kol, buf: integer;

 Proc, proci, u: integer;

 vyh: string;

 PolChisla\_1Proc, uvel: real;

 bufer: string;

 Priz\_pusto\_memo: boolean;

procedure DelProbel(var kadr: string); // Удаление пробелов

var

 k: integer;

begin

 while pos(' ', kadr) <> 0 do

 begin

 k:= pos(' ', kadr);

 delete(kadr, k, 1);

 end;

end;

procedure pustostroka;

begin

 stroka.Delete(m); // Удаляем пустую строку

 Dec(kol); // Уменьшаем кол-во строк на 1

 Dec(m); // Уменьшаем на одну стороку

end;

procedure nomerkadra(var kadr: string);

var

 error, nach, x: integer;

 poz\_N7: string [10];

begin

 poz\_N7:= ' ';

 nach:= pos('N',kadr); // Номер позиции 'N'

 if nach <> 0 then // Позиция найдена

 begin

 Inc(nach); // Наращиваем на 1

 poz\_N7:= copy(kadr, nach, 10);

 val(poz\_N7,x,error); // проверяем на ошибку

 if error = 1 then

 else

 begin

 delete(kadr,nach,error-1); // Удаляем старое значение

 Inc(buf); // Наращиваем кадр на 1

 insert(IntToStr(buf), kadr, nach); // Вставляем новое число

 end;

 end;

end;

Procedure PC\_MAYAK(var kadr: string);

var

 I\_est: boolean;

 st: string;

 s: string[10];

begin

 if (pos('%', kadr) <> 0) or (pos('$', kadr) <> 0) then exit;

 I\_est:= False;

 if (pos('I',kadr) <> 0) or (pos('J',kadr) <> 0) then I\_est:= true;

 if pos('X',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'X', vyh);

 if oshibka <> 1 then

 begin

 X\_nast:= StrToFloat(vyh);

 PreobrazovanieMayak(X\_nast, kadr); // не было точки

 end;

 end

 else

 if I\_est then // Добавление Х Если нет

 begin

 uslovie:= FindVstavki('X', kadr);

 s:= FloatToStr(X\_nast);

 if X\_nast <> 0 then s:= DobavitTochku(s);

 insert(Concat('X', s), kadr, uslovie);

 end;

 if pos('Y',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'Y', vyh);

 if oshibka <> 1 then

 begin

 Y\_nast:= StrToFloat(vyh);

 PreobrazovanieMayak(Y\_nast, kadr) // не было точки

 end;

 end

 else

 if I\_est then // Добавление Y Если нет

 begin

 uslovie:= FindVstavki('Y', kadr);

 s:= FloatToStr(Y\_nast);

 if Y\_nast <> 0 then s:= DobavitTochku(s);

 insert(Concat('Y',s), kadr, uslovie);

 end; // Подходы

 if (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0) then

 begin

 try

 s:= Stroka.Strings[Succ(m)]; // ошибка нет строки вообще

 except // так не должно быть это ошибка оператора

 on EStringListError do // добавляем пустую строку и исключаем ошибку

 stroka.Add('');

 end;

 if pos('X', kadr) = 0 then

 begin // Вставка Х и Y если нет

 uslovie:= FindVstavki('X', kadr);

 s:= FloatToStr(X\_nast); // В позиции X Y I J F M

 if X\_nast <> 0 then s:= DobavitTochku(s);

 Insert('X' + s, kadr, uslovie);

 end;

 if pos('Y', kadr) = 0 then

 begin

 uslovie:= FindVstavki('Y', kadr);

 s:= FloatToStr(Y\_nast);

 if Y\_nast <> 0 then s:= DobavitTochku(s);

 Insert('Y' + s, kadr, uslovie);

 end;

 if (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0)

 then

 begin

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'I', st);

 I:= StrToFloat(st);

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'J', st);

 J:= StrToFloat(st);

 Podhod\_G41\_G42\_radius;

 J:= J + Y\_nast;

 if (I\_sled = -1) and (J > 0) then J:= -J; // ЛОГИЧЕСКИЙ ПЕРЕСЧЁТ

 if (I\_sled = 1) and (J < 0) then J:= -J;

 I:= I + X\_nast;

 if(J\_sled = -1) and (I > 0) then I:= -I;

 if(J\_sled = 1) and (I < 0) then I:= -I;

 if pos('J', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieMayak(I, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(I), kadr, uslovie);

 end;

 if pos('I', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieMayak(J, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(J), kadr, uslovie);

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

 exit;

 end

 else

 begin

 Podhod\_G41\_G42\_line; // ЛИНЕЙНЫЕ ПОДХОДЫ РАСЧЁТ

 if pos('I', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieMayak(X\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(X\_sled) + '.', kadr, uslovie);

 end;

 if pos('J', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieMayak(Y\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(Y\_sled) + '.', kadr, uslovie);

 end;

 exit;

 end;

 end;

 uslovie:= pos('I',kadr); // расчёт радиусов

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'I',vyh);

 if oshibka = 1 then

 else

 begin

 I:= StrToFloat(vyh);

 I:= X\_pred + I;

 preobrazovanieMAYAK(I,kadr);

 end;

 end;

 uslovie:= pos('J',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'J',vyh);

 if oshibka = 1 then

 else

 begin

 J:= StrToFloat(vyh);

 J:= Y\_pred + J;

 preobrazovanieMAYAK(J,kadr);

 end;

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

end;

Function VM\_MAYAK(var kadr: string): string;

var

 I\_est: boolean;

 s, st: string;

begin

 if (pos('%', kadr) <> 0) or (pos('$', kadr) <> 0) then exit;

 I\_est:= False;

 if (pos('I',kadr) <> 0) or (pos('J',kadr) <> 0) then I\_est:= true;

 uslovie:= pos('X',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'X', vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 X\_nast:= StrToFloat(vyh) / 1000;

 preobrazovanieMAYAK(X\_nast,kadr);

 end;

 end

 else

 if I\_est = true then // Добавление Х Если нет

 begin

 uslovie:= pos('Y',kadr);

 insert('X'+ vyh, kadr, uslovie);

 end;

 uslovie:= pos('Y',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'Y', vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 Y\_nast:= StrToFloat(vyh) / 1000;

 preobrazovanieMAYAK(Y\_nast,kadr);

 end;

 end

 else

 if I\_est = true then // Добавление Y Если нет

 begin

 uslovie:= pos('I',kadr);

 insert('Y' + vyh, kadr, uslovie);

 end;

 if (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0) then

 begin

 try

 s:= Stroka.Strings[Succ(m)]; // ошибка нет строки вообще

 except

 on EStringListError do // добавляем пустую строку и исключаем ошибку

 stroka.Add('');

 end;

 if pos('X', kadr) = 0 then

 begin // Вставка Х и Y если нет

 uslovie:= FindVstavki('X', kadr);

 s:= FloatToStr(X\_nast); // В позиции X Y I J F M

 if X\_nast <> 0 then s:= DobavitTochku(s);

