МIНIСТЕРСТВО ОСВIТИ УКРАIНИ

ДОНБАСЬКА ДЕРЖАВНА МАШИНОБУДIВНА АКАДЕМIЯ

Кафедра комп’ютерних інформаційних технологій

Комплексний проект:

Розробка програмно-методичного комплексу для автоматизації конструкторської підготовки виробництва.

Тема: Дослідження методів автоматизованного проєктування динамічних систем

Спецчастина: Розробка програмно-методичного комплексу та типових параметричних моделей

2003 р.

**СОДЕРЖАНИЕ**

Введение

1. Структурно-информационный анализ методов моделирования динамических систем

1.1 Анализ причин возникновения динамических явлений в технических системах

1.2 Анализ подходов к моделированию динамических систем

Математическое моделирование динамических систем

Численные методы решения систем дифференциальных уравнений

1.3 Анализ программных продуктов для моделирования динамики систем

1.4 Словарь предметной области

2. Разработка структуры программного комплекса для анализа динамики механических систем

2.1 Разработка математической модели программного комплекса для моделирования динамических систем

2.2 Разработка логической структуры ПМК

2.3 Разработка модульной структуры ПМК

2.4 Виды обеспечения функционирования ПМК

2.5 Разработка рабочего проекта ПМК

3. Разработка программно – методического комплекса для моделирования динамики механических систем.

3.1 Структура и функциональное назначение отдельных модулей

3.2 Информационное обеспечение

4. Исследование колебательной системы (эксперимент)

4.1 Задачи исследования колебательной системы

4.2 Построение модели плана и порядка

4.3 Кодирование факторов

4.4 Составление план – матрицы

4.5 Полученная регрессионная модель

4.6 Графическое представление полученной модели

5. Охрана труда

5.1 Анализ опасных и вредных производственных факторов

6.2 Разработка мероприятий по обеспечению безопасных и комфортных условий труда

6.3 Расчет общего естественного освещения.

Выводы

Список использованной литературы

Приложение а

Приложение б

Приложение в

**РЕФЕРАТ**

Пояснительная записка к дипломной работе студента группы ИТ 89 – 1 Бурлея Павла Александровича содержит страниц машинописного текста, 29 рисунков, 6 таблиц, три приложения.

Основная цель работы анализ методов автоматизированного проектирования динамических систем и разработка программно – методического комплекса для анализа динамики механических систем.

В работе рассмотрены и проанализированы методы моделирования вообще, нюансы моделирования динамических систем. Рассмотрены существующие программные продукты для моделирования динамики систем. Был выполнен анализ этих систем.

Была разработана структура программного комплекса, по которой потом был разработан комплекс для анализа динамических систем. Программно методический комплекс предназначен для моделирования работы колебательной системы с одной степенью свободы.

По охране труда проведен анализ опасных и вредных производственных факторов, разработаны мероприятия по обеспечению безопасных и комфортных условий труда, выполнен расчет естественного освещения.

**ВВЕДЕНИЕ**

Актуальность темы. Развитие вычислительной техники увеличение скорости выполняемых операций, ёмкости памяти [1-3] позволяет переложить на плечи компьютера всё больше и больше задач. Кроме того, те задачи, которые даже не пытались решать раньше, в виду сложности вычислений, на современном этапе развития с успехом выполняются.

Одной из таких задач является расчёт динамики систем. В этой области хоть и существует мощный математический аппарат, но переложить его на ЭВМ оказалось не так уж и легко [4-10].

Среди существующих на данный момент програмных продуктов не встретишь продуктов для машиностроения, которые выполняли бы анализ динамики систем. А ведь информация о динамических процессах в эщё только на этапе проэктирования поможет сэкономить и средства и время. Поэтому разработка програмного обеспечения, которое выполняло бы расчёт параметров динамических систем актуально на сегодняшний день.

Как известно, в 2002 – 2003 годах рынок информационных технологий пережил спад [11]. Это вызвано, на мой взгляд, тем, что рынок информационных технологий насыщен программными продуктами решающими стандартные задачи. Выигрыш получает тот, кто предложит, что - то новое, что ещё не кто не решал [12-15]. Возникает вопрос, нужно ли кому нибуть это новшество. Поэтому важно, что бы новшество было востребовано. Кроме того, все поняли, что компьютер может многое, но не всё, то есть, компьютер должен не заменять, а помогать человеку в его деятельности.

Что поможет решить анализ динамики систем, любое проектируемое оборудование будет работать в условии динамических нагрузок. В природе ничто не стоит на месте. А вот будут ли влиять эти нагрузки на работу оборудования и как, предположить очень трудно. Для того, что бы выявить влияние этих нагрузок, уходит очень много времени, а если бы была возможность выполнить анализ динамических явлений в системе с помощью компьюьера, это бы значительно ускорило и удешевило процесс создания машиностроительного оборудования. Кроме того, если точно не просчитано как поведёт себя оборудование в той или иной ситуации (поломка инструмента, заготовок, элементов конструкции) возможны человеческие жертвы в процессе эксплуатации.

Таким образом, анализ динамики систем поможет не только сэкономить деньги и время, но и сохранить человеческие жизни.

**1 СТРУКТУРНО-ИНФОРМАЦИОННЫЙ АНАЛИЗ МЕТОДОВ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ**

## 1.1 Анализ причин возникновения динамических явлений в технических системах

Необходимость анализа динамических процессов оборудования обусловлена возрастающими требованиями, предъявляемыми к техническому уровню продукции машиностроения. Такой анализ выполняется как на стадии проектирования, так и в процессе эксплуатации для определения оптимальных режимов работы [15,26].

Проблема изучения колебаний актуальна для многих областей техники. В одних случаях колебательные явления опасны и способны принести значительный вред, в других – целенаправленно применяются в технике. Для учёта влияния колебаний на систему, необходимо провести анализ динамических процессов [27,28]. Особенно эффективен анализ динамических процессов на стадии выбора принципиальных схем оборудования для определения его работоспособности в условиях динамических режимов, т.к. динамический анализ работы сконструированного оборудования связан со значительными затратами на изменение конструкции в случае получения неудовлетворительных результатов [16,23,24].

В процессе проектирования новых видов оборудования возникают проблемы, связанные с моделированием динамических режимов его работы. Обычно расчёт производится для статического режима по правилам сопротивления материалов, если же влияет динамическая нагрузка, то учёт влияния этих нагрузок производится с помощью увеличения запаса прочности Кузнечно-штамповочное оборудование (КШО) работает в условиях динамических нагрузок, которые могут значительно превышать расчетные (технологические) и служить причиной поломок [17,18]. Возникновение динамических нагрузок на базовые детали оборудования и фундамент может быть вызвано следующими основными причинами (рисунок 1.1).

Рисунок 1.1 - Причины возникновения динамических процессов в КШО.

Причины возникновения динамических процессов в технологическом оборудовании

## конструктивные

технологические

связанные с работой привода

связанные с работой исполнительного механизма

регулярные

однократные

(аварийные)

Недостаточная жёсткость отдельных элементов и конструкции в целом приводит к тому, что в процессе выполнения технологической операции в приводе и силовой раме накапливается значительная потенциальная энергия. При разгрузке машины на каждом рабочем ходе эта энергия высвобождается и может приводить к возникновению динамических нагрузок в элементах конструкции [19,25]. Резкое изменение (исчезновение либо увеличение) технологической нагрузки также вызывает динамические явления в приводе и элементах силовой рамы [28,29]. Например, при выполнении разделительных операций (отрезка, вырубка, пробивка) прессы работают в условиях динамических нагрузок, которые могут значительно превышать расчетные и служить причиной снижения точности выполнения операций и поломок оборудования. Однократные динамические перегрузки могут быть связаны с разрушением элементов конструкции машины и оснастки, т.к. в момент разрушения накопленная потенциальная энергия преобразуется в кинетическую и вызывает колебания [20,21,22].

Таким образом, для решения данной проблемы необходим метод, который позволил бы, не создавая работающий экземпляр, предсказать его поведение в той или иной ситуации, то есть промоделировать его.

**1.2 Анализ подходов к моделированию динамических систем**

Модели реальных объектов, моделирование явлений издавна используются в науке и технике для проверки идей, отработки гипотез, получения экспериментального материала. За примером можно обратиться к авиастроению. Важным элементом проектной и конструкторской работы над новой машиной является выбор ее форм, оптимизация аэродинамических характеристик. Законченная теория здесь пока еще отсутствует, и этот недостаток приходится восполнять экспериментальными исследованиями. Вот тут и приходят на помощь модели. И реальный самолет, и его уменьшенная копия в воздушном потоке подчиняются одним и тем же законам аэродинамики. При определенных условиях, измерив, аэродинамические нагрузки на модели, оказывается возможным пересчитать их на “настоящий” летательный аппарат. Модель и самолет оказываются похожими не только внешне, но и по “физическому содержанию” – их аэродинамические характеристики связаны определенными соотношениями подобия. Итак, модель – это не только и не столько внешнее сходство. Главное лежит глубже – поведение модели реального объекта должно подчиняться одинаковым закономерностям. Изучив их на доступной для исследования модели, оказывается возможным предсказать свойства проектируемой конструкции.

Одной из проблем современной науки является разработка и внедрение в практику методов исследования функционирования сложных систем. К классу сложных систем относят технологические, производственные, энергетические комплексы, системы автоматизации управления и другие объекты. Моделирование является одним из наиболее мощных средств исследования подобных систем на сегодняшний день [30].

Моделирование - один из наиболее распространенных способов изучения различных процессов и явлений. Моделью исходного объекта называется представление объекта в некоторой форме, отличной от формы его реального существования. В инженерной практике модель обычно создается для следующих целей.

1. проведения на модели экспериментов, которые невозможно или сложно провести на реальном объекте (что предоставляет возможность получения новых знаний об объекте);
2. ускорения, удешевления, упрощения и любого другого усовершенствования процесса проектирования, достигаемого за счет работы с более простым объектом, чем исходный, то есть с моделью.

В настоящее время известны и широко используются в научных исследованиях и инженерной практике различные типы моделей и многочисленные методы моделирования. Если взять за основу степень абстрактности (степень отличия от реального объекта), то можно определить следующие типы моделей.

1. физические (натурные) модели (воспроизводят изучаемый процесс с сохранением его физической природы и являются инструментом физического моделирования);
2. аналоговые модели (заменяют один объект на другой с похожими свойствами);
3. математические модели (абстрактные модели, существуют в форме специальных математических конструкций и имеют смысл только для интерпретирующего их человека или машины).

Методы исследования динамических режимов работы оборудования можно разделить на следующие две категории: физическое и математическое моделирование.

Изучение динамики работы оборудования при физическом моделировании выполняется на модели или реальной машине путем экспериментального измерения физических параметров элементов машины во времени (скоростей, перемещений, напряжений, деформаций и др.). После их обработки строятся корреляционные зависимости связи параметров, определяется вид аппроксимирующих кривых, вырабатываются рекомендации по устранению тех или иных недостатков. Достоинством данного подхода является достаточно высокая достоверность полученных данных. Недостатки заключаются в необходимости физической модели, узкой специализации исследований, ограниченной возможности изменения конструктивных параметров объекта, достаточно высокой трудоемкости проведения экспериментов. Значительные трудности возникают при изучении критических режимов работы оборудования, вследствие возможности его поломки. В целом, метод физического моделирования служит для первоначального получения знаний и проверки теоретических исследований.

## Математическое моделирование динамических систем

При математическом моделировании исследование динамики работы оборудования осуществляется на основе изучения поведения его математической модели. Построение такой модели базируется на расчетной схеме исследуемого объекта, а также принятых ограничениях и допущениях. Большинство математических моделей динамических систем представляют собой систему дифференциальных уравнений, отражающих физические процессы в объекте. Рассмотрим математическую модель гидравлического пресса на макроуровне, как гидромеханической системы [31]. В качестве её элементов выбираются функциональные узлы пресса, для которых модель описывается на основании уравнений движения масс (структурные компоненты), уравнений связи масс в единую систему (топологические компоненты) и принципа Даламбера. В общем случае такая модель имеет вид:

( 1.1)

где ‑масса iго элемента;

, , ‑перемещение, скорость и ускорение iго элемента;

 - жёсткость kго элемента; t - время;

 - обобщенная сила, действующая на iый элемент.

На рисунке 1.2 приведена типичная расчётная схема гидравлического пресса. Расчётная схема представлена колебательной системой с одной степенью свободы.

Cфундамента

X1

X2

X3

Cколоп

Lk

M1

M2

M3

Cпоковки

Рисунок 1.2 – Пример расчётной схемы гидравлического пресса

Колебательная система состоит из трёх масс – М1, М2, М3; и из трёх элементов на расчётной схеме представленными пружинами с соответствующими жестокостями: Cфундамента, Cпоковки, Cколоп. Система дифференциальных уравнений описывающих данную модель будет выглядеть следующим образом:

(1.2)

В зависимости от сложности описания динамической системы можно условно выделить два направления изучения происходящих в ней процессов: аналитическое и численное [31,32]. При решении конкретных задач, в случае невозможности получения точного аналитического решения при учёте всех существенных факторов, принимается целый ряд допущений, отражающихся на адекватности модели. Например, при моделировании работы гидравлических прессов, в большинстве случаев, силы, действующие на элементы конструкции, полагаются постоянными во времени, не учитываются изменения жёсткости, зазоры, волновые процессы в гидросистеме (т.к. это повлекло бы за собой необходимость совместного решения системы дифференциальных обычных и волновых уравнений в частных производных) В таком случае аналитическое решение системы (1.3) можно представить в виде:

(1.3)



где параметры  и  определяются начальными условиями вида:

.

Следует отметить, что хотя такой метод и дает достаточно точное аналитическое решение, но на практике его применение ограничено моделированием достаточно простых объектов, содержащих максимум три‑четыре элемента [33,34].

## Численные методы решения систем дифференциальных уравнений

Рассмотрим методы решения обыкновенных дифференциальных уравнений (далее ОДУ) первого порядка с одним начальным условием. Это так называемая задача Коши: найти частное решение  ОДУ ,(1.4) удовлетворяющее начальному условию [35]:

. (1.5)

Эти методы легко обобщаются на системы уравнений первого порядка. Кроме того, уравнения более высоких порядков можно свести к системе ОДУ первого порядка. Например, изменяя в уравнении  y на y1, а y’ на y2, получим систему:

 (1.6)

Наиболее распространенными являются методы конечных разностей (МКР) [36]. В разностных методах вводится последовательность точек  и шаг В каждой точке xi, называемой узлом вместо значений функции y(xi) вводят числа yi, которые аппроксимируют точное решение y на данном множестве точек. Функцию y, заданную в виде таблицы I=0,1,2,…, называют сеточной функцией.

Изменяя значение производной в уравнении (1.5) отношением конечных разностей, осуществляют переход от дифференциальной задачи (1.5), (1.6) к разностной как то сеточной функции y:

. (1.7)

Конкретное выражение правой части (1.7) зависит от способа аппроксимации производной, и функция F определяет вычислительную схему метода.

Можно привести такую классификацию методов. Если r=0 и , то численный метод называется одношаговым, если или s>1 , то многошаговым.

Многошаговые и одношаговые методы называются явными, если s=0, и неявными, если . [36].

## 1.3 Анализ программных продуктов для моделирования динамики систем

С развитием вычислительной техники всё большее распространение получают численные методы математического моделирования. Эффект от такого подхода проявляется в сокращении сроков и удешевлении проектирования машин, повышении их надежности. Математическое моделирование на ЭВМ динамики машин является эффективным методом исследования, позволяющим оперативно оценить влияние на рабочий процесс любых факторов, в том числе и конструктивных, которые достаточно сложно промоделировать иным способом. Основная трудность при этом заключается в создании качественного программного обеспечения (ПО), включающего общие вычислительные алгоритмы, ориентированные на целый класс прикладных задач. Ранее, по причине ограниченных вычислительных возможностей ЭВМ и преобладания процедурного программирования, ПО было узкоориентировано на моделирование конкретной машины, что требовало значительных затрат на разработку [5].

С переходом на проблемно‑ориентированный подход ПО разрабатывается для целого класса машин на основе определённой концепции. Наибольшее распространение получили методы построения математической модели на основе обобщенной модели и эквивалентных схем

Первый метод заключается в создании обобщённой математической модели (ОММ), которая является формальной суммой всех возможных для данного класса устройств элементов и взаимосвязей между ними. Частная модель получается из ОММ простым удалением «лишних» для данного объекта элементов и связей. Естественно, что при достаточной простоте получения частной модели, задача создания законченной ОММ и её исследования представляет весьма значительную трудность.

В основе метода эквивалентных схем лежит идентичность математического описания процессов, происходящих в физически разнородных системах. Так уравнение механического движения массы  математически эквивалентно уравнению электрической ёмкости и уравнению сжимаемости жидкости . Таким образом, любая механическая или гидравлическая система может быть сведена к электрической и достаточно просто промоделирована как с помощью уже существующих средств моделирования электросхем, так и разработанных специально для данного реального класса объектов. Очевидное преимущество этого метода - возможность моделирования физически разнородных систем, как единой сложной однородной системы. Основной недостаток - ограничения, накладываемые на включение в систему нелинейных компонент, в результате чего эквивалентная электрическая схема моделируемого механического или гидравлического объекта многократно усложняется.

В таблице 1.1 приведены аналогии между фазовыми переменными для различных физических подсистем.

Таблица 1.1 - аналогии между фазовыми переменными для различных физических подсистем

|  |  |  |  |
| --- | --- | --- | --- |
| Подсистема | Поток | Потенциал | Базовый узел |
| Электрическая | Ток | Электрический потенциал | Шина “земля” |
| Механическая поступательная | Сила | Скорость | Неподвижная система отсчета |
| Механическая вращательная | Момент силы | Угловая скорость | Неподвижная система отсчета |
| Гидравлическая (пневматическая) | Расход | Давление | Атмосфера или абсолютный вакуум |
| Тепловая | Тепловой поток | Температура | Окружающая среда или абсолютный нуль |

В таблице 1.2 приведены аналогии базовых двухполюсников для различных физических подсистем.

Таблица 1.2 - аналогии базовых двухполюсников для различных физических подсистем

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Подсистема | Двухполюсники | | | |
|  | Типа C | Типа L | Типа G | Типа R |
| Электрическая | Емкость | Индуктивность |  | Сопротивление |
| Механическая поступательная | Масса | Упругость | Вязкое трение |  |
| Механическая вращательная | Момент инерции | Вращательная гибкость | Вязкое вращательное трение |  |
| Гидравлическая (пневматическая) | Гидравлическая емкость | Гидравлическая индуктивность |  | Гидравлическое сопротивление |
| Тепловая | Теплоемкость |  | Теплопроводность |  |

Применяя вышеописанные методы, были разработаны математические модели и программное обеспечение, позволяющее моделировать отдельные компоненты сложных динамических систем: гидравлического пресса как механического объекта, его гидросистемы, техпроцессов обработки металлов давлением. В настоящее время требуемая эффективность решения задачи моделирования технических объектов может быть обеспечена только системным подходом. Современный уровень развития компьютерной техники и технологии позволяет комплексно решить эту проблему на основе объектно‑ориентированного подхода (ООП), с применением которого возможна разработка объектно‑ориентированной библиотеки иерархии классов элементов систем практически любой степени сложности. При этом не возникает затруднений, связанных с взаимодействием объектов, программное обеспечение остается полностью открытым, возможна его модификация и дальнейшее развитие на базе уже созданного программного обеспечения.

Среди программных средств, могучей волной обрушившихся на пользователей персональных компьютеров немалую долю составляют инструментальные средства, предназначенные для бизнес-планирования и бизнес-моделирования. Они могут быть представлены как самостоятельные продукты либо как дополнения к программному обеспечению для презентаций или к другим программным средствам. Спектр подобных средств столь широк, что попытка выбрать для решения конкретной задачи оптимальный вариант очень сложна. Конечно, опытный человек способен сделать правильный выбор и скорее всего, выберет достаточно простой и уже известный либо ему, либо его друзьям инструмент. Менее опытный - скорее всего, предпочтёт новое ПО, в названии которого фигурируют непонятные, но такие многообещающие слова, как, например, "генетический", "интеллектуальный", "нейросетевой" и т. д. Попробуем непредвзято взглянуть на существующие инструментальные средства моделирования и, насколько это, возможно, выработать некоторые критерии, позволяющие облегчить выбор нужного программного продукта.