 Insert('X' + s, kadr, uslovie);

 end;

 if pos('Y', kadr) = 0 then

 begin

 uslovie:= FindVstavki('Y', kadr);

 s:= FloatToStr(Y\_nast);

 if Y\_nast <> 0 then s:= DobavitTochku(s);

 Insert('Y' + s, kadr, uslovie);

 end;

 if (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0)

 then

 begin

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'I', st);

 I:= StrToFloat(st);

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'J', st);

 J:= StrToFloat(st);

 Podhod\_G41\_G42\_radius;

 J:= J / 1000;

 J:= J + Y\_nast;

 if (I\_sled = -1) and (J > 0) then J:= -J;

 if (I\_sled = 1) and (J < 0) then J:= -J;

 I:= I / 1000;

 I:= I + X\_nast;

 if(J\_sled = -1) and (I > 0) then I:= -I;

 if(J\_sled = 1) and (I < 0) then I:= -I;

 if pos('J', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieMayak(I, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(I), kadr, uslovie);

 end;

 if pos('I', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieMayak(J, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(J), kadr, uslovie);

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

 exit;

 end

 else

 begin // Линейный подход

 st:= Stroka.Strings[Succ(m)];

 uslovie:= Pos('X', st);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(st, 'X', st);

 X\_sled:= StrToFloat(st) / 1000;

 X\_sled:= X\_sled - X\_nast;

 end

 else X\_sled:= 0; // Если нет в кадре то 0

 st:= stroka.Strings[Succ(m)];

 uslovie:= Pos('Y', st);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(st, 'Y', st);

 Y\_sled:= StrToFloat(st) / 1000;

 Y\_sled:= Y\_sled - Y\_nast;

 end

 else Y\_sled:= 0;

 if pos('I', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieMayak(X\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(X\_sled) + '.', kadr, uslovie);

 end;

 if pos('J', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieMayak(Y\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(Y\_sled) + '.', kadr, uslovie);

 end;

 exit;

 end;

 end;

 uslovie:= pos('Z',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'Z', vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 Z:= StrToFloat(vyh) / 1000;

 preobrazovanieMAYAK(Z,kadr);

 end;

 end;

 uslovie:= pos('I',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'I',vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 I:= StrToFloat(vyh) / 1000;

 I:= X\_pred + I;

 preobrazovanieMAYAK(I,kadr);

 end;

 end;

 uslovie:= pos('J',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'J',vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 J:= StrToFloat(vyh) / 1000;

 J:= Y\_Pred + J;

 preobrazovanieMAYAK(J,kadr);

 end;

 end;

 uslovie:= pos('U',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'U', vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 Uvmmayak:= StrToFloat(vyh) / 1000;

 preobrazovanieMAYAK(Uvmmayak,kadr);

 end;

 end;

 uslovie:= pos('K',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'K', vyh);

 if oshibka = 1 then

 else

 begin

 vyh:= TrimRight(vyh); // Удаляем лишние пробелы справа

 K:= StrToFloat(vyh) / 1000;

 preobrazovanieMAYAK(K,kadr);

 end;

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

end;

Function MAYAK\_VM(var kadr: string): string;

var

 I\_est: boolean;

 uslovie: integer;

 strok: string[10];

 s, st: string;

begin

 if pos('%', kadr) <> 0 then exit;

 if pos('$', kadr) <> 0 then exit;

 I\_est:= False;

 uslovie:= pos('I',kadr);

 if uslovie <> 0 then I\_est:= true;

 uslovie:= pos('X',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'X',vyh);

 if oshibka = 1 then

 else

 begin

 X\_nast:= StrToFloat(vyh) \* 1000;

 X\_nast:= Int(X\_nast); // Отбрасываем дробную часть

 preobrazovanieVM(X\_nast,kadr);

 end;

 end

 else

 if I\_est then

 begin

 uslovie:= pos('Y',kadr);

 strok:= FloatToStr(X\_pred); // Преобразуем в число строку

 insert('X' + strok, kadr, uslovie);

 end;

 uslovie:= pos('Y',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'Y',vyh);

 if oshibka = 1 then

 else

 begin

 Y\_nast:= StrToFloat(vyh) \* 1000;

 Y\_nast:= Int(Y\_nast); // Отбрасываем дробную часть

 preobrazovanieVM(Y\_nast,kadr);

 end;

 end

 else

 if I\_est then

 begin

 uslovie:= pos('I',kadr);

 strok:= FloatToStr(Y\_pred); // Преобразуем в число строку

 insert('Y' + strok, kadr, uslovie);

 end;

 if (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0) then

 begin

 try

 s:= Stroka.Strings[Succ(m)]; // ошибка нет строки вообще

 except

 on EStringListError do // добавляем пустую строку и исключаем ошибку

 stroka.Add('');

 end;

 if pos('X', kadr) = 0 then

 begin // Вставка Х и Y если нет

 uslovie:= FindVstavki('X', kadr); // В позиции X Y I J F M

 Insert('X' + FloatToStr(X\_nast), kadr, uslovie);

 end;

 if pos('Y', kadr) = 0 then

 begin

 uslovie:= FindVstavki('Y', kadr);

 Insert('Y' + FloatToStr(Y\_nast), kadr, uslovie);

 end;

 if (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0)

 then

 begin

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'I', st);

 I:= StrToFloat(st);

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'J', st);

 J:= StrToFloat(st);

 J:= J \* 1000;

 I:= I \* 1000;

 I:= I - X\_nast;

 J:= J - Y\_nast;

 Podhod\_G41\_G42\_radius;

 if (I\_sled = -1) and (J > 0) then J:= -J;

 if (I\_sled = 1) and (J < 0) then J:= -J;

 if(J\_sled = -1) and (I > 0) then I:= -I;

 if(J\_sled = 1) and (I < 0) then I:= -I;

 if pos('J', kadr) <> 0

 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieVM(I, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(I), kadr, uslovie);

 end;

 if pos('I', kadr) <> 0

 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieVM(J, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(J), kadr, uslovie);

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

 exit;

 end

 else

 begin

 Podhod\_G41\_G42\_line;

 if pos('I', kadr) <> 0

 then

 begin

 NaytiKoordinatu(kadr, 'I', st);

 PreobrazovanieVM(X\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(X\_sled), kadr, uslovie);

 end;

 if pos('J', kadr) <> 0

 then

 begin

 NaytiKoordinatu(kadr, 'J', st);

 PreobrazovanieVM(Y\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(Y\_sled), kadr, uslovie);

 end;

 exit;

 end;

 end;

 uslovie:= pos('I',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'I',vyh);

 if oshibka = 1 then

 else

 begin

 I:= StrToFloat(vyh) \* 1000;

 I:= Int(I); // Отбрасываем дробную часть

 I:= I - X\_pred;

 preobrazovanieVM(I,kadr);

 end;

 end;

 uslovie:= pos('J',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'J',vyh);

 if oshibka = 1 then

 else

 begin

 J:= StrToFloat(vyh) \* 1000;

 J:= Int(J); // Отбрасываем дробную часть

 J:= J - Y\_pred;

 preobrazovanieVM(J,kadr);

 end;

 end;

 uslovie:= pos('Z',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'Z',vyh);

 if oshibka = 1 then

 else

 begin

 Z:= StrToFloat(vyh) \* 1000;

 Z:= Int(Z); // Отбрасываем дробную часть

 preobrazovanieVM(Z,kadr);

 end;

 end;

 uslovie:= pos('A',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'A',vyh);

 if oshibka = 1 then

 else

 begin

 A:= StrToFloat(vyh) \* 1000;

 A:= Int(A); // Отбрасываем дробную часть

 preobrazovanieVM(A,kadr);

 Delete(kadr, uslovie, 1);

 Insert('U',kadr,uslovie);

 end;

 end;

 uslovie:= pos('K',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'K',vyh);

 if oshibka = 1 then

 else

 begin

 K:= StrToFloat(vyh) \* 1000;

 K:= Int(K); // Отбрасываем дробную часть

 preobrazovanieVM(K,kadr);

 end;

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

end;

procedure PC\_VM(var kadr: string);

var

 uslovie: integer;

 I\_est: boolean;

 s, st: string;

begin

 if (pos('%', kadr) <> 0) or (pos('$', kadr) <> 0) then exit;

 I\_est:= False;

 if (pos('I',kadr) <> 0) or (pos('J',kadr) <> 0) then I\_est:= true;

 if pos('X',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'X',vyh);

 if oshibka = 1 then

 else

 begin

 X\_nast:= StrToFloat(vyh) \* 1000;

 X\_nast:= Int(X\_nast); // Отбрасываем дробную часть

 preobrazovanieVM(X\_nast,kadr);

 end;

 end

 else

 if I\_est then

 begin

 uslovie:= pos('Y',kadr); // Преобразуем в число строку

 insert('X' + FloatToStr(X\_pred), kadr, uslovie);

 end;

 if pos('Y',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'Y',vyh);

 if oshibka = 1 then

 else

 begin

 Y\_nast:= StrToFloat(vyh) \* 1000;

 Y\_nast:= Int(Y\_nast); // Отбрасываем дробную часть

 preobrazovanieVM(Y\_nast,kadr);

 end;

 end

 else

 if I\_est then

 begin

 uslovie:= pos('I',kadr); // Преобразуем в число строку

 insert('Y' + FloatToStr(Y\_pred), kadr, uslovie);

 end;

 if (pos('G42', kadr) <> 0) or (pos('G41', kadr) <> 0) then

 begin

 try

 s:= Stroka.Strings[Succ(m)]; // ошибка нет строки вообще

 except

 on EStringListError do // добавляем пустую строку и исключаем ошибку

 stroka.Add('');

 end;

 if pos('X', kadr) = 0 then

 begin // Вставка Х и Y если нет

 NaytiKoordinatu(Stroka.Strings[Succ(m)], 'X', s);

 if oshibka <> 1 then

 begin

 uslovie:= FindVstavki('X', kadr);

 s:= TrimRight(s);

 X\_nast:= StrToFloat(s) \* 1000; // В позиции X Y I J F M

 Insert('X' + FloatToStr(X\_nast), kadr, uslovie);

 end

 else

 begin

 X\_nast:= X\_pred;

 uslovie:= FindVstavki('X', kadr); // В позиции X Y I J F M

 Insert('X' + FloatToStr(X\_nast), kadr, uslovie);

 end;

 end;

 if pos('Y', kadr) = 0 then

 begin

 uslovie:= FindVstavki('Y', kadr);

 NaytiKoordinatu(Stroka.Strings[Succ(m)], 'Y', s);

 s:= TrimRight(s);

 Y\_nast:= StrToFloat(s) \* 1000;

 Insert('Y' + FloatToStr(Y\_nast), kadr, uslovie);

 end;

 if (pos('I', Stroka.Strings[Succ(m)]) <> 0) and (pos('J', Stroka.Strings[Succ(m)]) <> 0)

 then

 begin

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'I', st);

 I:= StrToFloat(st);

 Naytikoordinatu(Stroka.Strings[Succ(m)], 'J', st);

 J:= StrToFloat(st);

 Podhod\_G41\_G42\_radius;

 J:= J \* 1000;

 J:= Int(J);

 if (I\_sled = -1) and (J > 0) then J:= -J;

 if (I\_sled = 1) and (J < 0) then J:= -J;

 if Pos('I', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', s);

 PreobrazovanieVM(J, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('I', kadr);

 Insert('I' + FloatToStr(J), kadr, uslovie);

 end;

 I:= I \* 1000;

 I:= Int(I);

 if(J\_sled = -1) and (I > 0) then I:= -I;

 if(J\_sled = 1) and (I < 0) then I:= -I;

 if Pos('J', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', s);

 PreobrazovanieVM(I, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('J', kadr);

 Insert('J' + FloatToStr(I), kadr, uslovie);

 end;

 Exit;

 end

 else

 begin

 Podhod\_G41\_G42\_line;

 X\_sled:= Int(X\_sled);

 if Pos('X', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'I', s);

 PreobrazovanieVM(X\_sled, kadr);

 end

 else

 begin

 uslovie:= FindVstavki('Y', kadr);

 Insert(FloatToStr(X\_sled), kadr, uslovie);

 end;

 Y\_sled:= Int(Y\_sled);

 if Pos('Y', kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr, 'J', s);

 PreobrazovanieVM(Y\_sled, kadr);

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

 exit;

 end

 else

 begin

 uslovie:= FindVstavki('X', kadr);

 Insert(FloatToStr(Y\_sled), kadr, uslovie);

 end;

 end;

 end;

 if pos('I',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'I',vyh);

 if oshibka = 1 then

 else

 begin

 I:= StrToFloat(vyh) \* 1000;

 I:= Int(I); // Отбрасываем дробную часть

 preobrazovanieVM(I,kadr);

 end;

 end;

 if pos('J',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'J',vyh);

 if oshibka = 1 then

 else

 begin

 J:= StrToFloat(vyh) \* 1000;

 J:= Int(J); // Отбрасываем дробную часть

 preobrazovanieVM(J,kadr);

 end;

 end;

 if pos('Z',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'Z',vyh);

 if oshibka = 1 then

 else

 begin

 Z:= StrToFloat(vyh) \* 1000;

 Z:= Int(Z); // Отбрасываем дробную часть

 preobrazovanieVM(Z,kadr);

 end;

 end;

 uslovie:= pos('A',kadr);

 if uslovie <> 0 then

 begin

 NaytiKoordinatu(kadr,'A',vyh);

 if oshibka = 1 then

 else

 begin

 A:= StrToFloat(vyh) \* 1000;

 A:= Int(A); // Отбрасываем дробную часть

 preobrazovanieVM(A,kadr);