В настоящее время компьютерная промышленность предлагает современному инженеру целый ряд разнообразных средств моделирования, позволяющих не только моделировать сложные динамические системы, но и проводить с ними эксперименты. Некоторые элементы анализа динамических процессов так же включаются в состав CAD систем. Например, SolidWorks. В данной системе имеется модуль Dynamic. Этот модуль позволяет визуализировать процесс работы механизма, а любой элемент механизма можно выбрать в интерактивном режиме и выполнить расчёт в CAE системе Cosmos. Наиболее полное исследование общесистемных проблем получается в результате моделирования объектов с помощью современных технологий, реализованных в специализированных вычислительных пакетах или пакетах визуального моделирования. На сегодня существует огромное число пакетов визуального моделирования. В них пользователю предоставляется возможность описывать моделируемую систему преимущественно в визуальной форме, например, графически представляя как структуру системы, так и ее поведение (например, при помощи карты состояний). Такой подход позволяет пользователю не заботится о реальной программной реализации модели, что значительно упрощает процесс моделирования. Результаты эксперимента в пакетах визуального моделирования предоставляются в более наглядной для человека форме: в виде графиков, гистограмм, схем, с применением анимации и т.д. Также в той или иной мере поддерживается технология объектно-ориентированного моделирования, что позволяет повторно использовать экземпляры моделей с возможностью внесения в них тех или иных корректив. Из множества существующих на сегодняшний день пакетов визуального моделирования особый интерес вызывают универсальные пакеты, не ориентированные на определенную узкоспециальную область (физика, химия, электроника и т.д.) или определенные типы моделей (чисто дискретные или чисто непрерывные), а позволяющие моделировать принадлежащие различным прикладным областям структурно - сложные гибридные системы. Несмотря на то, что современные универсальные пакеты визуального моделирования обладают рядом общих свойств (позволяют строить из блоков функциональные иерархические схемы, предоставляют пользователю схожие библиотеки численных методов, средства визуализации поведения и наборы анимационных возможностей, поддерживают технологию объектно-ориентированного моделирования). Одним из таких пакетов является пакет AnyLogic. AnyLogic – новая среда для виртуального прототипирования сложных систем с дискретным, непрерывным или гибридным поведением, разработанная компанией Experimental Object Technologies на базе систем COVERS и ModelVision. AnyLogic позволяет быстро создавать исполняемую модель – виртуальный прототип – разрабатываемой системы и ее окружения, в том числе физические объекты и поведение пользователей.

Структурные диаграммы.

Структурные диаграммы AnyLogic, основанные на стандарте UML-RT, – удобная концепция для описания иерархических, объектно-ориентированных систем. Базовым элементом модели AnyLogic является активный объект, который может включать в себя другие объекты. Активные объекты взаимодействуют с помощью посылки сообщений через порты или же через связи между фазовыми переменными объектами (расширение UML-RT).

Гибридные диаграммы состояний.

Гибридные диаграммы состояний – наиболее удобный и мощный способ интеграции дискретной логики и непрерывного во времени поведения. Гибридные диаграммы состояний, основанные на UML, позволяют приписать систему алгебро - дифференциальных уравнений любому состоянию диаграммы. При переходе системы из одного состояния в другое меняется и система уравнений – таким образом дискретная логика может влиять на непрерывное поведение модели. Если же условием срабатывания перехода между состояниями является предикат, включающий непрерывно меняющуюся переменную, то получается противоположный эффект: непрерывное во времени поведение создает событие для дискретной части модели.

Редактор моделей.

Редактор моделей AnyLogic, выполненный в виде графического интерфейса пользователя (GUI). Среди его функций настраиваемые окна, панели инструментов (toolbars), цвета и пиктограммы объектов, поддержка технологии drag and drop, навигация с помощью дерева проекта, масштабирование диаграмм, окно со свойствами объектов, подсветка синтаксиса, помощники (wizards). Так же следует отметить комплекс программ ПА9.

ПА9 - комплекс программ, предназначенный для анализа динамики электрических, механических, гидравлических, пневматических, тепловых и разнородных технических систем, основанный на методе физических аналогий. Моделируемый объект задается графическим изображением эквивалентной схемы, которая представляет собой совокупность связанных между собой по определенным правилам элементов, являющихся математическими моделями компонентов анализируемой технической системы. По графическому изображению эквивалентной схемы ПА9 автоматически формирует математическую модель в виде системы дифференциально-алгебраических уравнений (ДАУ), описывающей динамические процессы в исходной технической системе. Для интегрирования системы ДАУ в ПА9 применяются неявные А-устойчивые методы интегрирования: метод Эйлера (1-го порядка точности) и метод трапеций (2-го порядка точности). Графический редактор обеспечивает весь необходимый набор функций для формирования эквивалентной схемы моделируемого технического объекта. Результаты моделирования отображаются в виде графиков зависимостей фазовых переменных моделируемого объекта от времени.

Принципы моделирования динамики технических систем в комплексе ПА9.

Для математического моделирования технических систем различной физической природы (электрических, гидравлических, пневматических, механических, тепловых и др.) в ПА9 используется метод физических аналогий [1]. Согласно этому методу любой технической системе, функционирование которой описывается системой ДАУ, можно поставить в соответствие некоторую формальную эквивалентную схему, которая описывается точно такой же системой ДАУ.

Правила формирования эквивалентной схемы

Состояние эквивалентной схемы в любой момент времени характеризуется формальными безразмерными переменными двух типов: переменные типа “потока” и переменные типа “потенциала”.

а) “Потоки” через внешние узлы элементов эквивалентной схемы: *I*n(t), где n - порядковый номер внешнего вывода элемента, t - модельное время. Этому типу переменных соответствует первое фундаментальное уравнение связей элементов эквивалентной схемы: уравнение равновесия “потоков” в узлах – алгебраическая сумма “потоков” втекающих в любой узел схемы равна нулю.

б) “Потенциалы” узлов эквивалентной схемы относительно заранее выбранного базового узла (системы отсчета): *F*i(t), или “разности потенциалов”: *U*ij(t) = *F*i(t) - *F*j(t) между двумя узлами эквивалентной схемы с номерами i и j. Этому типу переменных соответствует второе фундаментальное уравнение связей элементов эквивалентной схемы: уравнение совместимости “разностей потенциалов” для любого замкнутого контура – алгебраическая сумма “разностей потенциалов” по любому замкнутому контуру схемы равна нулю.

## 1.4 Словарь предметной области

Динамическая система – система, в которой протекают динамические процессы, и эти процессы серьёзно влияют на работу системы.

Математическое моделирование – под математическим моделированием понимают способ исследования различных процессов путем изучения явлений, имеющих различное физическое содержание, но описываемых одинаковыми математическими соотношениями.

Математическая модель – уравнения, системы уравнений, формулы, и т. д., описывающие жизнь данного объекта с заданной точностью.

Проектирование – процесс создания описаний, необходимых для создания ещё не существующего объекта, путём переработки первичного описания, оптимизации заданных характеристик объекта или алгоритма его функционирования, путём устранения некорректности первичного описания.

Граф – математическая система, которая состоит из двух множеств: множества точек и множества линий, которые находятся между собой в некотором отношении.

Регрессионная модель – функциональная завистимость связывающая эксперементальные данные с параметрами модели.

Фактор – параметр модели, влияние которого на модель изучается вэксперименте.

Адекватность модели – параметр показывающий на сколько реально построенная регресионная модель отвечает экспериментальным данным.

ООП – объектно – ориентированный подход, подход при котором структура рассматриваемой системы рассматривается в виде классов, а данные в системе объединены с кодом для их обработки и называются обьектами.

*Цели и задачи*

Цель – повышение технико-экономических показателей при проектировании машиностроительного оборудования, На основе разработки математических моделей машиностроительного оборудования и проведения численного эксперимента.

Для достижения указанной цели в работе представленны и решены следующие основные задачи.

1. Изучение методов автоматизированного проектирования и моделирования многомассовых динамических систем с нелинейными связями.
2. Выполнение анализа информационных технологий и программного обеспечения для моделирования динамики систем.
3. Разработка методики расчёта параметров динамической системы.
4. Разработка логической структуры и модели ПМК и реализация програмного комплекса для моделирования динамических систем.
5. Разроботать структуру и определить параметры регресионной модели динамической системы.
6. С помощью разработанного програмного комплекса исследовать влияние факторов динамической системы на скорости масс в процессе выполнения технологической операции разделения меаллического лома.

# 2 РАЗРАБОТКА СТРУКТУРЫ ПРОГРАМНОГО КОМПЛЕКСА ДЛЯ АНАЛИЗА ДИНАМИКИ МЕХАНИЧЕСКИХ СИСТЕМ

## 2.1 Разработка математической модели программного комплекса для моделирования динамических систем

Итак, как мы условились выше, математическая модель динамической системы будет представлять собой систему дифференциальных уравнений, задача программного комплекса – обеспечить от пользователя ввод данных о системе, составить систему дифференциальных уравнений и решить ее, выдать пользователю результаты моделирования в виде графиков, отчётов. Мы будем рассматривать колебательную систему с одной степенью свободы. Данную систему легче всего представлять в виде графа, где узлы графа – массы , рёбра графа – связи. Проведем классификацию элементов типа «масса» и «связь». В основу классификационных признаков элементов «масса» были положены: тип распределения массы по объему и способность перемещаться (рисунок 2.1).

МАССЫ

Сосредоточенные

Распределенные

Подвижные

Неподвижные

- Узел «ИЛИ» - Узел «И»

Рисунок 2.1 – «И-ИЛИ» дерево элементов типа «масса»

При классификации элементов типа «связь» рассматривали характер зависимости силы (функциональный, интегральный, дифференциальный), вид (линейность) зависимости и влияние связи на энергию системы (рисунок 2.2).

СВЯЗИ

Комбинированные

Простые

Распределяющие  
F = f (y)

Линейные

Нелинейные

Диссипативные

F = f (y′)

Генерирующие

F = f (t)

Смешанные

F = f (y, y′, t)

Функциональные

Интегральные

Дифференциальные

Рисунок 2.2 – «И-ИЛИ» дерево элементов типа «связь»

Математически, граф представлен множеством вершин (масс) и множеством рёбер (связей) [32]. Численно граф может быть представлен матрицей инцидентности, но так хранить данные неудобно, так как за каждым элементом закреплено большое количество информации. Поэтому граф будет представлен массивом вершин и массивом рёбер. Каждое ребро будет указывать на вершины, которые оно соединяет.

Решать систему будем численным методом, осталось только выбрать метод для решения системы уравнений

Метод Эйлера.

Метод Эйлера является простейшим одношаговым методом решения задачи Коши.

Допустим, нам известна точка (xi,yi) на искомой кривой (рисунок 2.3).

e

L

Y=y(x)

Yi+1

Xi+1

Xi

h

yi

Рисунок 2.3 – Графическая интерпретация метода Эйлера

Тогда можно провести прямую линию с тангенсом угла наклона yi’=f(xi yi), которая пройдёт через точку (xi yi)[34]. Следующей точкой решения можно считать ту, где прямая L пересекает ординату, проведённую через точку . Уравнение прямой L имеет вид y=yi + y’I(x-xi). Поскольку  и , то имеем

.(2.1).

Формула (7) описывает метод Эйлера. Этот метод имеет 1-й порядок точности.

Исправленный метод Эйлера.

Возьмём значение правой части в схеме (2.1) равным среднему арифметическому между  и , то есть

.

 будем считать по формуле (2.1):

.

Подставляя это значение в формулу для получим:

 (2.2)

Метод имеет 2 – й порядок точности.

Метод Рунге – Кутта.

Метод описывается системой пяти соотношений:

где



Метод имеет 4-й порядок точности.

Вышеописанные методы являются методами с постоянным шагом, кроме этих методов существует ряд методов с переменным шагом: методы с автоматическим шагом и методы прогноза и коррекции [36]. К первым относятся: выбор шага на основе оценки ошибки и выбор шага на основе сохранения точности и устойчивости. К методам прогноза и коррекции относится метод трапеций, метод Адамса, метод Милна.

Выбор шага на основе оценки погрешности.

Для практической оценки ошибки используют правило Рунге. Обозначим через  соответственно точное решение уравнения и приближенные решения вычисленные в точке xi с шагом h и h/2. Тогда для ошибки метода интегрирования р – го порядка справедлива оценка:

(2.3)

Выбор шага на основе сохранения точности и устойчивости.

Автоматический выбор шага, конечно, проводится с целью интегрирования с размерами шагов, максимально возможными при условии сохранения необходимой точности и устойчивости.

Устойчивость вычислений в явных методах обеспечивается, если шаг выбрать по формуле:

(2.4)

где  - максимально допустимый шаг интегрирования.

При этом допустима ошибка аппроксимации  на один шаг интегрирования обычно полагается равною не больше 1% максимально возможного абсолютного значения ymax в ходе расчёта, то есть .

## 2.2 Разработка логической структуры ПМК

Наш программный комплекс будет проектироваться как комплекс двух программ. Задача первого – обеспечить ввод от пользователя информации о моделируемой системе в виде графа, записи её на диск. Задача второго считать информацию с диска построить по этим данным систему и промоделировать её работу. По результатам моделирования необходимо построить графики, выдать отчёты и т. д. Разработку логической структуры будем производить с помощью языка UML (Unified Modeling Language).

UML – Унифицированный язык моделирования, язык визуального моделирования для решения задач общего характера, который используется при определении, визуализации, конструировании и документировании артефактов программной системы. UML состоит из четырёх частей, описывающих различные аспекты системы: статические, динамические, организационные и относящиеся к окружению [37].

Разработку логической структуры начнём с представления использования, диаграмма, иллюстрирующая это представление – диаграмма использования. Представление использования описывает поведение системы с точки зрения пользователя. Деятельность разбивается на транзакции, которые называются вариантами использования (use cases). Вариант использования описывает взаимодействие системы с одним или несколькими действующими лицами (actors), в виде последовательности сообщений [38]. В понятие действующее лицо входят люди, компьютерные системы и процессы. Актёр – это идеализированная внешняя сущность, вступающая во взаимодействие с системой, подсистемой, или классом. Каждый актёр вступает во взаимодействие с одним или несколькими вариантами использования. Вариантом использования называется блок внешне наблюдаемой деятельности системы. Вариант использования описывает некоторую часть поведения системы, не вдаваясь при этом в особенности её внутренней структуры. В нашей модели выделим одного актёра – «Пользователь». Актёр взаимодействует с двумя системами: «Графический редактор» и «Расчёт колебательной системы». Разработанная диаграмма использования представлена на рисунке 2.4.

Подсистема «Графический редактор»

Пользователь

Подсистема «Расчёт колебательных систем»

Рисунок 2.4 – диаграмма использования

Описание взаимодействий пользователя с системой «графический редактор»:

«Открыть проект» - загрузка с диска ранее сохранённого проекта

«Редактирование проекта» - редактирование проекта, включает следующие взаимодействия: «Добавить массу» - создаёт в проекте ещё одну массу, «добавить связь» - создание связи, «Удалить элементы» - удаляет элементы из проекта, «Копирование» - копирует и добавляет в систему элементы.

«Сохранение» - сохранение проекта.

«Справка» - справочной информации по системе.

Описание взаимодействий пользователя с системой «расчёт колебательных систем»:

«Загрузить модель» - загружает модель с диска.

«Редактировать модель» - редактирование модели.

«Визуализация» - расчёт модели, визуализация работы модели, вывод графиков.

«Установить параметры моделирования» - Установить параметры моделирования.

В процессе проектирования пользователь взаимодействует с приложениями проектируемого ПМК. Сначала пользователь строит модель в графическом редакторе, добавляя массы и устанавливая связи между ними, задавая значения соответствующих параметров. Далее модель сохраняется в Ini – файл. Модель, созданная в графическом редакторе, открывается в расчётной программе, где пользователь может выполнить расчёт параметров модели посмотреть визуализацию работы модели, вывести графики.

Разработка статического представления модели ПМК.

Статическое представление – основа языка UML [37]. Статическое представление модели отражает структуру объектов. В статическое представление входит всё, что касается традиционных структур данных, а так же операции с этими данными. И операции и данные разбиваются на классы. Статическое представление описывает сущности обладающие поведением, в виде отдельных элементов модели, но при этом не представляет информацию о динамическом поведении этих сущностей. Статическое представление описывается диаграммами классов и объектов.

Разработка диаграммы классов для графического редактора.

Графический редактор должен обеспечить ввод информации от пользователя и сохранение её в файл для использования в программе расчётов колебательных систем. Как мы условились выше, колебательная система будет представляться в виде неориентированного графа, где каждая вершина соединяется с другой только одной связью, но в модели динамической две массы могут соединяться несколькими связями. Представлять графически два узла соединёнными несколькими связями неудобно для пользователя.В этом случае необходимо бы было изображать достаточно большие узлы, что бы можно было различить несколько связей, соединяющих эти узлы, то есть необходимо, что бы за каждой связью, которая графически изображает связь между массами, был закреплён список из связей которые соединяют эти узлы. В связи со всем выше сказанным была разработана диаграмма классов для графического редактора(см. рисунок 2.5).

StdPanel

“Dependency”

Massa

Link

List:StringList

First,next:massa

Relations: array of relation

1

2

ShowWindow()

*ShowWindow()*

*SetParam()*

*GetParam()*

“Usage”

PointMassa

Param:StringList

ClassName:String

Name:String

Relation

SupportMassa

ShowWindow()

SetParam()

GetParam()

ShowWindow()

SetParam()

GetParam()

*InitWindow*

*InitData*

*GetData*

Spring

InitWindow

InitData

GetData

Screw

InitWindow

InitData

GetData

Dempfer

InitWindow

InitData

GetData

DryFriction

InitWindow

InitData

GetData

HydroCylinder

InitWindow

InitData

GetData

Рисунок 2.5 – диаграмма классов для графического редактора

Итак, в данном представлении были выделены следующие базовые классы:

Панель (StdPanel) – собственно владелец всех объектов, на ней отрисовываются все элементы модели.

Масса (Massa) – вершина нашего графа, который представляет модель. Содержит одно свойство: List:StringList – список строк, в это свойство заносится все наименования и значения параметров объекта. Так же класс содержит три виртуальных метода: ShowWindow() – вызов диалогового окна для ввода параметров объекта, SetParam() – установка в полях ввода диалогового окна параметров объекта (установить параметры), GetParam() – записать информацию из диалогового окна в параметры объекта.

Связь (Link) –графически изображает связи между массами. Класс содержит свойства: First,next:massa – массы которые соединяет данная связь. Метод ShowWindow() вызывает диалоговое окно для ввода параметров связи.

Связь (Relation) – класс описывает абстрактный элемент соединяющий эти массы физически, содержит следующие свойства: Pram:StringList – список параметров, ClassName:String – имя класса (информация необходима для приложения расчётов). ClassName:String – имя объекта. Так же класс содержит три виртуальных метода: InitWindow – вызов диалогового окна для ввода параметров объекта, InitData – установка в полях ввода диалогового окна параметров объекта (установить параметры), GetData – записать информацию из диалогового окна в параметры объекта.

Эти классы связаны между собой следующими отношениями:

Класс StdPanel и классы Massa и Link связаны отношением зависимости, т. е. Massa и Link зависят от класса StdPanel, он является владельцем всех экземпляров данных классов. Классы Massa и Link связаны отношением ассоциации, класс Link содержит свойства типа Massa. Класс Relation используется классом Link (свойство Relations). Остальные классы наследуются от базовых: PointMassa и SupportMassa – наследованы от класса Massa. Отличие от базового класса в том, что классы потомки по-своему реализовывают методы ShowWindow(), SetParam(), GetParam(). Класс потомок знает свой набор параметров (list – определяется в конструкторе класса потомка) и знает, как заносить и извлекать данные из этого списка. Классы Spring, Screw, Dempfer, DryFriction, Hydrocylinder – наследованы от класса Relations. Эти классы описывают реальные объекты, соединяющие массы (пружина, резьбовое соединение, и т. д.), аналогично классам PointMassa и SupportMassa вышеописанные классы так же знают набор своих параметров и как с этими параметрами работать. Таким образом, данная диаграмма классов реализовывает концепцию, что за каждой связью, которая графически изображает связь между массами(Link), был закреплён список из связей, которые соединяют эти узлы(Relations). Хочу отметить, что при разработке приложения «графический редактор» конкретные свойства элементов не задаются, классы имеют только одно свойство – список параметров, что сделано с целью возможности применения данного приложения в других предметных областях

Разработка диаграммы классов для подсистемы «Расчёт колебательных систем»

Программа «Расчёт колебательных систем» должна выполнять следующее – считать данные о системе с диска или принять их от пользователя (должна иметь свой диалог ввода). Выполнить расчёт параметров элементов модели. Изобразить на экране визуально работу моделируемой системы, вывести результаты в виде графиков и файлов данных. Расчёт должен выполнятся методом, выбранным пользователем, с шагом заданным пользователем.