 Delete(kadr, uslovie, 1);

 Insert('U',kadr,uslovie);

 end;

 end;

 if pos('K',kadr) <> 0 then

 begin

 NaytiKoordinatu(kadr,'K',vyh);

 if oshibka = 1 then

 else

 begin

 K:= StrToFloat(vyh) \* 1000;

 K:= Int(K); // Отбрасываем дробную часть

 preobrazovanieVM(K,kadr);

 end;

 end;

 X\_pred:= X\_nast;

 Y\_pred:= Y\_nast;

end;

 // НАЧАЛО ОСНОВНОЙ ПРОГРАММЫ obrabotka

begin

 if (Form1.NomerKadra.State = cbUnchecked) and (Form1.Probel.State = cbUnchecked) and

 (Form1.Pustostr.State = cbUnchecked) and (Form2.Ne\_obrabatyvat.Checked) then

 begin

 ShowMessage(' Обработка не задана ');

 Goto metka;

 end;

 Priz\_pusto\_memo:= false;

 kol:= Memo1.Lines.Count; // Количество строк

 if kol = 0 then // Если поле пусто

 begin

 ShowMessage(' В поле редактирования пусто ');

 Goto metka;

 end;

 memo1mod:= true;

 Panel1.Caption:= IntToStr(kol);

 buf:= 0; m:= 0; I:= 0; J:= 0;

 PolChisla\_1Proc:= kol div 100;

 uvel:= PolChisla\_1Proc;

 Indikator.Position:= 0;

 Proc:= Succ(Indikator.Position);

 u:= 100 div kol;

 Proci:= u;

 stroka:= TStringList.Create();

 stroka.Text:= Memo1.Text;

 Memo1.Clear;

 repeat

 bufer:= stroka.Strings[m];

 Panel2.Caption:= IntToStr(Succ(m));

 Panel2.Refresh;

 if (Form1.Probel.Checked)then

 DelProbel(bufer); // Удаляем пробелы

 if Form1.NomerKadra.Checked then // включен переключатель

 nomerkadra(bufer);

 if Form2.PC\_MAYAK.Checked then

 pc\_mayak(bufer);

 if Form2.PC\_VM.Checked then

 pc\_vm(bufer);

 if Form2.VM\_MAYAK.Checked then

 VM\_MAYAK(bufer);

 if Form2.MAYAK\_VM.Checked then

 Mayak\_vm(bufer);

 if kol >= 100 then

 begin

 if m >= PolChisla\_1Proc then

 begin

 Indikator.Position:= Proc;

 Form1.PanelSostoyaniya.Panels[1].Text:= ' Выполнено ' + IntToStr(Indikator.Position) + ' %';

 PanelSostoyaniya.Refresh; // Перерисовка панели

 Inc(proc);

 PolChisla\_1Proc:= PolChisla\_1Proc + uvel;

 end;

 end

 else

 begin

 Indikator.Position:= Proci;

 Form1.PanelSostoyaniya.Panels[1].Text:= ' Выполнено ' + IntToStr(Indikator.Position) + ' %';

 PanelSostoyaniya.Refresh; // Перерисовка панели

 Inc(proci, u);

 end;

 stroka.Strings[m]:= bufer; // Для токарного станка

 if Length(bufer) = 0 then

 if Form1.PustoStr.Checked then

 pustostroka;

 Bufer:= ''; // Очистка строки Буфер

 Inc(m); // Следующая строка (+1)

 until m >= kol;

 Panel1.Caption:= IntToStr(Stroka.Count); // Сколько реально осталось

 Form1.PanelSostoyaniya.Panels[1].Text:= ' Выполнено ' + '100' + ' %';

 Indikator.Position:= 100;

 if (Form2.PC\_MAYAK.Checked) or (Form2.PC\_VM.Checked) or (Form2.VM\_MAYAK.Checked) or (Form2.MAYAK\_VM.Checked) then

 begin

 Form2.PC\_VM.Enabled:= false;

 Form2.PC\_MAYAK.Enabled:= false;

 Form2.VM\_MAYAK.Enabled:= false;

 Form2.MAYAK\_VM.Enabled:= false;

 Form2.Ne\_obrabatyvat.Checked:= true;

 end;

 try // Ошибка если в Memo1 пробелы

 if Stroka.Strings[0] <> '' then // Вставить пустую строку если её нет

 Stroka.Insert(0, '');

 except

 on EStringListError do

 end;

 Memo1.Text:= Stroka.Text;

 stroka.Free;

 metka: // если Поле пустое то выход сюда

end;

procedure TForm1.ToolButton6Click(Sender: TObject); // Кнопка очистить

begin

 Memo1.Clear;

 Panel1.Caption:= '0';

 Panel2.Caption:= '0';

 Form2.PC\_VM.Enabled:= true;

 Form2.PC\_MAYAK.Enabled:= true;

 Form2.VM\_MAYAK.Enabled:= true;

 Form2.MAYAK\_VM.Enabled:= true;

 Label1.Caption:= '';

 Form2.FormActivate(Sender);

 Memo1.SetFocus;

 Indikator.Position:= 0;

end;

procedure TForm1.ToolButton4Click(Sender: TObject); // Кнопка параметры

begin

 Form2.ShowModal;

end;

procedure TForm1.Toolbutton9Click(Sender: TObject); // Кнопка найти и заменить

begin

 ReplaceDialog1.Execute;

end;

procedure TForm1.ReplaceDialog1Replace(Sender: TObject); // Кнопка найти и заменить

label

 10;

begin

 Memo1.HideSelection:= true;

 10:

 if pos(ReplaceDialog1.FindText, Memo1.Text) <> 0 then

 begin

 Memo1.SelStart:= pos(ReplaceDialog1.FindText, Memo1.Text) - 1 ;

 Memo1.SelLength:= Length(ReplaceDialog1.FindText);

 Memo1.SelText:= ReplaceDialog1.ReplaceText;

 goto 10;

 end;

 Memo1.HideSelection:= false;

end;

procedure TForm1.memo1KeyDown(Sender: TObject; var Key: Word;

 Shift: TShiftState);

begin

 Panel1.Caption:= IntToStr(Memo1.Lines.Count);

end;

procedure TForm1.ToolButton5Click(Sender: TObject); // Кнопка в верхний регистр

begin

 Memo1.Text:= UpperCase(Memo1.Text);

end;

procedure TForm1.ToolButton8Click(Sender: TObject);

begin

 Form5.Visible:= true;

end;

end.