Выделим следующие классы:

Model – Модель, в этом классе будет записаны параметры моделирования системы, t: Tfloat – время моделирования, ht: TFloat - шаг интегрирования, e: Tfloat – точность,CalcMethod: Tmethod – метод интегрирования. Кроме того этот класс является управляющим, то есть управляет работой других классов. Класс содержит один метод, который вызывает диалоговое окно модели, в этом окне можно будет редактировать параметры модели, изменять свойства элементов модели, как мы условились выше, модель будет представлена списком масс и списком связей, выделяем ещё два базовых класса Massa и Relation. Класс Massa описывает массы в нашей системе, класс имеет следующие свойства: X0,Y0:Float – координаты центра масс;H,W:Float – высота и ширина (любой элемент структурно представляем в виде прямоугольника), f – сила, действующая на данный элемент. Класс имеет виртуальный метод Calculate(). Этот метод выполняет расчёт следующего положения в пространстве для данного элемента. Известна сила, действующая на элемент и предыдущее положение элемента в пространстве, далее решается система дифференциальных уравнений с параметрами заданными в классе Model. От данного класса наследованы следующие классы: PointMassa, SupportMassa. Класс SupportMassa описывает неподвижные элементы модели, отличие от базового в реализации класса Calculate(). Класс PointMassa (сосредоточенная масса) описывает подвижные элементы системы, кроме наследуемых свойств имеет свойства M,v0:float – масса и начальная скорость. Следующий базовый класс relation (связь) – класс описывает элементы связи, класс имеет свойства F: Tfloat – сила с которой она действует на соединяемые массы L0,L: Tfloat – длинна в свободном состоянии, и текущая длинна, Eelast: TFloat; - модуль упругости pm1, pm2:Tobject – указатели на соединяемые массы, c: Tfloat – жесткость. Класс имеет виртуальный метод CalcForce(P) – метод рассчитывает силу, с которой связь действует на соединяемые элементы, рассчитывается это следующим образом, через указатели pm1 и pm2 известны координаты центров масс и размеры соединяемых масс, можно рассчитать длину связи, зная длину связи и длину связи в свободном состоянии, можно посчитать силу с которой она действует на соединяемые связи. От класса relation наследован класс Spring (Пружина), кроме наследованных свойств имеет свойство d: Tfloat – диаметр. По своему реализует метод CalcForce. От класса spring наследованы следующие классы:

SpaceSpring (Пружина с зазором) отличается от базового класса реализацией метода CalcForce.

HydroCylinder (гидроцилиндр) имеет кроме наследуемых следующие параметры: p: Tfloat – давление, SHmax:Tfloat - максимальный ход,SH:Tfloat - ход при кодором давление максимально, по своему реализует метод CalcForce.

Cutt (разрезаемая заготовка) имеет кроме наследуемых следующие параметры: Pr: Tfloat – усилие резания, tv: Tfloat – коэффициент внедрения, по своему реализует метод CalcForce.

Dempfer (демпфер) кроме наследуемых имеет свойство Kf: Tfloat – коэффициент демпфирования, по своему реализует метод CalcForce.

Screw (резьбовое соединение) свойства: Fst:Tfloat - статическая сила действующая в начальный момент времени, z:Tfloat – затяжка, Cd, Cb: Tfloat - жесткость деталей и болта, Fd, Fb: Tfloat - силы деталей и болта, Lmax: Tfloat - длина болта, при которой стык раскрывается

StaticForce (статическая сила) свойство Ft: Tfloat – величина силы.

От класса StaticForce наследован класс GeneratorForce. Класс имеет свойства: Am: Tfloat – амплитуда,W: Tfloat – частота, Fi: Tfloat - начальная фаза, значение свойства Ft изменяется по синусоидальному закону.От класса GeneratorForce наследован класс ExpForce значение силы меняется по закону експоненты. От класса Dempfer наследованы классы:

DryFriction (сухое трение), отличается реализацией метода CalcForce.

Space (пространство), имеет свойство Kf: Tfloat – коэффициент восстановления.

От метода Screw наследован класс Butt (поставка или стык), отличается реализацией метода CalcForce.

Следует отметить, что хоть обе части нашего ПМК и оперируют одними и теми же классами (масса, связь), но диаграммы обеих программ отличаются, тат как для графического редактора в принципе, не важны физические свойства конкретных элементов. Поэтому, к примеру, дерево классов, описывающее класс связей в графическом редакторе достаточно просто (каждый класс отличается только списком конкретных параметров). На диаграмме же классов для программы расчётов эта часть имеет разветвлённую структуру, это обусловлено тем, что здесь нас интересуют конкретные физические свойства элементов и то, как этот элемент влияет на модель (реализация метода CalcForce()) Например, от класса Spring (пружина) наследован класс SpaceSpring (пружина с зазором).Диаграмма классов для программы расчётов колебательных систем приведена в приложении А. В данных диаграммах описаны лиш основные аспекты функционирования данных подсистем. Для графического редактора расматривался только аспект представления информации о системе в виде графа и передачи этой информации программе расчётов колебательных систем. Для программы расчётов рассматривался только расчёт параметров системы во времени. Такие аспекты как интерфейс пользователя (как сруктурно будет изображен узел, связь),взаимодействие с системой расчёта для редактора и такие аспекты как визуализация работы моделируемого обьекта, вывод графиков будут расмотрены в последующих разделах. Теперь приступим к разработке диаграммы обьектов.

Статическое представление модели определяет набор объектов, значений и связей могущих сущесвовать в одним снимке системы [38]. В принцепе возможной конфигурацией модели является любая совместимая со статическим представлением комбинация объектов и связей. Диаграмму объектов ещё называют диаграммой “снимка”. Диаграмма объектов изображает всю систему целиком в конкретный момент времени. Эту диаграмму будем использовать в качестве примера системы для иллюстрации сложности структуры данных. Изобразим пример подсистемы “Графический редактор” на диаграмме объектов которая приведена на рисунке 2.6.

Item[1]:SupportMassa

List=”

X0

0

y0

0

W

10

H

10”

Item[2]:PointMassa

List=”

X0

0

y0

0

W

10

H

10

M

100

V0

0”

Links1[1]:Link

First=item[1]

Next=item[2]

relations[1]:Spring

Param=”

C

100

L0

10

W

10

H

0

Description

Пружина

GraphFlag

TRUE

relations[2]:Dempfer

Param=”

K

100

W

10

H

0

Description

Демпфер

GraphFlag

TRUE

Рисунок 2.6 – диаграмма объектов для «системы графический редактор»

На диаграмме объектов система изображена в момент когда в системе создано две массы неподвижная опора (Item[1]:SupportMassa) и сосредоточенная масса (Item[2]:PointMassa), эти элементы графически соединяет связь (Links1[1]:Link) которая указывает на два объекта типа (relations[1]:Spring) и (relations[2]:Dempfer). Которые фактически соединяют эти две массы. Список параметров инициируется при создании объектов. Прежде чем приступить к следующему представлению опишем те же объекты в системе «Расчёт колебательных систем». Данная диаграмма поможет нам увидеть чёткое различие в представлении данных в этих двух системах. Диаграмма объектов, иллюстрирующая состояние той же модели в программе расчётов представлена на рисунке 2.7.

Relations[1]:Spring

Item[2]:PointMassa

Item[1]:SupportMassa

M1=item[1]

M2=item[2]

C=100

L0=10

W=10

H=0

Description=Пружина

GraphFlag=TRUE

X0=0

y0=0

W=10

H=10

M=100

V0=0

X0=0

y0=0

W=10

H=10

M1=item[1]

M2=item[2]

K=100

W=10

H=0

Description=Демпфер

GraphFlag=TRUE

relations[2]:Dempfer

Рисунок 2.7 – Диаграмма объектов в системе «Расчёт колебательных систем»

Итак, мы разработали статическую структуру нашего ПМК, теперь необходимо разработать поведение нашего ПМК, для этого необходимо рассмотреть представление взаимодействия.

Объекты осуществляют некоторое поведение путём взаимодействия между собой Представление взаимодействия является более общим и всеобъемлющим взглядом на объекты. Эта точка зрения на систему моделируется при помощи понятия кооперации. Кооперация – это описание нескольких объектов, которые взаимодействуют между собой для реализации определённого поведения. Статическое представление модели описывает присущие классу свойства. Например, связь имеет два указателя на объекты, которые связывает. Кооперация же описывает те свойства, которые конкретный экземпляр класса получает в результате участия в кооперации. У кооперации есть два аспекта – структурный и поведенческий. Поведенческий аспект кооперации описывает обмен сообщениями, между объектами, играющими в ней определённые роли [37]. Такой обмен сообщениями называется взаимодействием. Взаимодействие моделирует выполнение операции, варианта использования или другой поведенческой сущности. Сообщение представляет собой одностороннюю коммуникацию между двумя объектами в виде передачи управления от отправляющего объекта к получающему. Последовательность сообщений может быть показана на двух видах диаграмм: диаграмме последовательности, которая заостряет внимание на временной составляющей обмена сообщениями, и диаграмме коопераций, которая обращает внимание на структурные отношения между объектами. Для реализации мною выбрана диаграмма последовательностей.

Построение диаграммы последовательностей для подсистемы «Графический редактор»

Рассмотрим взаимодействие системы с пользователем, когда пользователь выполняет построение моделируемой системы, вводя в систему массы и связи. Во взаимодействие вступает внешняя сущность «Пользователь», класс главного окна приложения в роли Redactor, класс STDPanel в роли поля ввода, классы PointMassa и Support в роли соединяемых масс. Пользователь выполняет следующие действия. Сначала выбирает, какой тип массы он хочет создать, для этого необходимо нажать соответствующий переключатель на панели редактора; потом выполнить щелчок на поле ввода (STDPanel), это событие вызовет конструктор создания массы соответствующего вида.

На рисунке 2.8 изображена диаграмма последовательностей для создания двух масс.

Redactor

ToolButtonPM(Down)

StdPanel

Create()

:PointMassa

ToolButtonSM(Down)

Click()

Click()

:Support

Create()

Рисунок 2.8 – диаграмма последовательностей для создания двух масс в графическом редакторе

Теперь построим диаграмму для создания пользователем связи. Диаграмма изображена на рисунке 2.9.

Redactor

:PointMassa

:Support

ToolButtonLink(Down)

:Link

Click()

Click()

Create()

Рисунок 2.9 – диаграмма последовательностей для создания связи между двумя элементами

Для того, что бы создать связь между элементами пользователь должен выполнить следующие действия: на панели редактора установить флаг добавления связи; потом выделить две связи, для которых будет создана связь и когда пользователь отпустит кнопку мыши над второй массой, будет создана связь.

Рассмотрим взаимодействие пользователя с подсистемой «Расчёт колебательных систем». Основная задача этого приложения – выполнить расчёт параметров модели. Последовательность действий изображена на рисунке 2.10.

Model

Relations

CalcForce(p)

Masses

Calculate ()

Рисунок 2.10 – диаграмма последовательностей для процесса расчёта модели

При расчёте параметров модели выполняется следующее:

Класс Model вызывает для каждой связи метод CalcForce(p), этот метод рассчитывает с какой силой связь действует на соединяемые массы;

После того как рассчитаны реакции всех связей, Model вызывает для каждой массы метод Calculate. Этот метод, зная, какая сила, действует на массу и значение этой массы решает задачу Коши методом, выбранным пользователем с заданными параметрами. Таким образом, определяется положение каждой массы на следующем шаге, процесс продолжается до тех пор, пока не закончится время моделирования, установленное пользователем.

## 2.3 Разработка модульной структуры ПМК

Структура нашего комплекса представлена двумя блоками модулей, первый блок – блок модулей графического редактора, его задача принять от пользователя данные о системе в виде графа, записать информацию на диск. Назначение второго – считать данные и по ним построить модель системы выполнить расчёт параметров модели во времени с параметрами, которые укажет пользователь.

В состав первого блока входят следующие модули:

Управляющий модуль – управляет работой графического редактора. Реагирует на действия пользователя и вызывает методы экземпляров соответствующих классов.

Обслуживающая часть представлена блоком модулей, в которых описаны классы графического редактора.

Модуль «Tprocess» - в этом модуле описан класс Massa. Этот класс изображает узел системы, управляющий модуль создаёт экземпляры потомков класса Massa PointMassa и SupportMassa и записывает их в массив items.

Классы PointMassa и SupportMassa описаны в модулях PointMassa и SupportMassa.

Модуль «Link» - в модуле описан класс Link,управляющий модуль создаёт экземпляры класса, записывая их в массив links.

Модуль «relation» - в модуле описан класс relation.

Модули «Spring», «Screw», «Dempfer» и другие модули описывающие потомков класса relation, экземпляры этих классов создаются при создании экземпляра класса Link.

Управляющий модуль так же выполняет запись информации в файл.

В состав второго блока входят следующие модули:

Управляющий модуль – управляет работой программы расчётов.

Модуль «Model» - здесь описан класс Model, создаётся экземпляр класса. Модуль «Model» так же играет управляющую роль, так как управляет работой таких модулей как PointMassa и SupportMassa, и всеми модулями описывающими классы – потомки класса Relation.Так же считывает информацию о модели с жесткого диска.

Обрабатывающая часть.

Модули PointMassa и SupportMassa – содержат описание классов PointMassa и SupportMassa, отвечают за редактирование параметров конкретного элемента, представления выходной информации в виде графиков и визуализации.

Модули «Spring», «Screw», «Dempfer» и другие модули описывающие потомков класса relation, отвечают за редактирование параметров конкретного элемента, представления выходной информации в виде графиков и визуализации.

Модульная структура представлена на рисунке 2.11.

БД

Tprocess

Link

Управляющий модуль редактора

Управляющий модуль программы расчётов

Spring

Screw

Dempfer

Model

PointMassa

SupportMassa

Spring

Screw

Dempfer

Support

PointMassa

Рисунок 2.11 – Модульная структура проектируемого ПМК

## 

## 2.4 Виды обеспечения функционирования ПМК

Аппаратное обеспечение ПМК.

Для функционирования ПМК необходим следующий минимальный состав аппаратных средств:

* процессор Intel Pentium II и выше;
* минимум 64 MB ОЗУ;
* монитор SVGA, Видео карта.

Основные требования накладывает программа расчёта колебательных систем, так как для каждой массы на каждом шаге решается система ДУ, то при достаточно большом количестве масс в моделируемой системе, при использовании ЭВМ с меньшей производительностью значительно увеличатся затраты времени на выполнение моделирования.

Программное обеспечение ПМК.

Для функционирования ПМК необходима операционная система Windows 98, NT, 2000, XP. Приложения ПМК, графический редактор и программа расчёта колебательных систем функционируют независимо, поэтому на жестком диске могут располагаться в различных местах. Графический редактор на диске занимает 749056 байт. Программа расчётов параметров колебательных систем 1695546 байт.

Выходные данные.

Выходными данными в нашем ПМК являются параметры моделируемой системы: скорость, сила, перемещение для подвижных масс, сила для неподвижных, сила для связей. Выходные данные представляют массив (от начального момента времени до конечного момента времени с заданным шагом).

Между приложениями в ПМК предусмотрен обмен информацией с помощью ini – файла.

**2.5 Разработка рабочего проекта ПМК**

Обоснование выбора средств разработки.

В качестве средств разработки выбрана среда разработки Delphi 6.0. Выбор данной среды разработки обусловлен широкой функциональностью, распространенностью и хорошей документированностью. Delphi 6.0 в сравнении с другими аналогичными средствами визуальной разработки и поддерживающими объектно - ориентированую модель, является средой, которая наиболее быстро позволяет разрабатывать приложения.

Итак, согласно разработанной нами схеме модулей ПМК теперь приступим к непосредственной разработке модулей.

Управляющий модуль графического редактора, главный модуль приложения, который выполняет всю работу посредствам вызова вспомогательных модулей, так же этот модуль описывает главное окно приложения. В модуле описан так же вспомогательный класс ItemList – список элементов и класс linkList – список связей. Эти классы используются для описания списка выделенных объектов. На этом окне расположен элемент STDPanel (поле ввода), расположены элементы ToolBar содержащие кнопки управления. Окно реагирует на следующие события:

FormCreate (создание окна) – при создании окна динамически формируется панель, на которой установлены переключатели типов масс. Создаются экземпляры классов linkList и ItemList.

STDPanel1Click (щелчок на поле ввода) – при этом событии, если пользователь выбрал на панели тип создаваемого узла, тогда, создаётся узел в модели типа, который выбрал пользователь.

Модуль «Tprocess» - описывает класс massa. Класс наследован от класса TgraphicControl, при разработке логической структуры, мы разрабатывали только те свойства класса, которые отвечают за основную функцию приложения – формирование информации о системе для программы расчётов. Сейчас мы рассмотрим такие аспекты как изображение на экране моделируемой системы структурно (в виде графа).

класс massa имеет следующие свойства, отвечающие за отрисовку элемента:

Pic: Tpicture – пиктограмма изображающая конкретный тип массы (наследуемые классы так же имеют это свойство, значение определяется при создании объекта).

Width и Height – высота и ширина элемент отрисовывается в виде прямоугольника.

Переопределён метод Paint – выполняет отрисовку объекта в зависимости от режима (свойство Mode).

Модуль Link – описан класс Link. Так же опишем методы и свойства этого класса, описанные в данном модуле, которые отвечают за отрисовку элементов.

Элемент Связь отрисовывается в виде стрелки, свойства cx,cy,lnx,lny,xk1,xk2,yk1,yk2,x1,y1,x2,y2 – определяют параметры отображаемого элемента, через указатель на соединяемые массы можно узнать эти параметры, в классе так же переопределён метод Paint, который отрисовывает элементы.

Модуль Relation – в модуле описан базовый класс, класс Link содержит свойство Relations, которое является массивом указанного типа.

Модули «Spring», «Screw», «Dempfer» и другие модули описывающие потомков класса relation, описание работы этих модулей описано при разработке логической структуры, элементы этих классов не как не отрисовываются, они только при создании инициируют свой список параметров имеющий тип TstringList – массив строк, так же в этих модулях описаны процедуры для работы с диалоговыми окнами – InitWindow InitData GetData.

Рассмотрим структуру модулей программы расчётов колебательных систем. Модуль «Model» - управляющий модуль графического редактора.

Содержит следующие процедуры:

Create – создаёт элемент класса Model;

Destroy – уничтожение лемента;

InitInputParamDlg – инициирует диалоговое окно параметров модели;

ClearData – уничтожаются все элементы модели;

ShowDialog – выводит пользователю диалоговое окно редактирования параметров модели;

EditElementParams – вызывается окно редактирования параметров выбранного элемента;

DeleteElement – удаление из модели элемента;

SaveToIniFile – сохранить модель в файле;

LoadFromIniFile – загрузить модель с жесткого диска;

Visio – устанавливаются геометрические параметры модели, для визуализации её в окне модели;

Calculate – расчёт параметров модели, выполнение визуализации;

ShowGraphs – вывод окна графиков модели;

MakeReport – создать отчёт.

# 3 РАЗРАБОТКА ПРОГРАМНО – МЕТОДИЧЕСКОГО КОМПЛЕКСА ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИКИ МЕХАНИЧЕСКИХ СИСТЕМ

## 3.1 Структура и функциональное назначение отдельных модулей

Рассмотрим структуру основного модуля графического редактора (управляющего модуля), так как он выполняет основную функцию этого приложения – составление пользователем структуры моделируемой системы и сохранение этой информации для последующего расчёта в программе расчётов.

Частично, этот модуль был уже описан в предыдущих разделах, остановимся подробно на функциях, которые выполняет модуль.

Создание главного окна приложения.

При создании главного окна приложения автоматически заполняется панель (ToolBar) элементами (ToolButton), которые описывают каждый тип массы, происходит это следующим образом. В той же директории где находится графический редактор, создадим директорию images\, в этой директории находятся файлы со структурным изображением каждого типа массы, создадим файл images.lst. В этом файле напишем имя каждого файла с иконкой и соответствующее название этого элемента. Приложение находит соответствующий файл и создаёт кнопку на панели инструментов, в противном случае выводится сообщение об ошибке (файл не найден) и даже если существует модуль, описывающий данный подкласс, создать элемент данного класса не удастся.

Реакция на события мыши.

При щелчке (OnCLick) на панели ввода (STDPanel) если пользователь выбрал какой то тип массы нажатием на панели соответствующей кнопки [39], то будет создан экземпляр класса того объекта, который выбрал пользователь,пользователю будет выдано диалоговое окно в котором он укажет параметры создаваемого объекта экземпляр этого объекта будет занесен в массив item.(Подробнее смотри приложение Б).

При нажатии кнопки мыши (OnMauseDown) если левая кнопка нажата и не нажат правый Shift, то все выделенные объекты становятся не выделенными, запоминается точка нажатия.

При перемещении указателя мыши, если нажата левая кнопка, на панели рисуется четырёхугольник, который покажет, какие объекты будут выделены, если отпустить кнопку мыши.

При отпускании кнопки все объекты, которые попали в выделенную зону, становятся выделенными.

В главном модуле так же описаны процедуры, которые описывают реакцию элементов (масс и связей) на события мыши, так как классы наших элементов наследованы от класса TgraphicControl, то они имеют стандартные события OnCLick, OnMauseDown, OnMauseUp, OnMauseMove, при создании элементов им приписываются эти процедуры.

Событие нажатия кнопки мыши на элементе (ItemMouseDown), в этой процедуре выполняется следующее:

Если нажата левая кнопка мыши, то если не нажат Shift элемент, над которым нажата кнопка, становится выделенным, а элементы, которые были выделены до этого, становятся не выделенными. Если же Shift нажат, то к выделенным элементам добавляется элемент, над которым нажали кнопку, если этот элемент уже выделен, то его выделение отменяется, все выделенные элементы переходят в состояние pmMove.

При перемещении указателя мыши, если была нажата левая кнопка мыши, все выделенные объекты перемещаются.

Отпускание указателя мыши. (ItemMouseUp)

В этой процедуре происходит вызов конструктора связи, происходит это следующим образом, проверяется, нажал ли пользователь переключатель создания связи, далее просматриваются все связи, находится номер элемента, над которым нажата кнопка мыши, если флаг выбора первой массы не установлен, то тогда вызывается, конструктор связи. Если же установлен флаг выбора первой массы, то устанавливается флаг в значение «ложь», номер элемента, над которым отпустили кнопку, запомнили, тогда при нажатии на второй массе, если нет связи, которая уже соединяет данные элементы, произойдёт вызов конструктора связи. Далее, будет выведено диалоговое окно создания связи, в этом окне пользователь добавит элементы (пружина, резьба, и т. д.), которые в модели будут соединять эти массы, после того как пользователь закончит редактировать связь, созданный элемент будет занесён в массив связей.

Двойной щелчок над элементом (и над массой и над связью) происходит вызов диалогового окна для редактирования параметров элементов (такое же как и при создании).

Так же на главном окне приложения есть панель со стандартными элементами управления.

Создать новый проект – пользователю выводится диалоговое окно с предложением сохранить редактируемую модель, далее если пользователь выбрал сохранить, то текущая модель сохраняется в файл, если нет, не сохраняется, и все массы и связи из массивов Items и Links удаляются.

Сохранить проект – текущая модель сохраняется в файле, выбранном пользователем. Как было указанно выше. Интерфейс между графическим редактором и программой расчётов обеспечивается с помощью ini файлов. Ini файлы характеризуются секцией и именем параметра и его значением. Для каждого элемента модели будем создавать секцию, в которую мы будем записывать параметры и их значения. Так же создадим секцию, в которую запишем параметры моделирования (шаг, время метод, имя модели), секция “Model”.У элемента массы есть свойство param, список строк в котором в одной строке имя параметра, а в другой его значение. Таким образом, в цикле устраивая перебор всех элементов массива items, создаём секцию для каждого элемента, перебирая значение свойства param, записываем в секцию значение каждого параметра элемента, кроме того, в секцию записывается класс элемента. Дальше перебираем все элементы массива Links. У элементов типа Link есть свойство relations – массив, в котором записаны все элементы, которые физически связывают элементы модели. Для каждого элемента создадим секцию, в которую запишем параметры реальной связи (точно так же как и для массы), от элемента типа Link нам нужны только указатели на соединяемые массы, через эти указатели записываем в секцию номера тех масс, которые соединяет данная связь.

Открыть проект – загружается ранее сохранённая на диск модель. Происходит это следующим образом. Из файла берётся первая секция, читается её параметр, в котором записано имя класса, в зависимости от этого параметра вызывается конструктор, далее читаются строки секции и заполняются значения параметров. При этом если находится секция, в которой описана связь, и до этого не встретилось связи соединяющей те же массы, то создаётся элемент класса Link и элемент типа Screw, Spring, Dempfer и т. д. Если такая связь уже есть, то создаётся только экземпляр типа Screw, Spring, Dempfer и т. д.

Удаление элементов. При работе с моделью пользователь может, как добавлять элементы, так и удалять их. Реализовано это следующим образом, при нажатии на панели инструментов кнопки «Удалить» или кнопки «Del», на клавиатуре. В процедуре, которая закреплена за данным событием, происходит следующее. Проверяется количество выделенных масс и количество выделенных связей, если их суммарное количество больше нуля, то происходит перебор всех выделенных масс (один цикл) и перебор всех связей (вложенный цикл), если какая то связь содержит указатель на удаляемую массу, то это связь удаляется. Таким образом, удаляются все выделенные массы и связи, и все связи, которые связаны с удаляемыми массами, если удаляется связь, то её удаление не несёт за собой удаление масс.

Копирование элементов. При работе с графическим редактором у пользователя так же есть возможность скопировать ранее созданные элементы. Для копирования на панели инструментов есть кнопка «Копировать элементы». Процедура, которая выполняется при нажатии кнопки, имеет следующую структуру. В процедуре описана функция: DoCopyItem(CopyItem: Tprocess): Tprocess, выполняет следующее, создаёт элемент класса Tprocess и всем свойствам созданного объекта присваиваются значения свойств объекта, который передаётся в функцию. В самой процедуре для каждого выделенного объекта вызывается функция DoCopyItem, выделенные объекты становятся не выделенными, выделяются скопированные объекты.

Рассмотрим модуль Tprocess, в этом модуле описано диалоговое окно которое вызывается при создании связи. Рассмотрим подробно структуру функций модуля, которые связанны с добавлением связи в модель, и редактированием её свойств.

Создание диалогового окна.

Процедура создания дианового окна выполняет заполнение компонентами панель, на которой будут расположены кнопки с изображениями конкретных типов связей, происходит это, как и в главном окне приложения, только анализируется файл images1.lst.

Как уже указывалось выше, это окно вызывается при создании, это происходит следующим образом. При создании элемента класса Link после создания элемента вызывается метод ShowEdit. При работе этого метода, перебираются все компоненты свойства Relations создаваемого элемента, и для каждого вызывается метод InitData, который заполняет для каждого объекта в своём диалоговом окне значения своих свойств и добавляет в диалоговом окне связи, в список элементов этой связи своё имя.

Добавление элементов связи.

Так как графически связь изображается в виде одного элемента (Link), то для того, что бы добавить связь, реально соединяющую массы необходимо создать экземпляр какогото из классов Screw, Spring, и т. д. И записать этот экземпляр в свойство графической связи (Link) Relations. Создание экземпляров классов потомков класса Relations (Screw, Spring, и т. д.) происходит при нажатии кнопки на панели с изображением данной связи. При нажатии одной из кнопок происходит вызов конструктора данного класса и присваивание объекту уникального имени. При создании объекту присваивается имя, например «Пружина», но в принципе массы могут соединять и две пружины. Поэтому после того как создан объект, просматриваются все элементы и считается, сколько связей данного типа содержит редактируемый объект (графическая связь). После этого имя созданного объекта изменяется и становится, например «Пружина 3». Следующее действие, которое происходит при нажатии кнопки создания элемента связи – вызов диалогового окна для созданного нами элемента. Происходит это путём вызова метода InitWidow, тот метод вызывает диалоговое окно для созданного элемента и передаёт ему параметры этого объекта (в данном случае параметры по умолчанию). После редактирования свойств, пользователь принимает или отвергает параметры. Для того, что бы присвоить значение свойствам созданного объекта из диалогового окна вызывается метод GetData – записывает информацию из окна в свойства объекта.

Возможно так же удалять созданные объекты, для этого необходимо выбрать в списке элемент и нажать клавишу del или соответствующую кнопку в окне. При этом в свойстве редактируемой графической связи Relations находится элемент с таким именем и удаляется.

Рассмотрим структуру отдельных модулей программы расчётов.

Главный модуль приложения – Model, этот модуль содержит функции и процедуры расчёта параметров моделируемой системы. В данном модуле описан список масс и список связей. Порядок расчёта параметров системы был достаточно подробно рассмотрен выше, поэтому сейчас рассмотрим такой аспект функционирования системы как вывод результатов. Результатами расчёта являются для масс – список перемещений, скоростей и сил, для каждого момента времени моделирования, для связей это сила в каждый момент времени. Следует отметить, выходные данные организованы таким образом, что каждый класс «сам знает» какие данные для его экземпляров необходимо сохранять (класс содержит для каждого параметра список вещественных значений).

Итак, в модуле Model описаны функции для поддержки интерфейса с графическим редактором, это функции сохранения данных и чтения.

Кроме того, здесь описано окно редактора, в этом окне пользователь может редактировать параметры модели. Редактор содержит стандартные функции – добавление элементов, редактирование элементов, удаление. Отличие от редактора в том, что редактировать массы и связи пользователь выполняет в одном окне. Это обусловлено тем, что здесь система представляется не графически, а в виде описаний.

Расчёт параметров моделируемой системы выполняется с помощью функции Calculate, суть её работы заключается в следующем. Есть информация о каждой массе (положение и размеры) зная эти параметры в начальный момент времени можно рассчитать с какой силой каждая связь действует на соединяемые массы. Зная, какое расстояние между массами и их габариты, можем узнать, какая длинна связи в данный момент времени, а в зависимости от этого уже находится сила. Зная, какая сила действует на массу, можем рассчитать, решая задачу коши, какое положение будет занимать масса в следующий момент времени. Таким образом, происходит расчёт, до тех пор, пока не достигнем конечного времени.

Так же в модуле описано окно модели, это окно выводится пользователю для выполнения расчётов и вывода результирующей информации. В окне имеется поле, на котором изображаются элементы модели. Каждый класс элемента модели в программе расчётов имеет в предках класс TabstractElement, поэтому все экземпляры этих классов имеют свойство picture – рисунок и свойство hint – подсказка, которая выводится при наведении указателя мыши на элементе. Каждый элемент при создании загружает из файла своё изображение и размещается на поле модели по введенным параметрам. При выполнении расчёта на каждом шаге элементы на поле модели перемещаются согласно данным расчёта, таким образом, организована визуализация работы модели, по этой визуализации пользователь видит, как будут происходить колебания в моделируемой системе. Для каждого элемента определено так же событие OnClick по этому событию происходит вызов окна, на котором изображены графики для данного элемента, какие конкретно графики, «знает» сам элемент, то есть процедура заполнения этого окна описана в каждом классе.

Из окна модели можно вызвать общее окно для построения графиков, в этом окне есть поле, на котором изображаются графики, а справа находятся переключатели для каждого элемента. Заполнение этого окна происходит динамически на основе списков масс и списков связей.

В данном модуле так же описаны методы для создания отчёта по моделированию. Отчёт сохраняется в виде документа Word. Механизм создания реализован с помощью OLE Automation.

## 3.2 Информационное обеспечение

Общие сведения о программно - методическом комплексе.

Программно – методический комплекс предназначен для моделирования одномерных механических колебательных систем с произвольным числом степеней свободы. Моделируемый объект представляется в виде совокупности кинематических (массы) и топологических элементов (связи). При этом приняты следующие допущения: массы сосредоточенные (точечные), абсолютно жесткие, обладают геометрическими размерами; связи – невесомые, обладают упругостью, имеют геометрические размеры. В результате моделирования получают динамические зависимости изменения кинематических (перемещения, скорости) и силовых параметров элементов системы во времени. Комплекс представлен двумя приложениями:

Графическим редактором – программа, которая предназначена для ввода данных в систему в виде графа, при этом массы играют роль узлов в графе, связи играют роль рёбер графа;

Программой расчётов колебательных систем – программа, которая может принять данные о моделируемой системе из графического редактора и выполняет расчёт параметров, вывод результатов.

Руководство пользователя к графическому редактору.

Запуск программы.

Запуск программы выполняется одним из стандартных способов, принятых в операционной среде функционирования программы. Исполняемый файл редактора – redactor.exe. Для работы графического редактора необходимо, что бы в той же папке находились файлы images.lst и images1.lst. Файл images.lst имеет текстовый формат, в файле через пробел записаны наименование классов масс, и имя файла в котором находится рисунок с изображением данной массы. Файл images1.lst имеет такую же структуру, но в нём описаны классы связей. Также необходима папка images, в которой находятся файлы с пиктограммами элементов.

Ввод и редактирование данных

Исходными данными для моделирования являются: данные о модели и параметры моделирования. Осуществить ввод (редактирование) данных можно начать сразу после загрузки приложения, добавляя и удаляя элементы модели. Данные о модели включают сведения о структурных и топологических элементах.

Для того, что бы добавить массу необходимо выполнить следующие действия:

На панели инструментов нажать кнопку с пиктограммой той массы (смотри рисунок 3.1), которую вы хотите создать, далее выполняете щелчок левой кнопкой мыши на поле ввода, в появившемся диалоговом окне установите параметры создаваемой массы (смотри рисунок 3.2).

Создать новый проект

Кнопки выбора вида добавляемой массы

Открыть ранее сохранённый

проект

Сохранить проект

Вызов окна параметров модели

Переход в режим создания связи

Удалить элементы

Копировать элементы



Рисунок 3.1 – главное окно приложения «графический редактор»

Следует отметить, что все параметры вводимых элементов пользователь вводит сам, поэтому желательно, что бы пользователь «прикинул» схему создаваемой модели, подробнее в руководстве пользователя к программе расчётов.

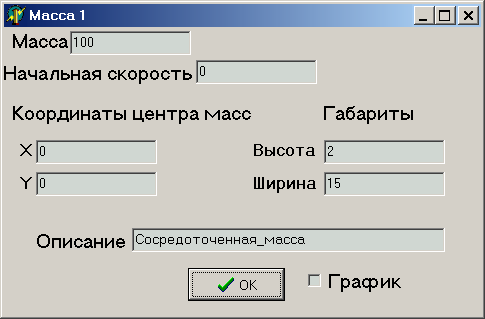


Рисунок 3.2 – окно редактирования параметров массы

Добавление связи.

Когда пользователь ввёл уже хотя бы две массы, он может добавить связи, которыми связаны эти массы, для этого необходимо выполнить следующие действия:

Нажать кнопку на панели инструментов со значком связи (см. рисунок 3.1), это будет означать, что вы перешли в состояние добавления связи, щелкаете левой кнопкой мыши сначала на первой связи, а потом на другой. После второго щелчка будет создан объект, графически соединяющий две связи. На экране появится окно редактирования параметров связи (см. рисунок 3.3), на этом этапе реально ваши массы ещё не чем не связаны, что бы добавить физические связи необходимо нажать на панели инструментов, окна редактирования параметров связи кнопку с изображением связи, которую вы хотите добавить (смотри рисунок 3.3). После этого на экране появится окно с параметрами добавляемой физической связи (рисунок 3.4), после того как пользователь введёт параметры связи, он нажимает кнопку Ок после чего в списке физических связей добавится связь выбранного вами типа. Если вы хотите удалить физическую связь из списка, выберете из списка элемент, который хотите удалить и нажмите клавишу del или кнопку «Удалить» в окне редактирования параметров связи. Предусмотрена возможность создания нескольких элементов одного типа, но с разными параметрами, при этом добавляемые элементы будут кроме названия иметь и свой номер.

Панель инструментов, для добавления физических связей между массами

Удалить физическую связь из списка

Принять параметры графической связи

Список физических связей

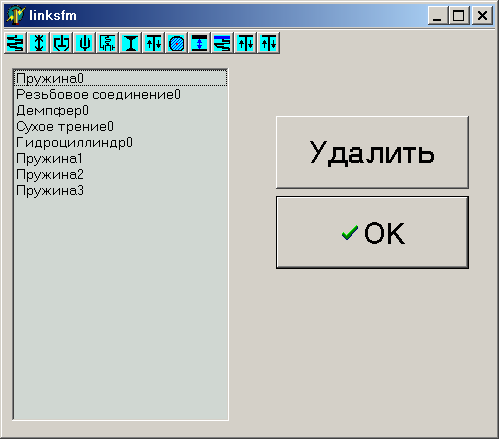


Рисунок 3.3 – Окно редактирования параметров графической связи

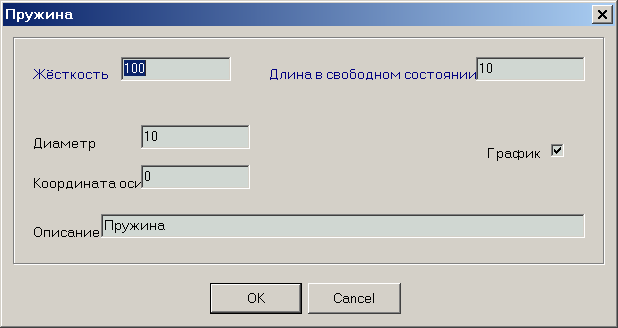


Рисунок 3.4 – диалоговое окно редактирования параметров физической связи

Для редактирования добавленных элементов выполните двойной щелчок левой кнопкой мыши над элементом, при этом в зависимости от того над каким элементом вы выполнили это действие, будет выведено соответствующее окно редактирования элемента.

Удаление элементов.

Для удаления одного элемента выделите элемент (щелчок левой кнопкой мыши на элементе), который хотите удалить и нажмите на кнопку панели инструментов с изображением корзины или нажмите клавишу del. При этом, если вы выделили массу, то удалятся все связи, связанные с удаляемой связью. Для удаления нескольких элементов выделите элементы (щелчок левой кнопкой на поле, и удерживая кнопку ведёте указатель мыши так, что бы в отрисовываемый на экране прямоугольник попали объекты которые вы хотите выделить; или удерживая на клавиатуре shift, левой кнопкой нажимайте на тех объектах, которые хотите выделить), которые хотите удалить и повторите те же действия, как и с одним элементом.

Так же выделенные объекты можно перемещать по полю ввода. Для этого выделите объект, или группу объектов, которую вы хотите переместить и нажав на одном из выделенных объектов левую кнопку мыши удерживая её переместите объекты на новое место.

Копирование элементов.

В программе предусмотрена возможность, копировать уже созданные массы, для того, что бы скопировать объекты выполните следующие действия: выделяете одну или группу масс и нажимаете кнопку с изображением красной стрелки на панели инструментов, создадутся такие же элементы, только с другими именами, появившиеся элементы будут выделенными.

В графическом редакторе предусмотрена возможность сохранения модели (проекта) в файле, одновременно с помощью этого файла осуществляется интерфейс с программой расчётов. Для сохранения модели нажмите кнопку на панели инструментов с изображением дискеты. В появившемся окне выберете файл, в котором необходимо сохранить данные. Позже этот файл можно будет открыть для того, что бы закончить создание схемы, или открыть в программе расчётов, для того, что бы промоделировать.

Для того, что бы открыть ранее сохранённый файл необходимо нажать кнопку с изображением открытой папки, в диалоговом окне выбрать файл который необходимо открыть.

Перед тем как приступить к созданию модели необходимо установить параметры моделирования. Для этого нажимаем кнопку «Параметры модели».

На экране появится окно, в котором необходимо установить требуемые параметры (см. рисунок 3.7).

После того как вы построили модель в графическом редакторе, необходимо вызвать программу расчётов.

Руководство пользователя для программы расчётов.

Запуск программы.

Запуск программы выполняется одним из стандартных способов, принятых в операционной среде функционирования программы, при этом в качестве параметра командной строки может быть указано имя файла модели, которая и будет автоматически загружена. Предусмотрен вызов из графического редактора.

Ввод и редактирование данных.

Исходными данными для моделирования являются: данные о модели и параметры моделирования. Осуществить ввод (редактирование) данных можно выбрав пункт меню *Модель/Редактирование* с помощью формы ввода данных (см. рис.3.5). Перед тем как приступить к вводу данных, желательно составить расчетную схему моделируемого объекта.

Данные о модели включают сведения о структурных и топологических элементах. В первую очередь вводятся данные о структурных элементах (массах), затем о топологических (связях). При этом на соответствующих формах параметры, влияющие на процесс моделирования, выделены синим цветом, не влияющие – черным (см. рис.3.6). Для большинства органов управления, расположенных на формах (кнопки, строки ввода и др.), существуют контекстные подсказки о их назначении, которые становятся видны при указании на данный орган управления «мышью».

Отредактировать введенные данные по конкретному элементу можно следующим способом:

выбрать в окне из списка всех элементов модели необходимый (см. рис.3.5);

нажать на кнопку редактирования (или двойное нажатие правой кнопки «мыши» на самом элементе в списке) (см. рис.3.5);

отредактировать параметры элемента модели в появившейся форме ввода данных для этого элемента (см. рис.3.6).

При вводе координат расположения элементов для визуализации следует учитывать, что ось Х расположена горизонтально и направлена вправо, ось У – вертикально и вниз. Размерность вводимых данных контролируется только пользователем. Рекомендуется использовать систему СИ (масса – кг, длина – м, сила – Н, жесткость – Н/м, давление - Па).