**Файл Unit2.pas**

unit Unit2;

interface

uses

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

 Dialogs, StdCtrls, ComCtrls, ExtCtrls;

type

 TForm2 = class(TForm)

 PereschotProg: TGroupBox;

 PC\_MAYAK: TRadioButton;

 Ne\_obrabatyvat: TRadioButton;

 PC\_VM: TRadioButton;

 VM\_MAYAK: TRadioButton;

 MAYAK\_VM: TRadioButton;

 Button1: TButton;

 Mayk223mp: TRadioButton;

 procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

 procedure FormActivate(Sender: TObject);

 procedure Button1Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form2: TForm2;

implementation

uses Unit1;

{$R \*.dfm}

procedure Sostoyanie\_perekluchateley;

begin

 if Form2.Ne\_obrabatyvat.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.Ne\_obrabatyvat.Caption

 else

 if Form2.PC\_MAYAK.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.PC\_MAYAK.Caption

 else

 if Form2.PC\_VM.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.PC\_VM.Caption

 else

 if Form2.VM\_MAYAK.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.VM\_MAYAK.Caption

 else

 if Form2.MAYAK\_VM.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.MAYAK\_VM.Caption

 else

 if Form2.Mayk223mp.Checked then

 Form1.PanelSostoyaniya.Panels[0].Text:= st + Form2.Mayk223mp.Caption;

end;

procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

 Sostoyanie\_perekluchateley;

 If Mayk223mp.Checked then

 begin

 Form2.Close;

 end;

end;

procedure TForm2.FormActivate(Sender: TObject);

begin

 Sostoyanie\_perekluchateley;

end;

procedure TForm2.Button1Click(Sender: TObject);

begin

 Sostoyanie\_perekluchateley;

 Form2.Close;

end;

end.

**Файл Unit5.pas**

unit Unit5;

interface

uses

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

 Dialogs, ExtCtrls, StdCtrls;

type

 TForm5 = class(TForm)

 GroupBox1: TGroupBox;

 RadioButton1: TRadioButton;

 Edit1: TEdit;

 Label1: TLabel;

 ButtonObpabotka: TButton;

 ButtonNastroyka: TButton;

 procedure FormActivate(Sender: TObject);

 procedure ButtonObpabotkaClick(Sender: TObject);

 procedure ButtonNastroykaClick(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form5: TForm5;

implementation

uses unit1, unit6;

{$R \*.dfm}

procedure TForm5.FormActivate(Sender: TObject);

begin

 Form5.Caption:= Application.Title;

end;

procedure TForm5.ButtonObpabotkaClick(Sender: TObject);

var

 stroka, stroka1: TStrings;

 Name\_File, kol\_v\_stroka: integer;

 chast, chot, ObshiySchot: integer;

 st, name: WideString;

begin

 stroka:= TStringList.Create();

 stroka1:= TStringList.Create();

 chot:= 0; Name\_File:= 1; chast:= 0;

 chast:= StrToInt(Edit1.Text) \* 955;

 kol\_v\_stroka:= 0;

 ObshiySchot:= 0;

 stroka1.Text:= Form1.memo1.Text;

 repeat

 try

 st:= stroka1.Strings[ObshiySchot];

 except

 on EStringListError do

 begin

 Stroka.Free;

 stroka1.Free;

 Form1.memo1.Text:= ' Ошибка в редакторе: Возможно в поле редактирования пусто или одни пробелы';

 Form5.Close;

 Exit;

 end;

 end;

 if chot <= chast then

 begin

 stroka.Insert(kol\_v\_stroka, st);

 Inc(chot, Length(st)); // Увеличиваем на длину строки

 Inc(kol\_v\_stroka);

 Inc(ObshiySchot);

 end

 else

 begin

 name:= Concat('E:\Наладчик\Пров\', IntToStr(Name\_File), '.txt');

 stroka.SaveToFile(name);

 stroka.Clear;

 chot:= 0;

 kol\_v\_stroka:= 0;

 Name\_File:= Succ(Name\_File);

 end;

 if (stroka1.Count = ObshiySchot) and (Length(stroka.Text) <> 0) then

 begin

 name:= Concat('E:\Наладчик\Пров\', IntToStr(Name\_File), '.txt');

 stroka.SaveToFile(name);

 end;

 until stroka1.Count = ObshiySchot;

 stroka.Free;

 stroka1.Free;

 Form5.Close;

 ShowMessage(' Обработка завершена ');

end;

procedure TForm5.ButtonNastroykaClick(Sender: TObject);

var

 NameF: WideString;

begin

 Form6.Visible:= true;

end;

end.

**Файл Unit6.pas**

unit Unit6;

interface

uses

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

 Dialogs, ExtCtrls, StdCtrls, Registry, FileCtrl;

type

 TForm6 = class(TForm)

 Panel1: TPanel;

 Memo1: TMemo;

 ButtonRedEndFile: TButton;

 ButtonRedPapku: TButton;

 Label1: TLabel;

 SaveDialog1: TSaveDialog;

 procedure FormActivate(Sender: TObject);

 procedure ButtonRedPapkuClick(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form6: TForm6;

 RegIni: TRegistry;

implementation

uses unit1;

{$R \*.dfm}

procedure TForm6.FormActivate(Sender: TObject);

begin

 Form6.Caption:= Application.Title;

 RegIni:= TRegistry.Create;

 RegIni.RootKey:= HKEY\_LOCAL\_MACHINE;

 RegIni.OpenKey('Software\Naladchik', false);

 Label1.Caption:= RegIni.ReadString('pach');

 RegIni.CloseKey;

 RegIni.Free;

end;

procedure TForm6.ButtonRedPapkuClick(Sender: TObject);

begin

 if SaveDialog1.Execute then

 begin

 Label1.Caption:= SaveDialog1.FileName;

 RegIni:= TRegistry.Create;

 RegIni.RootKey:= HKEY\_LOCAL\_MACHINE;

 RegIni.OpenKey('Software\Naladchik', true);

 RegIni.CreateKey('a');

 RegIni.WriteString('pach', SaveDialog1.FileName);

 RegIni.CloseKey;

 RegIni.Free;

 end;

end;

end.

# ПРИЛОЖЕНИЕ 2

# РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

**Главное окно программы**

При запуске программы, появляется главное окно, представленное на рис.П.2.1.

 Рис.П. 2.1

При нажатии кнопки «Открыть», открывается диалоговое окно выбора файла рис. П.2.2.



Рис.П.2.2

Выбираем обработку рис.П.2.3. и нажимаем «Закрыть»

 Рис.П.2.3

Нажимаем кнопку «Обработка» рис.П.2.4.

Рис.П.2.4

Всё программа полностью обработана заданным методом обработки

# ПРИЛОЖЕНИЕ 3

# РУКОВОДСТВО ПРОГРАММИСТА

## 3.1. Назначение программы

 Программа «Наладчик» предназначена для конвертирования исходного текста программ для станков с ЧПУ. И предназначена для использования на производстве в различных областях.

 Программа позволяет:

1. конвертировать программный текст в любою из доступных программе систем;
2. удалять лишние пробелы;
3. удалять пустые строки;
4. устанавливать периодическую нумерацию кадров;
5. задавать шрифт(цвет, размер, тип);
6. печать документа;
7. визуальный контроль обработки;
8. менять регистр букв к заглавным(что требует синтаксис программ);
9. разбивать программу на заданное количество частей;
10. осуществляет поиск и замену по образцу.