Окно со списком элементов модели

Панель выбора топологических элементов (связей)

Кнопка удаления текущего элемента модели

Кнопка редактирования параметров элементов модели

Панель выбора конструктивных элементов (масс)

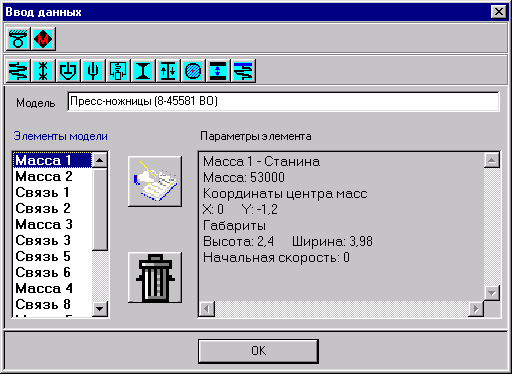


Рисунок 3.5 - Форма ввода данных

Необязательный параметр элемента модели (выделен черным цветом)

Переключатель отображения графиков изменения параметров элемента

Обязательный параметр элемента модели (выделен синим цветом)

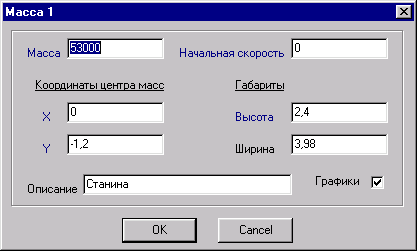


Рисунок 3.6 - Форма ввода данных для элемента модели

После ввода данных о модели следует задать параметры моделирования. Для этого выберете пункт меню *Расчет/Параметры* и на соответствующей форме (см. рис.3.7) введите требуемые данные:

численный метод решения – метод численного решения системы дифференциальных уравнений, описывающих поведение моделируемого объекта; в программе реализованы три метода: метод Рунге-Кутта 4-го порядка (рекомендуемый), методы Эйлера 2-го и 1-го порядка (порядок метода, в общем случае, отражает его точность);

время моделирования – период времени (модельный, нереальный), в течении которого исследуется поведение модели;

максимальный шаг по времени – максимально допустимый шаг изменения модельного времени (нереальное, дискретное время); зависит от выбранного численного метода решения (для Рунге-Кутта – 0,5-0,00001, для остальных – 0,1 – 0,0001);

точность вычислений – коэффициент точности вычисления шага изменения параметров при численном решении (рекомендуемое значение 1).

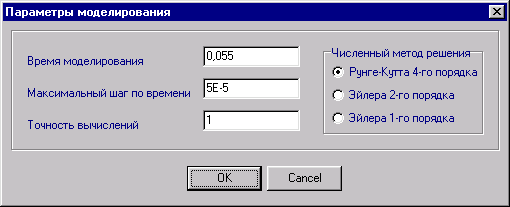


Рисунок 3.7 - Форма ввода параметров моделирования

После ввода всех этих данных можно приступать непосредственно к процессу моделирования.

Процесс моделирования

Моделирование исследуемого объекта производиться путем выбора пункта меню *Расчет/Визуализация*. После чего появляется соответствующая форма (см. рис. 3.8). Визуализация процесса моделирования позволяет не только визуально проконтролировать корректность «сборки» модели, но и наглядно отобразить перемещения элементов объекта моделирования с целью сравнения их характера (направления) с ожидаемыми.

Нажатием кнопки «GO» начинается процесс моделирования, ее повторное нажатие его приостанавливает. В процессе расчёта положение элемента на каждом шаге изменяется согласно данным расчёта, соответственно длинна связей изменяется таким образом, пользователь видит как будет «двигаться» его система. Визуализированная расчетная схема моделируемого объекта активна, т.е., переместив указатель «мыши» на определенный элемент модели, можно увидеть контекстную подсказку (тип элемента – масса или связь, а также заданное пользователем описание элемента). Выбрав же элемент модели (нажатие правой кнопки «мыши» на указанном элементе), можно просмотреть изменение его параметров (кинематических и/или силовых в зависимости от вида элемента) во времени (см. рис. 3.9).

Визуальное отображение расчетной схемы объекта моделирования

Вызов формы совместного отображения графиков изменения параметров модели во времени

Информация о модели (масса, жесткость)

Строка статуса процесса моделирования

Окно визуализации поведения модели

Запуск/остановка процесса моделирования

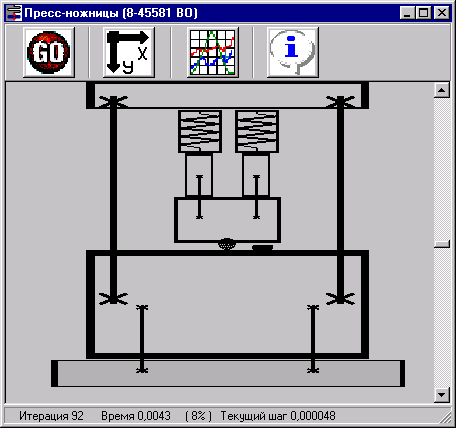


Рисунок 3.8 - Форма визуализации процесса моделирования

Вкл./выкл. график изменения скорости элемента

Вкл./выкл. график изменения действующей на элемент силы

Вкл./выкл. график перемещения

Выбор страницы отображения графиков

Выбор страницы отображения численных данных

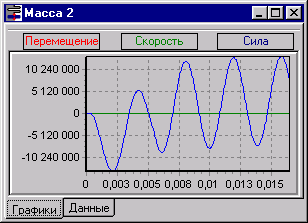


Рисунок 3.9 - Страница отображения графиков изменения параметров элемента во времени

Эта форма имеет две страницы: страницу графиков (рис.5) и страницу числовых данных (рис.3.10). На странице графиков цвет надписи на кнопке включающей/выключающей отображение графика идентичен цвету самого графика.

Окно отображения числовых данные (в следующем порядке: время, данные первого графика – например, перемещение и т.д. )

Записать числовые данные в файл на диск

Очистить список, содержащий числовые данные

Обновить числовые данные

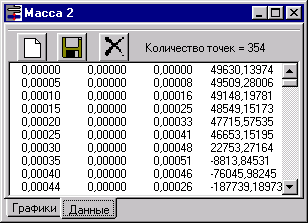


Рисунок 3.10 - Страница отображения числовых данных

Следует учесть, что режим отображения графиков для каждого элемента модели задается при вводе его параметров (отображать/не отображать – см. рис.2).

Для просмотра и наглядного сравнения графиков изменения параметров нескольких элементов предусмотрена специальная форма (см. рис.7), вызываемая нажатием кнопки «Графики». На форме расположено поле , на котором изображаются графики и выключатели, кнопки «закрыть» и «записать»(см. рис.3.11).

Кнопка записи графиков в буфер обмена

Окно отображения графиков изменения параметров элементов во времени

Вкл./откл. отображения графика изменения определенного параметра элемента

Окно выбора графиков для отображения

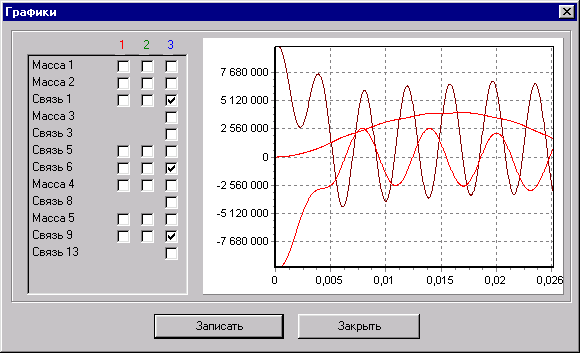


Рисунок 3.11 - Форма сопоставлений представлений графиков, изменения параметров элементов модели

Сохранение моделей.

Создаваемая модель в любой момент может быть сохранена в файле на жестком диске, если выбрать пункт меню *Модель/Записать*. В дальнейшем сохраненная модель может быть прочитана, для чего необходимо выбрать пункт меню *Модель/Загрузить* и с помощью появившейся стандартной формы выбрать файл, содержащий необходимую модель (по умолчанию файлы моделей имеют расширение \*.mdl).

Генерация отчета.

В программе предусмотрена возможность сгенерировать отчет по результатам моделирования. Для этого необходимо выбрать пункт меню *Отчет/Создать*. Отчет создается в формате документа Microsoft Word (естественно для этого он должен быть установлен на данном компьютере).

Отчет включает в себя:

* изображение расчетной схемы модели;
* описание параметров всех элементов модели;
* произвольное число выбранных пользователем графиков изменений параметров отдельных элементов модели (см. рис. 7) – при каждом нажатии кнопки «Записать» окно с изображениями графиков помещается в отчет.

**Внимание!** Если на вашем компьютере установлена программа «Лексикон - RTF конвертер» с автоматической конвертацией текстовых файлов в Word документ (о чем свидетельствует соответствующий пункт подменю в меню «Файл»), то генерация отчетов будет невозможна (по причине некорректной работы RTF-конвертера с OLE объектами). Устранение данной ошибки возможно, если удалить RTF-конвертер из автозагрузки Microsoft Word. Для чего необходимо удалить файл …\Office\Startup\lexrtf2.wll.

Сгенерированный программой отчет можно просмотреть загрузив установленный на компьютере Microsoft Word, например, сделать это можно выбрав пункт меню *Отчет/Показать* и указать требуемое имя файла отчета.

# 4 ИССЛЕДОВАНИЕ КОЛЕБАТЕЛЬНОЙ СИСТЕМЫ (ЭКСПЕРИМЕНТ)

## 4.1 Задачи исследования колебательной системы

Cфундамента

Cколоп

Lk

M1

M2

M3

M4

Рисунок 4.1 – расчётная схема пресс – ножниц

Объектом исследования является пресс – ножницы(8-45581 ВО). Этот вид оборудования работает в условиях динамических нагрузок. Резкое увеличение нагрузки чаще всего происходит, когда резко исчезает технологическая нагрузка, что и приводит к возникновению колебательных явлений. Расчётная схема представлена на рисунке 4.1.

М1 – нижняя масса (станина), М2 – верхняя (траверза) , М3 – плунжер, М4 – Узел резания, М3+М4 - ползунCфундамента – жесткость фундаментных болтов, Cколонн – жесткость колонн.

Рассмотрим такую ситуацию: при работе ножниц произошло разрушение заготовки, при этом технологическая нагрузка исчезает, под действием силы (которая возникла в результате разрушения заготовки) узел резания начинает резко увеличивать свою скорость, сила через болтовые соединения передаётся плунжеру (возможно раскрытие стыка или разрушение болтов) , а через него на траверсу, увеличивается сила которая действует на колонны, далее на станину, станина действует на фундамент, потом под действием силы реакции опоры фундамента станина начинает двигаться вверх (сильная нагрузка на фундамент, разрушение). Таким образом на работу оборудования будут влиять следующие факторы:

1 Масса узла резания, этот фактор будет оказывать существенное влияние на характер колебаний протекающих в системе, очевидно, что чем меньше масса этого узла, тем меньше сила, которая является источником колебаний, но узел резания должен иметь достаточную массу для выполнения технологической операции, необходимо подобрать оптимальный вариант.

2 Отношение массы станины к массе узла резания, чем больше масса станины, тем меньше амплитуда колебаний, которые она будет совершать и соответственно меньше действие на фундамент, но с увеличением массы станины увеличивается её стоимость, затраты на транспортировку, и т. д.

3 Отношение массы траверзы к массе станины, важность этого фактора аналогична станине.

4 Жесткость колонн, чем жестче колонна, тем больше сила, действующая на колонны, но колонна меньшей жесткости должна быть более затянутой, для сохранения не раскрытия стыка, что связано с большей трудностью на их установку, а так же необходим более качественный материал.

5 Отношение жесткости колонны к жесткости стыка, сила действующая на колонну зависит от жесткости стыка, так же данный фактор оказывает значительное влияние на характер колебаний всей системы.

Итак, нами выделено пять факторов, которые влияют на характер колебаний, в качестве отклика выберем следующие параметры:

Максимальная скорость узла резания, сила, действующая на станину

Задачи експеримента.

Найти уравнение регрессии 2-го порядка и выполнить статистический анализ модели.

1. Построить графики зависимости отклика от каждого из факторов Y=f(Xi) при фиксированных значениях остальных факторов (каждый рисунок должен содержать 3-4 кривые).
2. Используя двумерные сечения поверхности, выполнить анализ влияния факторов в изученных интервалах их изменения на функцию отклика

## 4.2 Построение модели плана II порядка

Модель второго порядка выбрана нами так как модель первого порядка скорее всего будет не адекватна, а модели более высоких порядков сложны в построении и в принципе, возможно почти любую модель аппроксимировать параболой. Для построения плана II порядка можно использовать следующую модель [40]:

|  |  |
| --- | --- |
| . | (4.1) |

Для этого необходимо провести эксперимент так, чтобы каждый фактор варьировался на трех уровнях. Простейшим решением этой задачи является план типа 3k. Реализация этого плана для k>3 требует большого числа опытов.(в нашем случае 35=243)

Для построения модели второго порядка обычно используют ортогональный план первого порядка в качестве ядра, на котором достраивается план второго порядка, поэтому такие планы называются композиционными и соответствуют шаговой идее построения планов [43].

Для удобства работы с приведенной моделью II порядка, с помощью обозначений (4.2) преобразуем ее к виду (4.2’):

|  |  |
| --- | --- |
|  | (4.2) |
| . | (4.2’) |

Задача заключается в том, чтобы по результатам наблюдений определить значения коэффициентов bi, дисперсии и доверительные границы для них, а также определить их значимость.

Согласно МНК, для нахождения коэффициентов bi, необходимо минимизировать функцию:

|  |  |
| --- | --- |
| , | (4.3) |

где N – количество опытов;

xui –значение i-й переменной в u-м опыте;

yu – значение экспериментальных y в u-м опыте;

Из условия минимизации функции ss, можно получить систему нормальных уравнений НМК:

|  |  |
| --- | --- |
|  | (4.4) |

Представив все результаты в матричной форме, получим:

|  |  |
| --- | --- |
| , , , | (4.5) |

где X – матрица условий эксперимента; Y – матрица результатов опытов;B – матрица коэффициентов.

Умножив транспонированную матрицу X на матрицу X, получим матрицу системы нормальных уравнений, которая называется информационной матрицей Фишера (матрицей моментов):

|  |  |
| --- | --- |
|  | (4.6) |

Умножив транспонированную матрицу X на матрицу Y, получим:

|  |  |
| --- | --- |
| , | (4.7) |

Используя данные обозначения, систему нормальных уравнений можно записать в матричной форме:

|  |  |
| --- | --- |
| . | (4.8) |

Обозначая обратную матрицу моментов как:

|  |  |
| --- | --- |
|  | (4.9) |

получим выражение для матрицы коэффициентов:

|  |  |
| --- | --- |
|  | (4.10) |

## 4.3 Кодирование факторов

Кодирование факторов используется для перевода натуральных факторов в безразмерные величины, чтобы построить стандартную план – матрицу эксперимента.

Для перевода заполняется таблица кодирования факторов на двух уровнях. В качестве 0-го уровня обычно выбирается центр интервала, в котором предполагается вести эксперимент.

Связь между кодовым и натуральным значениями фактора:

|  |  |
| --- | --- |
|  | (4.11) |

где Xi – натуральное значение фактора;

Xi0 –значение этого фактора на нулевом уровне;

δI – интервал варьирования факторов.

Составим таблицу кодирования факторов, используя исходные данные.

Таблица 4.1 – Таблица кодирования факторов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Интервал варьирования и уровень факторов | Мползуна, кг\*1000 | Мнижн/Мползуна | Мверхн/Мнижн | Сколлонны ГН/м | Сколлонны/С2 |
| Нулевой уровень xi=0 | 14 | 6,5 | 0,75 | 65 | 1,25 |
| Интервал варьирования δI | 6 | 3,5 | 0,25 | 35 | 0,75 |
| Нижний уровень xi=-1 | 8 | 3 | 0,5 | 30 | 0,5 |
| Верхний уровень xi=+1 | 20 | 10 | 1 | 100 | 2 |
| Кодовые обозначения | X1 | X2 | X3 | X4 | X5 |

**4.4 Составление план – матрицы**

В план – матрице должны быть указаны все возможные комбинации уровней факторов. В таблице 4.2 приведена Расширенная план – матрица плана типа Bk [45].

Таблица 4.2 – Расширенная план – матрица плана типа Bk

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № оп. | X0 | X1 | x2 | X3 | X4 | X5 | x1x2 | X1x3 | x1x4 | X1x5 | X2x3 | X2x4 | X2x5 | X3x4 | X3x5 | X4x5 | X12 | X22 | X32 | X42 | X52 | Vp |
| 1 | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | 5.38 |
| 2 | + | - | + | + | + | + | - | - | - | - | + | + | + | + | + | + | + | + | + | + | + | 7.83 |
| 3 | + | + | - | + | + | + | - | + | + | + | - | - | - | + | + | + | + | + | + | + | + | 5.10 |
| 4 | + | - | - | + | + | + | + | - | - | - | - | - | - | + | + | + | + | + | + | + | + | 6.90 |
| 5 | + | + | + | - | + | + | + | - | + | + | - | + | + | - | - | + | + | + | + | + | + | 5.34 |
| 6 | + | - | + | - | + | + | - | + | - | - | - | + | + | - | - | + | + | + | + | + | + | 7.50 |
| 7 | + | + | - | - | + | + | - | - | + | + | + | - | - | - | - | + | + | + | + | + | + | 4.98 |
| 8 | + | - | - | - | + | + | + | + | - | - | + | - | - | - | - | + | + | + | + | + | + | 6.65 |
| 9 | + | + | + | + | - | + | + | + | - | + | + | - | + | - | + | - | + | + | + | + | + | 5.39 |
| 10 | + | - | + | + | - | + | - | - | + | - | + | - | + | - | + | - | + | + | + | + | + | 7.62 |
| 11 | + | + | - | + | - | + | - | + | - | + | - | + | - | - | + | - | + | + | + | + | + | 5.10 |
| 12 | + | - | - | + | - | + | + | - | + | - | - | + | - | - | + | - | + | + | + | + | + | 6.93 |
| 13 | + | + | + | - | - | + | + | - | - | + | - | - | + | + | - | - | + | + | + | + | + | 5.35 |
| 14 | + | - | + | - | - | + | - | + | + | - | - | - | + | + | - | - | + | + | + | + | + | 7.51 |
| 15 | + | + | - | - | - | + | - | - | - | + | + | + | - | + | - | - | + | + | + | + | + | 5.00 |
| 16 | + | - | - | - | - | + | + | + | + | - | + | + | - | + | - | - | + | + | + | + | + | 6.67 |
| 17 | + | + | + | + | + | - | + | + | + | - | + | + | - | + | - | - | + | + | + | + | + | 5.38 |
| 18 | + | - | + | + | + | - | - | - | - | + | + | + | - | + | - | - | + | + | + | + | + | 7.62 |
| 19 | + | + | - | + | + | - | - | + | + | - | - | - | + | + | - | - | + | + | + | + | + | 5.11 |
| 20 | + | - | - | + | + | - | + | - | - | + | - | - | + | + | - | - | + | + | + | + | + | 6.92 |
| 21 | + | + | + | - | + | - | + | - | + | - | - | + | - | - | + | - | + | + | + | + | + | 5.34 |
| 22 | + | - | + | - | + | - | - | + | - | + | - | + | - | - | + | - | + | + | + | + | + | 7.51 |
| 23 | + | + | - | - | + | - | - | - | + | - | + | - | + | - | + | - | + | + | + | + | + | 4.99 |
| 24 | + | - | - | - | + | - | + | + | - | + | + | - | + | - | + | - | + | + | + | + | + | 6.66 |
| 25 | + | + | + | + | - | - | + | + | - | - | + | - | - | - | - | + | + | + | + | + | + | 5.43 |
| 26 | + | - | + | + | - | - | - | - | + | + | + | - | - | - | - | + | + | + | + | + | + | 7.67 |
| 27 | + | + | - | + | - | - | - | + | - | - | - | + | + | - | - | + | + | + | + | + | + | 5.12 |
| 28 | + | - | - | + | - | - | + | - | + | + | - | + | + | - | - | + | + | + | + | + | + | 6.93 |
| 29 | + | + | + | - | - | - | + | - | - | - | - | - | - | + | + | + | + | + | + | + | + | 5.38 |
| 30 | + | - | + | - | - | - | - | + | + | + | - | - | - | + | + | + | + | + | + | + | + | 7.55 |
| Продолжение таблицы 4.2 – Расширенная план – матрица плана типа Bk | | | | | | | | | | | | | | | | | | | | | | |
| 31 | + | + | - | - | - | - | - | - | - | - | + | + | + | + | + | + | + | + | + | + | + | 5.01 |
| 32 | + | - | - | - | - | - | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | 6.70 |
| 33 | + | -1.596 | 0.000 | 0.000 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2.547 | 0.000 | 0.000 | 0.000 | 0.000 | 7.65 |
| 34 | + | 1.596 | 0.000 | 0.000 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2.547 | 0.000 | 0.000 | 0.000 | 0.000 | 4.88 |
| 35 | + | 0.000 | -1.596 | 0.000 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 2.547 | 0.000 | 0.000 | 0.000 | 5.41 |
| 36 | + | 0.000 | 1.596 | 0.000 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 2.547 | 0.000 | 0.000 | 0.000 | 6.35 |
| 37 | + | 0.000 | 0.000 | -1.596 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 2.547 | 0.000 | 0.000 | 6.13 |
| 38 | + | 0.000 | 0.000 | 1.596 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 2.547 | 0.000 | 0.000 | 6.27 |
| 39 | + | 0.000 | 0.000 | 0.000 | -1.596 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 0.000 | 2.547 | 0.000 | 6.33 |
| 40 | + | 0.000 | 0.000 | 0.000 | 1.596 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 0.000 | 2.547 | 0.000 | 6.21 |
| 41 | + | 0.000 | 0.000 | 0.000 | 0.000 | -1.596 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 0.000 | 0.000 | 2.547 | 6.22 |
| 42 | + | 0.000 | 0.000 | 0.000 | 0.000 | 1.596 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.000 | 0.000 | 0.000 | 0.000 | 2.547 | 6.21 |

## 4.5 Полученная регрессионная модель

По результатам эксперимента согласно формулам, приведенным выше была построена модель второго порядка, описывающая зависимость влияния факторов модели (Х1-Х5) на максимальную скорость движения ползуна (Y). Так как параллельные опыты не проводились, то опыты на воспроизводимость не проверялись, по этому было проведено несколько опытов в центре плана. По результатам этих опытов была посчитана дисперсия опытов. Дисперсия опытов рассчитывалась по формуле [43];

(4.11)

Результаты приведены в таблице 4.3

Таблица 4.3 – расчёт дисперсии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Yi | Yср | Yср-Yi | S2(y) |
| 1 | 6.180000 | 6.205 | 0.0006250000 | 0.00211 |
| 2 | 6.240000 | 0.0012250000 |
| 3 | 6.280000 | 0.0056250000 |
| 4 | 6.200000 | 0.0000250000 |
| 5 | 6.200000 | 0.0000250000 |
| 6 | 6.260000 | 0.0030250000 |
| 7 | 6.200000 | 0.0000250000 |
| 8 | 6.080000 | 0.0156250000 |

Далее по формуле (4.10) были рассчитаны коэффициенты регрессии и проверены на значимость по критерию Стьюдента [44]:

|  |  |
| --- | --- |
|  | (4.12) |

Если неравенство выполняется, значит, коэффициент – значим с надежностью 1-α. В случае невыполнения неравенства, незначимый коэффициент принимается равным 0, а остальные коэффициенты не пересчитываются.

Значимые коэффициенты регрессии:

B0= 6.137

B1= -0.975

B2= 0.274

B3= 0.015

B4= 0.007

B5= -0.001

B12= -0.120

B13= -0.022

B14= -0.006

B15= 0.004

B23= -0.008

B24= 0.023

B25= 0.007

B34= 0.014

B35= 0.060

B45= 0.003

B11= 0.050

B22= -0.101

B33= 0.025

B44= 0.052

B55= 0.031

Модель имеет следующий вид:

Y=6.14-0.98X1+0.27X2+0.02X3+0.01X4-0.001X5-0.12X1X2-0.02X1X3-0.01 X1X4+0.004X1X5-0.01X2X3+0.02X2X4+0.01X2X5+0.01X3X4+0.06X3X5+0.003X4X5+0.05X11-0.10 X22+0.02 X33+0.05 X44+0.03 X55(4.13)

Проверка адекватности модели.

Адекватность модели проверяется с помощью критерия Фишера [41]:

|  |  |
| --- | --- |
| , | (4.14) |
| , | (4.15) |

где Sад2 – дисперсия адекватности, рассчитываемая по формуле (4.15);

Sy2 – дисперсия опыта;

α=0.05 - уровень значимости;

fад=N-l, число степеней свободы дисперсии адекватности;

fy=N(m-1), число степеней свободы дисперсии опыта;

l – количество значимых коэффициентов.

Если неравенство (15) выполняется, значит, модель адекватна.

Табличное значение критерия Фишера: 17.085.

Значение Sад2 по расчётам равно 0,033. дисперсия – 0,002. Значение критерия Фишера: 16.5 – вывод модель адекватна.

**4.6 Графическое представление полученной модели**

Для иллюстрации свойств полученной модели построим графики зависимости функции отклика от каждого фактора. При этом мы будем изображать на одном графике 3 линии. Изменяя кроме этого ещё один фактор на трёх уровнях. Итак, изобразим зависимость скорости движения ползуна от первого фактора (массы ползуна), при этом меняя значение второго фактора (отношения массы станины к массе ползуна) на рисунке 4.2.



3

2

1

Рисунок 4.2 – графики зависимости функции отклика от первого фактора

На рисунке изображены следующие графики:

1 – при значении отношения массы станины к массе ползуна – 1; уравнение имеет вид: y = 0.05x2 - 1.119x + 6.538;

2 - при значении отношения массы станины к массе ползуна – 0; уравнение имеет вид:y = 0.05x2 - 0.9991x + 6.343;

3 – при значении отношения массы станины к массе ползуна – -1; уравнение имеет вид: y = 0.05x2 - 0.8791x + 5.946;

На графике чётко видно что, с увеличением массы ползуна скорость его значительно уменьшается, что, очевидно, что чем тяжелее ползун, тем он более инертен, следовательно, сила, которая действует на него «разгоняет» его до меньшей скорости. Так же видно, что с увеличением второго фактора значение скорости возрастает.

Теперь изобразим график, на котором будет показана зависимость функции отклика от отношения массы станины к массе ползуна, при неизменных всех остальных факторах, так же изобразим три линии при различных значениях массы ползуна (см. рисунок 4.3).



3

2

1

Рисунок 4.3 – График зависимости функции отклика от отношения массы станины к массе ползуна

На рисунке изображены следующие графики:

1 – при значении массы ползуна – -1; уравнение имеет вид:

y = -0.1008x2 + 0.416x + 7.392;

2 - при значении массы ползуна – 0; уравнение имеет вид:

y = -0.1008x2 + 0.296x + 6.343;

3 – при значении массы ползуна – 1; уравнение имеет вид:

y = -0.1008x2 + 0.296x + 6.343;

График в принципе, показывает тоже, что уже было сказано выше, при увеличении отношения массы станины к массе ползуна (массы станины), скорость увеличивается, это обусловлено следующим:

В момент разрушения заготовки колонны начинают «стягивать» обе мессы – верхнюю и нижнюю, следовательно, чем тяжелее станина, тем она более инертна и двигается с меньшей скоростью, значит, увеличивается скорость верхней части вместе с ползуном. Влияние мессы ползуна было уже описано выше.

Следующий фактор, влияние которого на модель мы покажем – отношение массы станины к массе верхней поперечины. Как и на предыдущих графиках, остальные факторы неизменны, варьируем только массу ползуна (на последующих графиках так же), это обусловлено тем, что влияние массы ползуна на функцию отклика наиболее значимо. На рисунке 4.4 изображена зависимость функции отклика от отношения массы станины к массе верхней поперечины.

Рисунок 4.4 - зависимость функции отклика от отношения массы станины к массе верхней поперечины

3

2

1

На рисунке изображены следующие графики:

1 – при значении массы ползуна – -1; уравнение имеет вид:

y = 0.025x2 + 0.1029x + 7.579;

2 - при значении массы ползуна – 0; уравнение имеет вид:

y = 0.0249x2 + 0.0811x + 6.4321;

3 – при значении массы ползуна – 1; уравнение имеет вид:

y = 0.0249x2 + 0.0589x + 5.3851;

В начале значение скорости ползуна при увеличении отношения так же возрастает, но уже влияние этого фактора на модель значительно меньше.

Четвёртый фактор – значение жесткости колонн.

На рисунке 4.5 изображена зависимость функции отклика от жесткости колонн.



3

2

1

Рисунок 4.5 - зависимость функции отклика от жесткости колонн

На рисунке изображены следующие графики:

1 – при значении массы ползуна – -1; уравнение имеет вид:

y = 0.052x2 + 0.0533x + 7.602;

2 - при значении массы ползуна – 0; уравнение имеет вид:

y = 0.052x2 + 0.0467x + 6.439;

3 – при значении массы ползуна – 1; уравнение имеет вид:

y =0.0519x2 + 0.041x + 5.376;

При увеличении жесткости колонн значение функции отклика с начала падает, а со значения фактора –0,4 начинает возрастать, то есть на данном промежутке у модели есть точка перегиба.

Зависимость скорости ползуна от отношения жесткости колонны к жесткости стыка изображена на рисунке 4.6.

На рисунке изображены следующие графики:

1 – при значении массы ползуна –1; уравнение имеет вид:

y = 0.0307x2 + 0.0731x + 5.3651

2 - при значении массы ползуна –0; уравнение имеет вид:

y = 0.0312x2 + 0.0689x + 6.4378

3 - при значении массы ползуна – 1; уравнение имеет вид:

y = 0.0307x2 + 0.0731x + 5.3651

Рисунок 4.6 - Зависимость скорости ползуна от отношения жесткости колонны к жесткости стыка

3

2

1

С увеличением отношения жесткости колонны к жесткости стыка скорость ползуна увеличивается. Из всего выше описанного вытекает, что наибольшее влияние на модель оказывает первый фактор (при варьировании этого фактора скорость меняется почти на две единицы, варьирование же других влияет на модель на уровне десятых долей). Причём график близок линейной модели, такая тенденция наблюдается при варьировании почти всех факторов (второй фактор (отношение массы станины к массе ползуна) имеет наименьший радиус кривизны, значение коэффициента при x2 наибольшее –0,1).

# 5 ОХРАНА ТРУДА

## 5.1 Анализ опасных и вредных производственных факторов

Проанализируем условия труда работника – программиста. Рабочее место имеет следующие характеристики:

площадь 10 м2, объем 30 м3,

Компьютер:

Монитор – Samsung SAMTRON 55E, процессор Intel Pentium Celeron 800MHz, 128Mb ОЗУ, CD – ROM, 52 – х скоростной.

Все производственные факторы в соответствии с классификацией по ГОСТ 12.0.003 – 74 можно подразделить на опасные и вредные.

Опасный производственный фактор – это фактор, воздействие которого на работающего может привести к травме или другому внезапному резкому ухудшению здоровья.

Вредный производственный фактор – это фактор, воздействие которого на работающего в определенных условиях может привести к профессиональному заболеванию или снижению работоспособности.

Опасные производственные факторы:

Общие опасности несчастного случая на соответствующем рабочем месте: падения, подскальзывания, падение предметов и т. п.

Ушибы при наклонах под стол, чтобы дотянутся (например, присоединить или отсоединить провода) до часто помещаемого там системного блока.

Электрический шок при попытке починить неисправное оборудование, розетки и т. д.

Возможность возникновения пожаров.

Ожоги в результате случайного контакта с горячими поверхностями внутри лазерного принтера.

Вредные производственные факторы:

физические факторы:

операторы персональных компьютеров постоянно подвергаются электромагнитной неионизирующей радиации (в основном низкочастотной и сверх низкочастотной), электростатическим и магнитным полям, испускаемым компьютерами;

уровень шума на компьютеризованном рабочем месте обычно намного меньше предельного; однако, даже негромкий шум (особенно на высоких тонах) от принтеров и мониторов может восприниматься как раздражающий фактор.

б) химические факторы:

* 1. общие химические опасности соответствующего рабочего окружения;
  2. контакт с веществами, специфичными для компьютеризованных рабочих мест (тонер принтеров, озон, испускаемый лазерными принтерами) в правильно проветриваемых помещениях ниже предельного уровня и не представляет опасности: однако, он может стать опасным в плохо вентилируемой комнате, содержащей несколько лазерных принтеров и копировальных машин.

в) биологические факторы: операторы компьютеров, работающие в густо населенных и/или неадекватно вентилируемых комнатах (например, в подвалах) могут подвергаться повышенному риску инфекционных заболеваний.

г) эргономические, психосоциальные и организационные факторы:

1. проблемы со зрением (синдром дискомфорта глаз), вызванные излишней визуальной нагрузкой, продолжительной концентрацией на экране, неподходящим углом зрения, плохим освещением, отблесками, мерцанием изображения, плохой комбинацией цветов;
2. астенопические симптомы: напряжение глаз, боль в глазах, головная боль;
3. окулярные симптомы: сухие глаза, раздраженные глаза, слезящиеся глаза;
4. визуальные симптомы: расплывчатое изображение, двойное изображение, остаточное изображение;
5. проблемы, связанные с ношением очков (особенно мультифокальных) и контактных линз (в случае сухости глаз);
6. травма повторного растяжения или синдром профессионального излишнего употребления в результате продолжительной работы с клавиатурой, мышью и другими средствами ввода и обработки информации;
7. расстройства спины, шеи и плечей в результате продолжительного раздельного или совместного воздействия таких факторов, как: неправильная сидячая поза, эргономически неадекватный стул, неправильно (особенно слишком высоко) расположенный монитор, отсутствие соответствующей опоры для запястий, ног и т. д.;
8. когнитивные проблемы взаимодействия человека и компьютера в результате недостаточного учета человеческого фактора при разработке программных продуктов (трудно использовать графический интерфейс пользователя, неоднозначные подсказки и указания, неполная справка и т. д.);
9. психологические проблемы (психологический барьер) адаптации работников (особенно постарше) к работе на компьютере и информационной перегрузки;
10. психолингвистические проблемы, связанные с использованием неродного языка (в основном английского) и неадекватно переведенными текстами документации и руководстве по применению;
11. психосоциальные проблемы увеличивающейся рабочей нагрузки, ожидаемого увеличения производительности и недостатка личной независимости из–за увеличения доступности (за пределами рабочего дня и рабочего места) работника с портативным компьютером;
12. проблемы, связанные с увлечением компьютерами (хакеры, интернетоманы т. д.).

Воздействие этих факторов приводит к возникновению различных заболеваний у пользователей ПВЭМ.

## 6.2 Разработка мероприятий по обеспечению безопасных и комфортных условий труда

Рабочее место – это место постоянного или временного пребывания рабочего в процессе трудовой деятельности. Были проведены следующие мероприятия обеспечению безопасных и комфортных условий труда:

Организация рабочих мест предусматривает:

- правильное размещение рабочего места в рабочем помещении;

- выбор эргономического обоснованного рабочего положения, рабочей мебели;

- рациональную компоновку оборудования на рабочих местах;

- учет характера и особенностей трудовой деятельности.

Площадь, выделенная для одного рабочего места с ПК, не меньше 6 м2, а объем – не меньше 20м3. Рабочее место с ЭВМ размещаются на расстоянии не меньше 1 м от стен.

Рабочее помещение имеет следующие характеристики: длина помещения 10 м., ширина 6 м. и высота 3 м.

ДНАОП 0.00 – 1.31 – 99 регламентирует требования к организации рабочего места пользователя ВДТ. Лучше разместить рабочие места с ВДТ рядами, причем относительно окон они должны размещаться так, чтобы естественный свет падал сбоку, преимущественно слева. Это даст возможность исключить зеркальное отражение на экране источников естественного света (окон) и попадание последних в поле зрения пользователей. Организация рабочего места пользователя ВДТ должна обеспечивать соответствие всех элементов рабочего места и их взаимного расположения эргономическим требованиям ГОСТ 12.2.032 – 78. «ССБТ Рабочее место при выполнении работ сидя. Общие эргономические требования»; характеру и особенностям трудовой деятельности.

При размещении мест необходимо придерживаться таких требований:

рабочие места с ВДТ размещаются на расстоянии не меньше 1 м. от стен со световыми проемами;

расстояние между боковыми поверхностями видеотерминалов должно быть не меньше 1.2 м.;

расстояние между тыльной поверхностью одного терминала и экраном другого не должно быть меньше 2.5 м.;

проход между рядами рабочих мест должен быть не меньше 1 м.

Требования относительно расстояния между боковыми поверхностями ВДТ и расстояния между тыльной поверхностью одного ВДТ и экраном другого учитываются также при размещении рабочих мест с видеотерминалами и персональными компьютерами в сопредельных помещениях, с учетом конструктивных особенностей стен и перегородок.

При необходимости высокой концентрации внимания во время выполнения работ с высоким уровнем напряженности сопредельные рабочие места с ВДТ необходимо отделять одно от другого перегородками высотой 1.5 и 2 м.

Требования к конструкции рабочего стола, стула, подставки для ног на рабочих местах с ВДТ определяются ДНАОП 0.00 – 1.31 – 99. Конструкция рабочего стола должна отвечать современным требованиям эргономики и обеспечивать оптимальное размещение на рабочей поверхности всего оснащения и используемых приспособлений с учетом их размеров и конструктивных особенностей. Высота рабочей поверхности стола для ВДТ должна быть в пределах 680 – 800 мм., а ширина и глубина – обеспечивать возможность выполнения операций в зоне достижимости моторного поля. Рекомендованные размеры стола: высота – 725 мм., ширина – 600 – 1400 мм., глубина – 800 – 1000 мм. Рабочий стол должен иметь пространство для ног высотой не меньше 600 мм., шириной не меньше 500 мм., глубиной на уровне колен не меньше 450 мм., на уровне извлеченной ноги – не меньше 650 мм.

Рабочий стол для ВДТ, как правило, может быть оборудованный подставкой для ног шириной не меньше 300 мм. и глубиной не меньше 400 мм., с возможностью регулирования по высоте в пределах 150 мм. и угла наклона опорной поверхности – в пределах 20. Подставка должна иметь рифленую поверхность и бортик на переднем крае высотой 10 мм. Применение подставки для ног тех, у кого ноги не достают пол, если рабочее сидение находится на высоте, нужной для обеспечения оптимальной рабочей позы, является – обязательным.

Рабочий стул пользователя ВДТ должен иметь такие основные элементы: сидение, спинку и стационарные или съемные подлокотники.

Конструкция рабочего стула должна обеспечивать поддержание рациональной рабочей позы во время выполнения основных производственных операций, создавать условия для изменения позы. Поэтому стул должен быть подъемно – поворотный и регулироваться по высоте и углам наклона сидение и спинки, а также расстояния спинки от переднего края сидения, высоте подлокотников.

Регулирование каждого параметра может быть независимым, плавным или ступенчатым, иметь надежную фиксацию. Усилие при этом не должны превышать 20 Н. Ход ступенчатого регулирования элементов сидения может составлять для линейных размеров 15 – 20 мм., для угловых – 2 - 5.

Ширина и глубина сидения должны быть не меньше 400 мм. Высота поверхности сидения может регулироваться в границах 400 – 500 мм., а угол наклона поверхности – от 15 вперед до 5 назад. Поверхность сидения может быть плоской, передний край – округленный.

Высота спинки сидения должна составлять 30020 мм., ширина – не меньше 380 мм., радиус кривизны в горизонтальной плоскости – 400 мм. Угол наклона спинки должен регулироваться в границах 0 - 30 относительно вертикального положения. Расстояние от спинки к переднему краю сидения должно регулироваться в границах 260 – 400 мм.

Для снижения статического напряжения мышц рук необходимо применять стационарные или съемные подлокотники длиной не меньше 250 мм., шириной – 70 мм., которые регулируются по высоте над сидением в пределах 23030 мм. и по расстоянию между подлокотниками в пределах 350 – 500 мм.

Сидение, спинка и подлокотники стула должны быть полумягкими, нескользким, таким, которые не электризуется, и воздухопроницаемым покрытием, материал которого обеспечивает возможность легкой очистки от загрязнения.

Конструкция производственной мебели для пользователя ВДТ должна быть такой, чтобы обеспечивать поддержание оптимальной рабочей позы с такими эргономическими характеристиками: ступни ног – на полу или на подставке для ног, бедра – в горизонтальной плоскости, верхние (плечевые) части рук – вертикальные; угол локтевого сустава (между плечом и предплечьем) – 70 - 90; запястья согнутые под углом не больше 20относительно горизонтальной плоскости; наклон головы вперед в пределах 15 - 20 к вертикали.

Размещение экрана ЭВМ обеспечивает удобство зрительного наблюдения в вертикальной плоскости под углом 30 от линии взгляда пользователя.

Наилучшие зрительные условия и возможность распознавания знаков достигается такой геометрией размещения, когда верхний край видеотерминала находится на высоте глаз, а взгляд направлен вниз на центр экрана. Поскольку при работе на ЭВМ наиболее удобным считается наклон головы вперед, приблизительно на 20 градусов от вертикали (при таком положении головы мышцы шеи расслабляются), то экран видеотерминала тоже наклонен назад приблизительно на 20 от вертикали. Экран и клавиатура располагаются на оптимальном расстоянии от глаз пользователя, но не ближе 600 мм, с учетом размера цифровых знаков и символов.

Расстояние от экрана до глаз пользователя в зависимости от размера экрана приведено в таблице 6.1.

Таблица 6.1 - Расстояние от экрана до глаз пользователя в зависимости от размера экрана

|  |  |
| --- | --- |
| Размер экрана по диагонали | Расстояние от экрана до глаз, мм |
| 35-38 см(14″/15″) | 600-700 |
| 43 см(17″) | 700-800 |
| 48 см(19″) | 800-900 |
| 53см(21″) | 900-1000 |

Требования к клавиатуре:

- изготовление клавиатуры в виде отдельного устройства с возможностью свободного перемещения;

- присутствие опорного устройства, которое дает возможность изменять угол наклона клавиатуры в пределах 5-15;

- высота на уровне первого ряда не больше 15 мм;

- выделение цветом местом расположения отдельных групп клавиш;

- присутствие углублений посередине клавиш.