## 3.2. Условия применения программы

Разработанная программа является приложением MS Windows. Для ее работы требуется наличие на компьютере операционной системы MS Windows NT4.0/2000/XP и все вытекающее отсюда техническое обеспечение.

Кроме того, требуется:

1. физическая память 1Mб + память на используемые и создаваемые текстовые файлы (не менее 5 Мб);
2. оперативная память 2\*размер текстового файла. При нехватке оперативной памяти может использоваться виртуальная память. А за счет того, что все данные находятся в памяти, сильно сокращается время решения задач.

Для написания программы использовалась среда разработки Borland Delphi 7.

## 3.3. Характеристики программы

После запуска программы пользователь может запускать решение задачи в следующей последовательности:

- выбор (открытие) текстового файла;

- выбор метода конвертирования;

- выбор шрифта;

- выбор дополнительных методов обработки

## 3.4. Обращение к программе

Запустить программу можно из оболочки Windows, посредством запуска файла **Наладчик.ехе**

## 3.5. Входные и выходные данные

В качестве входных и выходных данных используются текстовые файлы.

## 3.6 Сообщения

Сообщения, выдаваемые оператору приведены в табл. П.3.1.

 Таб.П.3.1

|  |  |  |
| --- | --- | --- |
| Вид сообщения | Описание | Действие |
| В поле редактирования пусто | В поле редактирования нет ни какого текста | Введите текст в поле редактирования |
| Обработка не задана | Не поставлена ни одна галочка задач обработки | Задайте обработку |

#  ПРИЛОЖЕНИЕ 4

# РУКОВОДСТВО ОПЕРАТОРА

## 4.1. Назначение программы

Программа «Наладчик» предназначена для конвертирования исходного текста программ для станков с ЧПУ. И предназначена для использования на производстве в различных областях.

 Программа позволяет:

1. конвертировать программный текст в любою из доступных программе систем;
2. удалять лишние пробелы;
3. удалять пустые строки;
4. восстанавливать периодическую нумерацию кадров;
5. задавать шрифт(цвет, размер, тип);
6. печать документа;
7. визуальный контроль обработки;
8. менять регистр букв к заглавным(что требует синтаксис программ);
9. разбивать программу на заданное количество частей;
10. осуществляет поиск и замену по образцу.

## 4.2. Условия выполнения программы

Разработанная программа является приложением MS Windows. Для ее работы требуется наличие на компьютере операционной системы MS Windows 98/NT4.0/2000/XP и все вытекающее отсюда техническое обеспечение.

Кроме того, требуется:

1. физическая память 1Mб + память на используемые и создаваемые из текстовые файлы (не менее 10 Мб);
2. оперативная память 2\*размер используемого файла. При нехватке оперативной памяти может использоваться виртуальная память. А за счет того, что все данные находятся в памяти, сильно сокращается время решения задач.

Для написания программы использовалась среда разработки Borland Delphi 7.

## 4.3. Выполнение программы

Программа представлена в виде формы, в главном окне которого производится выбор функции программы. Режим работы программы – диалоговый. Все выполняемые программой функции запускаются с помощью соответствующих кнопок.

Для запуска программы оператор из оболочки системы Windows должен произвести запуск файла **Наладчик.ехе**

Главное окно программы

Рис.П.4.1

Панель инструментов

Рис.П.4.2

 ----- открыть файл

 ----- сохранить файл

 ----- распечатать содержимое поля редактирования

 ----- при нажатии этой кнопки появляется дополнительный диалог обработки (см. далее)

----- очистить поле редактирования

 ----- при нажатии этой кнопки появляется дополнительный диалог замены (см. далее)

----- при нажатии всё содержимое поля редактирования преобразуется к верхнему регистру

------ при нажатии появляется дополнительный диалог (разбивка файла на несколько частей)

**Меню файл**

**Выход** – выход из программы;

**Сохранить** – вызов диалогового окна (Сохранить как);

**Открыть** – вызов диалогового окна (Сохранить как);

**Печать** – вызов диалогового окна (Печать).

**Меню настройки**

**Шрифт** - вызов диалогового окна(Шрифт);

**Принтер** - вызов диалогового окна(Настройка печати);

**Параметры** - вызов диалогового окна(Параметры).

**Меню справка**

Вызов диалогового окна(Версия) Рис.П.4.3.

Рис.П.4.3

Работа начинается с открытия текстового файла  либо выбором пункта меню Файл – открыть. В результате открывается диалог выбора файла Рис. П.4.4.

 Рис. П.4.4

Выбираем файл, нажимаем открыть. Следующим действием нажимаем,  появляется диалог, в котором выбираем желаемый метод обработки.

**Не обрабатывать** – пересчет производится не будет.

**ПК – МАЯК** – выбрана обработка конвертирование системы (Sinumeric – Маяк).

**ПК – ВМ** – выбрана обработка конвертирование системы (Sinumeric – ВМ).

**ВМ – МАЯК** – выбрана обработка конвертирование системы (ВМ – Маяк).

**МАЯК – ВМ** – выбрана обработка конвертирование сис Рис.П.4.5 темы (Маяк - ВМ).

Кнопка «Закрыть» закрывает диалог, В результате выбранных вами действий результат выбора отображается в левом нижнем углу главной формы рис.П.4.6.



Рис.П.4.6

Выбрано обработка ПК – МАЯК

**Дополнительная панель настройки обработки**

**Номера кадров** – упорядочить номера кадров, начиная с 1.

**Пробелы** – будут удалены все пробелы.

**Пустые строки** – будут удалены все пустые строки.

Рис.П.4.7

## 4.4 Информация во время обработки

нажимаем кнопку **<**Обработка**>**

Рис.П.4.8

Это - информационная панель, отображающая в процентах и графически, сколько уже обработано данных файла во время обработки рис.П.4.9.



Рис.П.4.9

Во время обработки в верхней части окна показано, сколько обработано строк, в нижней сколько всего рис.П.4.9.1.



Рис.П.4.9.1

**Всё файл обработан, сохраняем**   или меню файл – сохранить

В результате нажатия вышесказанной кнопки появляется следующий диалог.

Выберите желаемый файл из списка или напишите имя в поле «Имя файла» рис.П.4.9.2.

Рис.П.4.9.2

всё, можно закрывать программу кнопка **«**Закрыть**»**

**Функция разделения файла**

При нажатии кнопки на панели инструментов (Рис.П.4.9.3.) появляется диалог «Разбивка файла на несколько частей»

Рис.П.4.9.3

При нажатии на кнопку «Обработка» происходит разбивка файла на файлы заданной ёмкости. Нажимаем кнопку «Настройка» рис.П.4.9.3.

Рис.П.4.9.4

 В этом диалоге при нажатии кнопки «Редактирование вставки в конец файла» текст набранный в поле редактора (находится внизу формы) вставляется в каждую часть файла рис.П.4.9.4.

 При нажатии кнопки «Редактировать конечную папку» появляется диалог «Сохранить как» (Рис.П.4.9.2 ).