Если у конструкции клавиатуры не предусмотрено пространства для опоры ладонь, то ее необходимо размещать на расстоянии не меньше 100 мм от края стола.

Размещение принтера или другого устройства ввода-вывода информации обеспечивает хорошую видимость экрана ЭВМ, возможность ручного управления устройством ввода-вывода в зоне доступности моторного поля: по высоте 900- 1300 мм, по глубине 400-500 мм[45]. Для снижения отрицательного воздействия излучений компьютера на здоровье пользователей Санитарными правилами и нормами СанПиН 2.2.2.452.96 предусмотрен режим труда и отдыха в зависимости от выполняемой работы и возраста.

При вводе информации количество знаков за смену не превышает 40000. Время непосредственной работы с компьютером, независимо от вида и ее общей продолжительности, не превышает 6ч. СанПиН 2.2.2.542.96 предусматривает проведение через определенные промежутки времени обязательные перерывы в работе для отдыха.

Длительность перерывов установлена 15мин с периодичностью через каждые 2ч при вводе информации, а при считывании ее с экрана дисплея перерыв установлен 15мин через каждые 1.5-2 ч или 10мин через каждый час. При выполнении комплексной работы (ввод и считывание) периодичность перерывов через 1ч продолжительностью 15мин [3].

Для преодоления утомления предусмотрено выполнение комплекса упражнений для глаз через каждые 20-25мин.

В СанПиН 2.2.2.542.96 приведен комплекс “физкультминуток”, которые включают упражнения: для общего воздействия, улучшения мозгового кровообращения, снятия утомления мышц, плечевого пояса и рук, для снятия утомления туловища и ног.

При установлении режима труда и отдыха СанПиН 2.2.2.542.96 учитывает, что на рабочих местах состояние воздушной среды, освещения, шум, вибрация, и параметры ионизирующих электромагнитных излучений соответствует требованиям санитарных норм.

Температура в зависимости от периода года установлена в интервале от 21 до 25С при относительной влажности от 40% до 60%. Уровень шума не превышает 50дБА.

Работники в соответствии с приказом Минздрава в целях предупреждения у них профессиональных заболеваний проходят предварительные и периодические медицинские осмотры. При определении оптимального режима труда и отдыха учтено, что при тяжелых или острых формах заболеваний работа с компьютером недопустима или требуются длительные периоды отдыха.

Кроме перечисленных выше мероприятий по обеспечению безопасных и комфортных условий труда, осуществляются и другие, которые включают: воздух рабочей зоны, освещенность, электробезопасность и пожарная безопасность.

Оптимальными параметрами температуры при почти неподвижном воздухе являются 19-21˚С, допустимыми - 18 и 22˚С в соответствии с ГОСТ 12.1.005-88 [46].

Комфортное состояние при данных температурах воздуха определяется влагосодержанием, составляющим 10 г/м3, допустимое - не ниже 6 г/м3, что для указанных температур эквивалентно относительной влажности, представленной в таблице 6.2.

Таблица 6.2 - Величина абсолютной и относительной влажности в зависимости от температуры воздуха

|  |  |  |
| --- | --- | --- |
| Температура воздуха,°С | Влажность воздуха, % | |
| Оптимальная | Допустимая |
| 18 | 65 | 39 |
| 19 | 62 | 37 |
| 20 | 58 | 35 |
| 21 | 55 | 33 |
| 22 | 52 | 31 |

Наилучший обмен воздуха осуществляется при сквозном проветривании. Другой путь обеспечения воздухообмена, может быть, достигнут, установлением в оконных проемах автономных кондиционеров типа БК-1500, БК-2500, БК-2000Р.

Расчет необходимого количества кондиционеров проводиться по теплоизбыткам.

Режим работы кондиционера обеспечивает максимально возможное поступление наружного воздуха, но не менее 50% от производительности кондиционера.

При проектировании помещений предусмотрена приточно-вытяжная вентиляция. Подача воздуха производиться в верхнюю зону малыми скоростями из расчета создания подвижности воздуха на рабочем месте менее 0.1 м/с, лучше через подшивной перфорированный потолок. Вытяжка - естественная из верхней зоны стены, противоположной оконным проемам.

Для повышения влажности воздуха используются увлажнители или устанавливаются емкости с водой типа аквариумов вблизи отопительных приборов.

Содержание кислорода в помещении в пределах 21-22 об.%. Двуокись углерода не превышает 0,1 об.%, озон – 0,1 мг/м3, фенол – 0,01мг/м3, хлористый винил – 0,005 мг/м3.

В помещении ограничено использование полимерных материалов для отделки интерьера и оборудования.

Защита от шума имеет большое значение. Шум, неблагоприятно воздействуя на организм человека, вызывает психические и физиологические нарушения, снижающие работоспособность и создающие предпосылки для общих и профессиональных заболеваний и производственного травматизма. Шум возникает, в основном, при работе принтеров, вентиляционной установки и, в меньшей степени, самого компьютера. В соответствии с ГОСТ 12.1.003-83 для помещений конструкторских бюро, программистов вычислительных машин и т. д. приняты нормативные значения уровня шума - 45 дБ (при частоте 1000 Гц) и уровня звука 50 дБА, а уровни звукового давления в октавных полосах частот со среднегеометрическим значением 63, 125, 250, 500, 1000, 2000, 4000, 8000 Гц - соответственно 71, 61, 54, 49, 45, 42, 40 и 38 дБ ГОСТ.ССБТ.12.1.003-83 «Шум. Общие требования безопасности. (СТ.СЭВ 1930-79)» и «Санитарные нормы допустимых уровней шума на рабочих местах» N3223-85, утв. Минздравом СССР 12.03.85).

Звукоизоляция ограждающих конструкций помещений отвечает нормативным требованиям согласно главы СHиПП-12-77.

Основными мерами защиты от шума являются строительно-планировочные методы (акустическая обработка помещений). Под акустической обработкой помещений понимается облицовка части внутренних поверхностей ограждений звукопоглощающими материалами, а также размещение в помещении штучных поглотителей, представляющих собой свободно подвешенные объемные поглощающие тела различной формы. Звукопоглощающие облицовки размещаются на потолке и в верхних частях стен при высоте помещения не более 6-8м таким образом, чтобы акустически обработанная поверхность составляла не менее 60% от общей площади ограничивающих помещение поверхностей.

Для снижения уровня шума потолок или стены выше панелей (1.5-1.7 м от пола), а иногда и стены и потолок облицовываются звукопоглощающим материалом с максимальным коэффициентом поглощения в области частот 63-8000Гц.

В узких и очень высоких помещениях целесообразно облицовку размещать на стенах, оставляя нижние части стен необлицованными, либо проектировать конструкцию звукопоглощающего подвесного потолка.

Дополнительным звукопоглощением в помещении могут быть занавеси, подвешенные в складку на расстоянии 15-20 см от ограждения, выполненного из плотной тяжелой ткани. Ширина занавеси должна быть в 2 раза больше ширины оконного проема.

Освещение рабочего места - важнейший фактор создания нормальных условий труда. Практически возникает необходимость освещения как естественным, так и искусственным светом. Первый случай характерен для светлого времени суток и при работе в помещениях, в которых имеются проемы в стенах и крыше здания, во втором случае применяются соответствующие осветительные установки искусственного света. Конструктивно проемы могут быть различными по исполнению и местонахождению.

От этого зависит и характер освещенности:

- она может быть боковой, если световые проемы (окна) расположены в наружных стенах;

- верхней, если световые проемы устроены в крыше;

- комбинированной.

Если отсутствует достаточная освещенность поля зрения работающего равномерно распределенным световым потоком, надо создать такое искусственное освещение, при котором суммарный световой поток от всех установленных светильников распределяется равномерно. Так как при работе программиста необходима высокая точность выполнения работ, то к освещению предъявляются специфические требования. Наиболее удобным здесь является комбинированное освещение (СНиП II-4-79).

Как правило, на ВЦ используют люминесцентные лампы. Эти лампы имеют высокую световую отдачу (до 75 лм/Вт), большой срок службы (до 10000 часов) и хорошую цветопередачу. К недостаткам люминесцентных ламп относятся малая единичная мощность при больших размерах ламп и значительное снижение светового потока к концу срока службы. Для общего и местного освещения помещений общественных и промышленных зданий применяют лампы типа ЛБ 18-1, ЛДЦ 18 и ЛБ 58 и т.д.

В осветительных установках (ОУ) помещений следует использовать систему общего освещения, выполненную потолочными или подвесными люминесцентными светильниками, равномерно размещенными по потолку рядами, параллельно светопроемам, так, чтобы экран монитора находился в зоне защитного угла светильника, и его проекция не приходилась на экран монитора.

Работающие за видеотерминалами не должны видеть отражение светильников на экране ЭВМ.

Применять местное освещение при работе на ЭВМ в помещении не рекомендуется. Выбор светильников проводиться с учетом ограничения прямой и отраженной блескости. Для ограничения отраженной блескости необходимо тесно увязывать взаиморасположение светильников и экранов мониторов.

Люминесцентные светильники должны включаться рядами.

Минимальная освещенность рабочей поверхности стола рекомендуется в пределах 400-500 люкс.

Яркость экрана устанавливается равной 0.5 или более яркости рабочей поверхности стола при освещенности 400-500 люкс.

Допустимая величина дискомфорта, одного из основных качественных параметров ОУ, регламентируемого для ограничения прямой блескости не должна превышать 15. При проектировании ОУ следует пользоваться инже­нерным методом оценки слепящего действия ОУ по дискомфорту.

Величина коэффициента пульсации не может превышать 10%, для чего следует применять многоламповые светильники при электромонтаже ОУ.

Для освещения помещения рекомендуется применять светильники серий ЛПО13, ЛПО31, ЛПО33 исполнение 001 и 006, ЛСО02, ЛСО04 с металлической экранирующей решеткой и непрозрачной боковинами.

В качестве источников света рекомендуется использовать люминесцентные лампы мощностью 40 Вт или энергоэкономичные мощностью 36 Вт типа ЛБ, ЛХВ и ЛЕЦ как наиболее эффективные и приемлемые с точки зрения спектрального состава, цветовая температура (Тцв) излучения которых находится в диапазоне 3500-4200 К.

На уровень освещенности помещения оказывает влияние цветовая отделка интерьера и оборудования, их отражающая способность.

Для уменьшения поглощения света потолок и стены выше панелей (1.5-1.7м), если они не облицованы звукопоглощающим материалом, окрашиваются белой водоэмульсионной краской (коэффициент поглощения не должен быть менее 0.7). Допускается окраска стен до потолка цветом панелей. Для окраски стен и панелей рекомендуется применять светлые тона красок (коэффициент естественной освещённости 0.5-0.6 ). Предпочтение следует отдавать холодным тонам: светло-голубому, светло-зеленому, светло-серому. Допускается окраска стен светло-желтым, светло-бежевым цветом или цветом слоновой кости. Нельзя окрашивать стены, расположенные напротив экрана монитора, более темными тонами красок (коэффициент естественной освещённости 0.3-0.4). Оконные переплеты рам, подоконники следует окрашивать белой масляной краской.

Hа окнах монтируются занавеси, по цвету гармонирующие с окраской стен. Занавеси не должны пропускать естественный свет и полностью закрывать оконные проемы.

Особые требования должны быть предъявлены к электробезопасности помещения при комплектации его видеотерминалами с электропитанием 200 - 230 В (ГОСТ 12.1030-81, ГОСТ 12.1.009-76). Помещения должны быть оснащены устройствами защитного отключения. Электророзетки, находящиеся на рабочих местах, нужно располагать в труднодоступном месте. Свободные розетки необходимо закрыть заглушками. Должны быть соблюдены нормы, препятствующие легкому извлечению сетевых вилок из розеток (на розетках устанавливаются защитные кожухи).

Средства вычислительной техники требуется устанавливать и подключать в строгом соответствии с инструкциями по их эксплуатации и обязательно заземлить. Провода электропитания не должны свешиваться со столов или висеть под столами. Необходимо исключить возможность случайного касания ногами проводов или электророзеток.

Для обеспечения электробезопасности при работе за компьютером требуется предусмотреть защитное заземление. Защитное заземление - это преднамеренное соединение с заземляющим устройством металлических частей электроустановок и корпусов оборудования, которые вследствие нарушения изоляции могут оказаться под напряжением. В результате защитного заземле­ния величина отпускающего тока не превышает 6 мА (переменный ток) и 15 мА (постоянный ток) при условии нахождения человека под током не более 30 секунд. Кроме того, на корпусах оборудования могут возникать заряды статического электричества. Для жизни человека они не опасны, но могут привести к неприятным ощущениям. Для уменьшения величины заряда статического напряжения покрытие пола следует выполнять из материала, поглощающего статическое электричество.

По пожароопасности автоматизированное рабочее место работника относится к категории Д.. Источниками воспламенения могут быть электрические искры, дуги, короткое замыкание, перегруженные приводы, перегретые опорные поверхности, неисправная аппаратура. Окислителем, как правило, служит кислород.

Для электронных устройств характерно частое появление источников открытого огня при коротких замыканиях, пробоях и перегрузках, однако мощность и продолжительность действия этих источников воспламенения сравнительно малы, поэтому горение, как правило, не развивается. Возникновение пожара в электронных устройствах возможно, если применяются сгораемые и трудносгораемые материалы и изделия. Выбор общепромышленного и взрывозащищенного электрооборудования зависит от класса помещения.

К противопожарным мероприятиям в электроосвещении относится правильный выбор типов светильников с учетом условий, в которых они эксплуатируются. Важнейшее значение имеет правильный выбор и соблюдение режимов эксплуатации электроприборов.

Пожарная профилактика основывается на исключении условий, необходимых для горения, использование принципов обеспечения безопасности. При обеспечении пожарной безопасности решаются 4 задачи: предотвращение пожаров и возгораний; локализация возникших пожаров; защита людей и материальных ценностей; тушение пожаров.

Пожарная безопасность обеспечивается предотвращением пожаров и пожарной защитой. Предотвращение пожара достигается исключением образования горючей среды и источников зажигания, а так же поддерживанием параметров среды в пределах, исключающих горение. Предотвращение образования источников зажигания достигается следующими мероприятиями: соответствующим исполнением, применением и режимом эксплуатации машин и механизмов; устройств молниезащиты зданий и сооружений; ликвидация условия для самовозгорания; регламентацией допустимой температуры и энергии теплового разряда и др.

Пожарная защита реализуется следующими мероприятиями (ГОСТ 12.1.004-76):

- применением негорючих и трудногорючих веществ и материалов;

- ограничением количества горючих веществ;

- ограничением распространения пожара;

- применением средств пожаротушения;

- регламентация пределов огнестойкости;

- создание условий для эвакуации людей;

- применение противодымной защиты;

- пожарная сигнализация.

Существуют требования к системам отопления, вентиляции, освещения и электроустановок. Защита от распространения пламени в вентиляционных установках достигается с помощью огнепреградителей, быстродействующих заслонок, отсекателей и т. д.

Основными огнегасительными веществами являются вода, водные растворы, водяной пар, пена, углекислота, сжатый воздух, порошки, песок, земля.

Из средств пожаротушения в помещении необходимо иметь огнетушители углекислотные (ОУ, ОУ-2, ОУ-2а, ОУ-5, ОУ-8, ОУ-2ММ, ОУ-5ММ) или порошковые (ОП-1, ОП-2, ОП-2Б, ОП-8Б, ОП-5, ОП-10), которые позволяют тушить пожары в помещениях с вычислительной техникой.

**6.3 Расчет общего естественного освещения**

Расчет искусственного освещения произведем согласно предложенной методике [45].

Рассчитаем естественное освещение в помещении:

длина помещения А=10 м;

ширина помещения В=6 м;

высота Н=3 м.

Естественное освещение в помещении создаётся солнечным светом через световые проёмы и подразделяется на боковое освещение и верхнее. Для проектируемого рабочего места выбираем боковое освещение.

Конечной целью расчёта естественного освещения является определение отношение площади световых проёмов к площади пола помещения помещений и сравнение его с максимально допустимым.

Расчёт естественного освещения производится в следующем порядке. Определяем разряд выполняемых работ по зрительной характеристике и нормированное значение коэффициента естественной освещённости (КЕО) в зависимости от вида освещённости. При работе с приложением пользователю придется работать с текстовыми данными (поля ввода в диалоговых окнах), данный класс работ отнесём к средней категории.

Ен=1.5.

Отношение площади световых проёмов к площади пола помещения помещений при боковом освещении определяется по формуле:

(5.1)

где  - площадь световых проёмов при боковом освещении м2;

- площадь пола помещения, м2;

;

Кз – принимается по данным табл. 5 [45];Кз=1.5

 - световая характеристика окон, определяем по таблице 4 [45]; отношение длинны помещения к его глубине – 10/6=1,5; расстояние от уровня рабочей поверхности до верха окна – 1м; ;

Кзд – коэффициент, учитывающий затенение окон противоположными зданиями, определяем по таблице 5[45].

Кзд=1;

 - общий коэффициент светопропускания материала, определяемый по формуле:

(5.2)

 - коэффициент светопропускания материала, определяем по таблице 6[45]; =0,8 (стекло оконное листовое двойное);

 - коэффициент, учитывающий потери света в переплётах светопроёма, определяем по таблице 6[45];

=0,75 (переплёты одинарные);

 - коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении =1;

 - коэффициент, учитывающий потери света в солнцезащитных устройствах, определяем по таблице 7[45];

=1 (убирающиеся регулируемые жалюзи и шторы);



 - коэффициент, учитывающий повышение КЭО при боковом освещении благодаря свету, от поверхностей помещения и подстилающего слоя, прилегающего к зданию, определяем по таблице 8. Определим отношение глубины помещения В к высоте от уровня условной рабочей поверхности до верха окна В/н=6/2,5=2,4. Определим отношение расстояния расчётной точки от наружной стены к глубине помещения В 3/6=0,5. Принимаем средневзвешенный коэффициент отражения потолка, стен, пола равным 0,5, отношение длинны помещения к его глубине 1,5

=1,1;

Отношение площадей:



По таблице 12 требуемое значение >=0.25, таким образом, требования выполняются.

# ВЫВОДЫ

Таким образом, в ходе выполнения работы были решены следующие задачи.

1. Изучены методы автоматизированного проектирования и моделирования многомассовых динамических систем с нелинейными связями, наиболее целесообразно, в данной ситуации, авляется применение численных методов решения систем дифференциальных уравнений.
2. Выполнен анализ информационных технологий и программного обеспечения для моделирования динамики систем, рассматривались системы AnyLogic и ПА9, были выявлены недостатки и достоинства данных систем.
3. Разработана методика расчёта параметров динамической системы, данная методика описана в разделе 3.1.
4. Разработана логическая структура и модель ПМК. Выполнена реализация програмного комплекса для моделирования динамических систем, для реализации была выбрана среда разработки Delphi 6, разработанный програмный комплекс позволяет выполнить моделирование динамической системы с одной степенью свободы. Графический редактор разработан таким образом, что может быть изменён таким образом, что с помощью программы расчёта можно будет млделировать не только механисеские системы.
5. Разроботана структура и определены параметры регресионной модели динамической системы. Были выделены пять параметров модели в качестве факторов и один параметр модели в качестве функции отклика. Была построена план – матрица плана типа Вк.
6. С помощью разработанного програмного комплекса исследовалось влияние факторов динамической системы на скорости масс в процессе выполнения технологической операции разделения меаллического лома. Согласно построенной план – матрицы реализовывались эксперименты. По данным эксперимента была построена регресионная модель второго порядка, были проверены на значимость коэффициенты модели. Модель проверена на адекватность.

**7. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1 Полонская Е. Форум информационные технологии в ХХI веке// Компьютер плюс программы.- 2003.-№4.- С. 34 – 39.

2 Пахомов С. Современные процессоры// Компьютер пресс.- 2003.- №4.- С. 28 – 31.

3 Каньковски П. Устройства хранения данных: технологии будущего// Компьютер пресс.- 2003.- №4.- С. 12 - 17.

4 Елманова Н. Borland StarTeam// Компьютер пресс. 2003, №4.- С. 20 – 25.

5 Шляхтина С. Прохоров А. Рынок информационных технологий в 2002 – 2003 г.// Компьютер пресс. –2003.- №1.-С. 5 – 12.

6 Татарников О. Виртуальная периферия, или новые методы управления компьютером// Компьютер пресс.- 2003.- №1.- С. 23 – 32.

7 Букина В. Индустрия ПО в России// Компьютер пресс.- 2003.-№1.-С. 20 - 31.