## 4.5. Дополнительные настройки

**Меню – Настройки**

 **Шрифт** – выбор шрифта для поля редактирования рис.П.4.9.5.

Рис.П.4.9.5

**Принтер** – установка свойств принтера для вывода на печать рис.П.4.9.6.

 Рис.П.4.9.6

Это диалог вызывается нажатием кнопки  и работает как замена чего-то чем-то. В данном примере все вхождения **X** заменяются **Y** рис.П.4.9.7.


Рис.П.4.9.7

## 4.6. Инструкции по обработке исключительных ситуаций

Во время работы программы могут поступать сообщения пользователю (исключительные ситуации), которые запрограммированы лишь **для повышения надёжности программы и исключения, как можно больше ошибок.**

*- Это сообщение поступает, когда вы нажимаете кнопку* ***<Обработка>****, но в поле редактирования пусто.*

**Действия** – откройте не пустой файл и повторите обработку.Рис.П.4.9.8

Сообщение появляется при попытке закрыть программу, не сохранив изменившееся поле редактирования.

**Действия** *– если хотите сохранить изменённый файл нажмите* ***<Yes>***  *иначе* ***<No>*** рис.П.4.9.9


## 4.7. Дополнительные инструкции по программированию и математическая постановка

Как отмечалось выше, эта версия программы обрабатывает исходный код, который генерирует программа **Unigraphics.**

*Пример кода* изUnigraphics

Программа 1

N0800 X90.91 Y-160.881

 N0810 G3 X91.643 Y-166.402 I423.693 J53.462

 N0820 X92.657 Y-173.301 I350.64 J47.998

 N0830 G1 X91.625 Y-178.443

 N0840 G2 X91.623 Y-178.442 I2.026 J6.7

 N0850 X90.181 Y-177.736 I1.441 J4.76

 N0860 G1 X89.494 Y-177.174

 N0870 X89.242 Y-176.969

Как видно из примера, координаты задаются в миллиметрах до 0.001 долей (при обработке в программе **Наладчик** размеры разрядностью более 0.001 отбрасываются).

*Пример*  **Unigraphics** 0.0016 после обработки **Наладчик** 0.001

Центр радиуса задаётся относительно начальной точки радиуса (относительно).

 N0810 G3 X91.643 Y-166.402 I423.693 J53.462

**ПК – МАЯК**

У Маяков, как известно координаты задаются тоже в миллиметрах, но **I, J –** координаты центра радиусов задаются относительно начала системы координат (абсолютно) таб.П.4.10.

Таблица П4.10

Программа до и после обработки.

|  |  |
| --- | --- |
| Программа до обработки(программа 1) | Программа после обработки ПК – МАЯК(программа 2) |
| N0800 X90.91 Y-160.881 N0810 G3 X91.643 Y-166.402 I423.693 J53.462 N0820 X92.657 Y-173.301 I350.64 J47.998 N0830 G1 X91.625 Y-178.443 N0840 G2 X91.623 Y-178.442 I2.026 J6.7 N0850 X90.181 Y-177.736 I1.441 J4.76 N0860 G1 X89.494 Y-177.174 N0870 X89.242 Y-176.969 | N73X90.91Y-160.881N74G3X91.643Y-166.402I514.603J-107.419N75X92.657Y-173.301I442.283J-118.404N76G1X91.625Y-178.443N77G2X91.623Y-178.442I93.651J-171.743N78X90.181Y-177.736I93.064J-173.682N79G1X89.494Y-177.174N80X89.242Y-176.969 |

Принцип пересчёта:

Пересчёт радиусов

**Iнаст = Xпред + Iнаст  Это формулы пересчёта радиусов с относительной в**

**Jнаст = Yпред + Jнаст абсолютную систему координат.**

 То есть, для кадра N74 (программа 2) берём из кадра N0800 (программа 1) **X, Y** из N0810 берём **I, J** получаем:

**Iнаст =** **X90.91 + I423.693 = 514.603** результат обработки в программе 2 (кадр N74)

**Jнаст = Y-160.881 + J53.462 = -107.419**

**Расчёт линейных заходов (функции G41, G42) на эквидистанту контура обрабатываемой детали.**

Таблица П4.11

Добавление недостающих координат

|  |  |
| --- | --- |
| до обработки (программа 1) | после обработки (программа 2) |
| N5G1Z-25.F500M8N6G90G42D20X-104.6Y0F20N7Y-23. | N5G1Z-25.F500M8N6G90G42D20X-104.6Y0I0J-23.F20N7Y-23. |

Здесь мы видим, что в программу были вставлены недостающие части кадра **I** и **J** координаты таб.П4.11.

**I = X\_sled – X\_nast; Это формулы расчёта координат заходов на линейный**

**J = Y\_sled – Y\_nast контур детали.**

 То, есть для кадра N6 (программа 2) берём из кадра N6 и N7 (программа 1) настоящие и следующие значения координат **X** и **Y**.

**I =** (его нет значит значение то же, что и предыдущее) 104.6 – 104.6 = 0

**J =** 23 – 0 = 23

**Расчёт заходов на радиус (функции G41, G42) на эквидистанту контура обрабатываемой детали.**

С начала рассчитываем значение (I и J) из следующего кадра.

**Логическое решение задачи**

Если G3, то проверяем значение заданных координат:

**Если I\_sled > 0 и J\_sled >= 0, то I = 1, J = -1;**

**Если I\_sled < 0 и J\_sled <= 0, то I = -1, J = 1;**

**Если I\_sled < 0 и J\_sled >= 0, то I = 1, J = 1;**

**Если I\_sled > 0 и J\_sled <= 0, то I = -1, J = -1**

Если G2, то проверяем значение заданных координат:

**Если I\_sled > 0 и J\_sled >= 0, то I = -1, J = 1;**

**Если I\_sled < 0 и J\_sled <= 0, то I = 1, J = -1;**

**Если I\_sled < 0 и J\_sled >= 0, то I = -1, J = -1;**

**Если I\_sled > 0 и J\_sled <= 0, то I = 1, J = 1**

 Следующее операция описана формулами:

**I\_nast = Y\_nast + J\_sled**

**J\_nast = X\_nast + I\_sled**

 Заключительные вычисления:

**если I = -1 и J > 0, то J\_nast:= -J\_nast;**

**если I = 1 и J < 0, то J\_nast:= -J\_nast;**

**если J = -1 и I > 0, то I\_nast:= -I\_nast;**

**если J = 1 и I < 0, то I\_nast:= -I\_nast.**

В данном примере срабатывает условие, **если I\_sled > 0 и J\_sled >= 0**,

значит **I** = 1, **J** = -1, то есть **I** – будет положительным, а **J** - отрицательным

 Далее по формулам: **I\_nast = Y\_nast + J\_sled; J\_nast = X\_nast + I\_sled**

находим: **I\_nast** = 0 + 0 = 0; **J\_nast** = 104.6 + (-2.5) = 102.1

 Залючительные вычисления, **если I = 1 и J < 0, то J\_nast:= -J\_nast;**

**J\_nast** = -102.1, что мы и видим в таб. П4.12.

Таблица П4.12

Расчёт заходов на радиус (функции G41, G42)

|  |  |
| --- | --- |
| до обработки (программа 1) | после обработки (программа 2) |
| N5G1Z-25.F500M8N6G90G42D20X-104.6Y0I0J-23.F20N8G3X-102.1Y-25.5I2.5J0.0 | N1G1Z-25.F500M8N2G90G42D20X-104.6Y0I0J-102.1F20N3G3X-102.1Y-25.5I-102.1J0 |

**Не пересчитывайте программу дважды**. Обычно бывает так **–** программист сгенерировал программу в (**Unigraphics)** или где-то в другом редакторе на МАЯК или ВМ в **«Наладчике»** и забыл ещё вставить код, он его добавляет и опять открывает и пересчитывает. Вообще-то программа не даст пересчитать дважды, но если открыть и закрыть файл то можно просчитать ещё **раз (сколько угодно раз можно пересчитывать только номера кадров, пустые строки и пробелы, без урона для программы).** Здесь же чисто математический расчёт и программа, пересчитанная дважды, **не будет правильной.**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc201448092)

[1. РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО КОН ВЕРТИРОВАНИЯ ИСХОДНОГО ТЕКСТА ПРОГРАММ ДЛЯ СТАНКОВ С ЧПУ 6](#_Toc201448093)

[1.1. Обоснование целесообразности разработки системы автоматического конвертирования текста исходных программ для станков с ЧПУ 6](#_Toc201448094)

[1.1.1. Назначение системы 6](#_Toc201448095)

[1.1.2. Характеристика функциональной структуры системы 6](#_Toc201448096)

[1.1.3. Обоснование цели создания системы 7](#_Toc201448097)

[1.1.4. Обоснование состава автоматизируемых задач 7](#_Toc201448098)

[1.2. Аналитический обзор 8](#_Toc201448099)

[1.3. Основные требования к системе 10](#_Toc201448100)

[1.3.1. Основные цели создания системы и критерии эффективности её функционирования 10](#_Toc201448101)

[1.3.2. Функциональное назначение системы 10](#_Toc201448102)

[1.3.3. Особенности программы и условия её эксплуатации 11](#_Toc201448103)

[1.3.4. Требования к функциональной структуре 11](#_Toc201448104)

[1.3.5. Требования к техническому обеспечению 12](#_Toc201448105)

[1.3.6. Требования к информационному обеспечению 12](#_Toc201448106)

[1.3.7 Требования к программному обеспечению 12](#_Toc201448107)

[1.4. Основные технические решения проекта системы 13](#_Toc201448108)

[1.4.1. Решение по комплексу технических средств 13](#_Toc201448109)

[1.4.2. Описание системы программного обеспечения 13](#_Toc201448110)

[2. РАЗРАБОТКА ЗАДАЧИ «Конвертирование исходного текста программ для станков с ЧПУ» 14](#_Toc201448111)

[2.1. Описание постановки задачи 14](#_Toc201448112)

[2.1.1. Характеристика задачи 14](#_Toc201448113)

[2.1.2. Входная информация 14](#_Toc201448114)

[2.1.3. Выходная информация 14](#_Toc201448115)

[2.1.4. Математическая простановка задачи 15](#_Toc201448116)

[2.2. Описание алгоритма метода выбора обработки 20](#_Toc201448117)

[2.2.1. Назначение и характеристика алгоритма 20](#_Toc201448118)

[2.2.2. Используемая информация 20](#_Toc201448119)

[2.2.3. Результаты решения 20](#_Toc201448120)

[2.2.4. Алгоритм решения 20](#_Toc201448121)

[2.2.5. Список условных обозначений 21](#_Toc201448122)

[2.3 Описание алгоритма выбора вспомогательных методов обработки 23](#_Toc201448123)

[2.3.1. Назначение и характеристика алгоритма 23](#_Toc201448124)

[2.3.2. Используемая информация 23](#_Toc201448125)

[2.3.3. Результаты решения 23](#_Toc201448126)

[2.3.4. Алгоритм решения 23](#_Toc201448127)

[2.4. Описание алгоритма конвертирования 24](#_Toc201448128)

[2.4.1. Назначение и характеристика алгоритма 24](#_Toc201448129)

[2.4.2. Алгоритм решения 24](#_Toc201448130)

[2.5 Описание программы «Наладчик» 27](#_Toc201448131)

[2.5.1. Водная часть 27](#_Toc201448132)

[2.5.2. Функциональное назначение 27](#_Toc201448133)

[2.5.3. Описание информации 28](#_Toc201448134)

[2.5.4. Используемые подпрограммы 28](#_Toc201448135)

[2.5.5. Описание логики 29](#_Toc201448136)

[2.6. Описание контрольного примера 30](#_Toc201448137)

[2.6.1. Назначение 30](#_Toc201448138)

[2.6.2. Исходные данные 30](#_Toc201448139)

[2.6.3. Результаты расчета 30](#_Toc201448140)

[2.6.4. Результаты испытания 30](#_Toc201448141)

[3. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ 31](#_Toc201448142)

[3.1. Обоснование необходимости разработки системы конвертирования текста программ для станков с ЧПУ 31](#_Toc201448143)

[3.2. Планирование разработки 31](#_Toc201448144)

[3.3. Расчет расходов на создание системы 32](#_Toc201448145)

[4. БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ ПРОЕКТА 37](#_Toc201448146)

[4.11 Анализ вредных и опасных производственных факторов 37](#_Toc201448147)

[4.2. Техника безопасности при работе с компьютером 39](#_Toc201448148)

[4.3. Электробезопасность при работе с компьютером 41](#_Toc201448149)

[4.4. Требования к освещению рабочего места и его расчет 42](#_Toc201448150)

[4.4.1. Требования к освещению рабочего места 42](#_Toc201448151)

[4.4.2. Расчет освещенности рабочего места 44](#_Toc201448152)

[4.5. Параметры микроклимата, загрязнение воздуха рабочей зоны вредными веществами 47](#_Toc201448153)

[4.6. Пожаробезопасность при работе с компьютером 48](#_Toc201448154)

[ЗАКЛЮЧЕНИЕ 52](#_Toc201448155)

[СПИСОК ЛИТЕРАТУРЫ 52](#_Toc201448156)

[ПРИЛОЖЕНИЕ 1 ТЕКСТ ПРОГРАММЫ 54](#_Toc201448157)

[ПРИЛОЖЕНИЕ 2 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 84](#_Toc201448159)

[ПРИЛОЖЕНИЕ 3 РУКОВОДСТВО ПРОГРАММИСТА 86](#_Toc201448161)

[ПРИЛОЖЕНИЕ 4 РУКОВОДСТВО ОПЕРАТОРА 88](#_Toc201448169)