8 Грибачёв К. Bold – инструмент реализации MDA в Delphi// Компьютер пресс.- 2003.-№2.-С.- 10 – 20.

9 Виноградов К. Windows, Delphi для защиты секретов// Компьютер плюс программы.- 2003.- №1.- С.- 41 – 56.

10 Черенковский К. Трёхмерные виртуальные миры: будущее или настоящее// Компьютер плюс программы.- 2003.-№1.-С. 34 – 43.

11 Безуглый Д. Технология разработки программного обеспечения// Компьютер плюс программы.- 2003.- №2.- С. 54 – 67.

12 Чеботарёв А. Power designer 9.5 последнее слово проектирования от Sybase// Компьютер плюс программы.- 2003.-№2.- С. 57 – 68.

13 Дорн Т. Облегчить работу конструктору// Компьютер плюс программы.- 2003.-№3.-С. 34 – 47.

14 Капицин И. Что бы не было мучительно больно// Компьютер плюс программы.- 2003.-№1.-С. 31 – 67.

15 Потураев В.Н., Миронюк А.Ф. О динамической нагрузке на звенья гидравлических прессов при вырубке. /Кузнечно-штамповочное производство.-1963.- №12.- С. 23-26.

16 Васильев Б.П. Динамические процессы в гидравлическом прессе при мгновенном снижении рабочей нагрузки /Кузнечно-штамповочное производство.- 1965.- №3.- С. 27-32.

17 Синицкий В.М. Исследование динамики быстроходных гидравлических прессов. /Кузнечно-штамповочное производство.- 1995.- №8.- С. 22-25.

18 Баничук Н.В., Иванова С.Ю., Шаранюк А.В. Динамика конструкций. Анализ и оптимизация. - М.: Наука, 1989. - 262 с.

19 Гусев В.Н., Карамышкин В.В., Иванов В.А., Прывчев П.В. Анализ на ЭВМ динамических усилий в прессах при вырубании нежестких материалов. /В сб. Методы исследования динамических систем на ЭВМ. - М.: Наука.- 1984.- С. 112‑117.

20 Крейнин Г.В., Павлов Б.И., Цуханова Е.А. Применение обобщенных моделей при моделировании динамики машин. /В сб. Методы исследования динамических систем на ЭВМ. - М.: Наука.- 1984.- С. 29‑36.

21 Петренко А.И., Ладогубец В.В., Чкалов В.В Автоматизация схемотехнического проектирования в машиностроении. - К.: УМК ВО, 1988. ‑180 с.

22 Гелерман Б.М. Исследование динамических процессов в гидравлических системах кузнечно-прессовых машин. /Кузнечно-штамповочное производство.- 1989.- №2.- с. 27‑31.

23 Кузнечно-штамповочное оборудование /Под ред. А.Н. Банкетова, Е.Н. Ланского. – М.: Машиностроение, 1982. – 576 с.

24 Кожевников С.Н. Динамика нестационарных процессов в машинах. - Киев: Наукова думка, 1986. - 286 с.

25 Орлов П.И. Основы конструирования. Кн.2. – М.: Машиностроение, 1988. - 544 с.

26 Прочность, устойчивость, колебания. Т. 3 /Под ред. И.А. Биргера, Я.Г. Пановко – М.: Машиностроение, 1968. – 568 с.

27 Быков В. П. Методическое обеспечение САПР в машиностроении. – Л.: Машиностроение, 1989. – 255 с.

28 Справочник конструктора штампов: Листовая штамповка /Под ред. Л. И. Рудмана.– М.: Машиностроение, 1988. – 496 с.

29 Скворцов Г. Д. Основы конструирования штампов для холодной листовой штамповки. – М.: Машиностроение, 1970. – 320 с.

30 Моисеева Н. К. Выбор технических решений при создании новых изделий. – М.: Машиностроение, 1980. – 181 с.

31 Гелерман Б.М. Исследование динамических процессов в гидравлических системах кузнечно-прессовых машин. /Кузнечно-штамповочное производство.- 1989.- №2- С. 27‑31.

32 Горбатов В.А. основы дискретной математики М.: Высш. Шк., 1986 –311 с.

33 Зыков А.А. Основы теории графов М.: Наука, 1987.- 384 с.

34 Корячко В.П., Курейчик В.М., Норенков И.П. Теоретические основы САПР. – М.: Энергоатомиздат, 1987.-400 с.

35 Турчак Л.И Основы численных методов. – М.: Наука 1987.-320 с.

36 Справочник по САПР / под редакцией В.И. Скурихина К.: Техника 1988. – 375 с.

37 Рембо Дж., Якобсон А., Буч Г.,UML: специальный справочник. – СПБ.: Питер, 2002 – 656 с.: с ил.

38 Рембо Дж., Якобсон А., Буч Г., Унифицированный процесс разработки програмного обеспечения СПБ.: Питер, 2002 – 656 с.: с ил.

39 Кэнту М. Delphi для профессионалов СПБ.: Питер, 2000 - 1125 с.

40 Новик Ф.С., Арсов Я.Б. Оптимизация процессов технологии металлов методами планирования экспериментов. – М.: Машиностроение; София: Техника, 1980. – 304 с., ил.

41 Конспект лекцiй з дисциплiни «Ймовiрнiснi процеси и математична статистика в автоматизованих системах» (для студентiв заочного вiддiлення фаху 7.080402) / Скл.: Г.Б. Бiлик, О.В.Веремiй. – Краматорськ: ДДМА, 2000. – 80с.

42 Конспект лекцiй по дисциплiнi «Методы синтезу та оптимiзацii» (для студентiв заочного вiддiлення спецiальностi 7.080402) / Скл.: Г.Б.Бiлик, О.В.Веремiй. – Краматорськ: ДДМА, 2000. – 88 с.

43 Методические указания к выполнению раздела «Охрана труда и окружающей среды » в дипломных проектах (для студентов специальности 0510) / Сост.: М.Д. Гридасов, В.И.Шейко, Е.П. Шульгин. – Краматорск: КИИ, 1990. – 35 с.

44 Жидецкий В.Ц., Джигирей В.С., Мельников А.В. Основы охраны труда. Учебник. – Изд. 2 – е, дополненное. – Львов: Афиша, 2000 – 351 с.

45 Справочник по охране труда на промышленном предприятии / К. Н. Ткачук, Д. Ф. Иванчук, Р. В. Сабарно, А. Г. Степанов. - К.: Техника, 1991.-112с.

# ПРИЛОЖЕНИЕ А

Диаграмма классов для программы расчётов колебательных систем

*CalcForce(P);*

PointMassa

Calculate()

M,v0:float

CalcForce(P

SpaceSpring

CalcForce(P

HydroCylinder

ExpForce

CalcForce(P)

Am: TFloat;

W: Tfloat

Fi: TFloat

GeneratorForce

CalcForce(P)

CalcForce(P)

Ft: TFloat

StaticForce

Butt

Space

CalcForce(P)

Pr: Tfloat

tv: TFloat

Cutt

“Usage”

\*

2

“Usage”

F: TFloat;

L0,L: TFloat

Eelast: TFloat; pm1, pm2:TObject;

c: TFloat;

Relation

*Calculate()*

X0,Y0:Float

H,W:Float;f:float

Massa

Calculate()

SupportMassa

ShowDialog():integer;

t: TFloat

ht: TFloat;

e: TFloat;

CalcMethod: TMethod;

Model

CalcForce(P

d: TFloat;

Spring

CalcForce(P

DryFriction

CalcForce(P

Kf: TFloat

Dempfer

CalcForce(P

Eelast:TFloatFst: TFloat;

z, L0b, L0d: Tfloat

Cd,Cb:TFloat

Screw

CalcForce(P

Kf: TFloat

CalcForce(P)

# ПРИЛОЖЕНИЕ Б

#### Главный модуль графического редактора

unit Unit1;

interface

uses

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

ImgList, ComCtrls, ToolWin, ExtCtrls, STDPanel, StdCtrls, Process, Link,

Menus, comobj,activeX,

OleServer, FileCtrl, ShellAPI, Variants, IniFiles, Buttons;

type

TLinkList = class(TList)

private

function GetItem(Index: Integer): TLink;

procedure SetItem(Index: Integer; Value: TLink);

public

property Items[Index: Integer]: TLink read GetItem write SetItem;

function Add(Item: TLink): Integer;

procedure Clear; override;

end;

TItemList = class(TList)

private

function GetItem(Index: Integer): TProcess;

procedure SetItem(Index: Integer; Value: TProcess);

public

property Items[Index: Integer]: TProcess read GetItem write SetItem;

function Add(Item: TProcess): Integer;

procedure Clear; override;

end;

TForm1 = class(TForm)

ImageList1: TImageList;

ToolBar2: TToolBar;

ImageList2: TImageList;

tbOpen: TToolButton;

tbSave: TToolButton;

ToolButton3: TToolButton;

ToolButton4: TToolButton;

ToolButton5: TToolButton;

ToolButton6: TToolButton;

OpenDialog1: TOpenDialog;

ToolButton7: TToolButton;

ToolButton8: TToolButton;

PopupMenu1: TPopupMenu;

N1: TMenuItem;

ToolButton13: TToolButton;

CoolBar1: TCoolBar;

ToolButton20: TToolButton;

N2: TMenuItem;

ToolButton21: TToolButton;

ToolButton22: TToolButton;

ScrollBox1: TScrollBox;

STDPanel1: TSTDPanel;

ToolBar1: TToolBar;

ImageList3: TImageList;

BitBtn1: TBitBtn;

SaveDialog1: TSaveDialog;

Button1: TButton;

procedure STDPanel1Click(Sender: TObject);

procedure ToolButtonClick(Sender: TObject);

procedure STDPanel1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemDblClick(Sender: TObject);

procedure LinkDblClick(Sender: TObject);

procedure ItemMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure LinkMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ToolButton6Click(Sender: TObject);

procedure ToolButton7Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure AcadDocument1BeginCommand(Sender: TObject;

var CommandName: OleVariant);

procedure AcadDocument1EndCommand(Sender: TObject;

var CommandName: OleVariant);

procedure ToolButton4Click(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure STDPanel1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel1MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure STDPanel2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel2MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel2MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure ToolButton20Click(Sender: TObject);

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

Shift: TShiftState);

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

procedure ToolButton21Click(Sender: TObject);

procedure tbSaveClick(Sender: TObject);

procedure ToolButton5Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

Successful: Boolean;

CommandEnded: Boolean;

CommandsCount: Integer;

public

{ Public declarations }

Item:array [1..10000] of TProcess;

links:array [1..10000] of TLink;

SaveFile: TStringList;

OpenFile: TStringList;

SelectItems: Boolean;

Drag: Boolean;

OldSelect: TPoint;

SelectedItems: TItemList;

SelectedLinks: TLinkList;

// procedure DefineLinkInACAD(Link: TLink; NewParams: TStrings; IdentLinks: TLinkList);

// procedure SelectParams(Item: TProcess);

// function GetInput(Bits: Integer; KeyWordList, Prompt: OLEVariant): WideString;

// procedure CreateProfile(Item: TProcess); //тючтЁр•рхЄ False хёыш яюы№чютрЄхы№ юЄьхэшы

end;

type

item\_type=record

left,top,width,height:integer;

hint:string[80];

pic\_path:string[80];

kol: Word;

par:array [1..100] of string[80];

end;

link\_type=record

first,second: Word;

caption:string[80];

kol: Word;

par:array [1..100] of string[80];

end;

det=record

kol\_tool: Word;

end;

const

FlipInds: array[1..2] of Char = ('a','b');

var

Form1: TForm1;

program\_dir:string;

index: Word;

Item\_index,index\_item: Word;

click\_arr:array [1..2] of Word;

mx,my:integer;

link\_index,index\_link: Word;

first\_click,second\_click:boolean;

dx,dy: Word;

destroy\_flag:boolean;

rr:TRect;

p\_rr:PRect;

detal1,detal2:word;

current\_item:word;

current\_Link: Word;

flip\_flag:Boolean;

labs:array of TLabel;

kol:byte;

t,ht,e:real;

method:integer;

desc:string;

implementation

uses WaitCommand, LinkEdit, Variables,

Progress,Unit2, Unit3, relation,model\_param\_dlg;

{$R \*.DFM}

procedure TForm1.LinkDblClick(Sender: TObject);

var st:TStringList;

t:Tlink;

s:string;

begin

st:=(sender as Tlink).Param;

t:=sender as Tlink;

linksfm.showEdit(t);

end;

procedure TForm1.ItemDblClick(Sender: TObject);

begin

if (Sender as TProcess).Hint='=хяюфтшцэр \_юяюЁр' then

begin

massafm.Edit6.Visible:=false;

massafm.Edit7.Visible:=false;

massafm.label8.Visible:=false;

massafm.label9.Visible:=false;

massafm.Edit1.Text:=(Sender as TProcess).FileName.Strings[2];

massafm.Edit2.Text:=(Sender as TProcess).FileName.Strings[3];

massafm.Edit3.Text:=(Sender as TProcess).FileName.Strings[4];

massafm.Edit4.Text:=(Sender as TProcess).FileName.Strings[5];

massafm.Edit5.Text:=(Sender as TProcess).FileName.Strings[6];

if (Sender as TProcess).FileName.Strings[7]='TRUE' then

massafm.CheckBox1.Checked:=true

else massafm.CheckBox1.Checked:=false;

massafm.Caption:=(Sender as TProcess).FileName.Strings[0];

massafm.ShowModal;

(Sender as TProcess).FileName.Clear;

(Sender as TProcess).FileName.Add(massafm.Caption);

(Sender as TProcess).FileName.Add('notmov');

(Sender as TProcess).FileName.Add(massafm.Edit1.Text);

(Sender as TProcess).FileName.Add(massafm.Edit2.Text);

(Sender as TProcess).FileName.Add(massafm.Edit3.Text);

(Sender as TProcess).FileName.Add(massafm.Edit4.Text);

(Sender as TProcess).FileName.Add(massafm.Edit5.Text);

if massafm.CheckBox1.Checked=true then

(Sender as TProcess).FileName.Add('TRUE')

else (Sender as TProcess).FileName.Add('FALSE');

end;

if (Sender as TProcess).Hint='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

massafm.Edit6.Visible:=true;

massafm.Edit7.Visible:=true;

massafm.label8.Visible:=true;

massafm.label9.Visible:=true;

massafm.Edit1.Text:=(Sender as TProcess).FileName.Strings[2];

massafm.Edit2.Text:=(Sender as TProcess).FileName.Strings[3];

massafm.Edit3.Text:=(Sender as TProcess).FileName.Strings[4];

massafm.Edit4.Text:=(Sender as TProcess).FileName.Strings[5];

massafm.Edit5.Text:=(Sender as TProcess).FileName.Strings[6];

massafm.Edit6.Text:=(Sender as TProcess).FileName.Strings[7];

massafm.Edit7.Text:=(Sender as TProcess).FileName.Strings[8];

if (Sender as TProcess).FileName.Strings[9]='TRUE' then

massafm.CheckBox1.Checked:=true

else massafm.CheckBox1.Checked:=false;

massafm.Caption:=(Sender as TProcess).FileName.Strings[0];

massafm.ShowModal;

(Sender as TProcess).FileName.Clear;

(Sender as TProcess).FileName.Add(massafm.Caption);

(Sender as TProcess).FileName.Add('notmov');

(Sender as TProcess).FileName.Add(massafm.Edit1.Text);

(Sender as TProcess).FileName.Add(massafm.Edit2.Text);

(Sender as TProcess).FileName.Add(massafm.Edit3.Text);

(Sender as TProcess).FileName.Add(massafm.Edit4.Text);

(Sender as TProcess).FileName.Add(massafm.Edit5.Text);

(Sender as TProcess).FileName.Add(massafm.Edit6.Text);

(Sender as TProcess).FileName.Add(massafm.Edit7.Text);

if massafm.CheckBox1.Checked=true then

(Sender as TProcess).FileName.Add('TRUE')

else (Sender as TProcess).FileName.Add('FALSE');

end;

Abort;

end;

procedure TForm1.LinkMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i,j, k: Integer;

begin

j := 0;

while (j < SelectedLinks.Count) and (Sender as TLink <> SelectedLinks.Items[j]) do

Inc(j);

if j = SelectedLinks.Count then

begin

if not (ssShift in Shift) then

SelectedLinks.Clear;

SelectedLinks.Add(Sender as TLink);

(Sender as TLink).Selected := True;

end

else

begin

SelectedLinks.Delete(j);

(Sender as TLink).Selected := False;

end;

if ToolBar2.Buttons[5].Down then

begin

for k := 0 to SelectedLinks.Count - 1 do

begin

i := 1;

while (i < link\_index) and (links[i]<>SelectedLinks.Items[k]) do

Inc(i);

if i = link\_index then

Exit;

links[i].Free;

for j:=i to link\_index-2 do

begin

links[j]:=links[j+1];

end;

link\_index:=link\_index-1;

end;

ToolBar2.Buttons[5].Down:=False;

SelectedLinks.Free;

SelectedLinks := TLinkList.Create;

end;

end;

procedure TForm1.ItemMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

var

i: Integer;

j: Integer;

begin

if ssLeft in Shift then

begin

rr.Left:=STDPanel1.Left+Form1.Left+7;

rr.Top:=STDPanel1.Top+Form1.Top+25;

rr.Right:=STDPanel1.Left+STDPanel1.Width+Form1.Left;

rr.Bottom:=STDPanel1.Top+STDPanel1.Height+Form1.Top+20;

p\_rr:=@rr;

ClipCursor(p\_rr);

for j := 0 to SelectedItems.Count - 1 do

begin

if (SelectedItems.Items[j].Left+x-dx>0) and

(SelectedItems.Items[j].Left+x-dx+SelectedItems.Items[j].Width<STDPanel1.Width-5) then

begin

SelectedItems.Items[j].Left:=SelectedItems.Items[j].Left+x-dx;

end;

if (SelectedItems.Items[j].Top+y-dy>0) and

(SelectedItems.Items[j].Top+y-dy+SelectedItems.Items[j].Height<STDPanel1.Height-5) then

begin

SelectedItems.Items[j].Top:=SelectedItems.Items[j].Top+y-dy;

end;

end;

for i:=1 to link\_index-1 do

for j := 0 to SelectedItems.Count - 1 do

begin

if links[i].First= SelectedItems.Items[j] then

Links[i].First:=SelectedItems.Items[j];

if links[i].Second=SelectedItems.Items[j] then

Links[i].Second:=SelectedItems.Items[j];

end;

end;

end;

procedure TForm1.ItemMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

label Again;

var

i, j, k: Integer;

l1,l2:integer;

Sel\_Item: Tprocess;

Sel\_Link: Tlink;

IdentLinks: TLinkList;

str:TStringList;

begin

rr.Left:=Screen.DesktopLeft;

rr.Top:=Screen.DesktopTop;

rr.Right:=Screen.DesktopLeft+Screen.DesktopWidth;

rr.Bottom:=Screen.DesktopTop+Screen.DesktopHeight;

p\_rr:=@rr;

ClipCursor(p\_rr);

//яЁютхЁър эр єфрыхэшх

for k := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[k].Mode := pmNormal;

//яЁютхЁър эр єёЄрэютъє ёт чш

Sel\_Item := Sender as TProcess;

if ToolBar2.Buttons[4].Down then

begin

for i:=1 to item\_index-1 do

if Sel\_Item = item[i] then

Break;

if (not first\_click) then

begin

click\_arr[1]:=i;

first\_click:=True;

exit;

end;

if i<>click\_arr[1] then click\_arr[2]:=i;

for i:=1 to 2 do

if click\_arr[i]=0 then exit;

ToolBar2.Buttons[4].Down:=False;

links[link\_index]:=TLink.Create(STDPanel1);

first\_click:=False;

links[link\_index].Parent:=STDPanel1;

links[link\_index].Caption:='';

links[link\_index].First:=item[click\_arr[1]];

links[link\_index].Second:=item[click\_arr[2]];

links[link\_index].Param:=TStringList.Create;

links[link\_index].OnMouseDown:=LinkMouseDown;

links[link\_index].OnDblClick:=LinkDblClick;

links[link\_index].NameBoxSize:=0;

links[link\_index].Param.Add('Tт ч№ '+inttostr(index\_link));

str:=links[link\_index].Param;

linksfm.showEdit(links[link\_index]);

links[link\_index].Param:=str;

click\_arr[1]:=0;

click\_arr[2]:=0;

{ if (AnsiUpperCase(links[link\_index].First.Hint) = '¦¦+LLT-') or

(AnsiUpperCase(links[link\_index].Second.Hint) = '¦¦+LLT-') then

// begin

links[link\_index].Kind := lkJoin;

{ IdentLinks := TLinkList.Create;

try

DefineLinkInACAD(links[link\_index], links[link\_index].Param, IdentLinks);

except

links[link\_index].Free;

Exit;

end;

end

else}

inc(link\_index);

inc(index\_link);

end;

end;

procedure TForm1.ItemMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

j: Integer;

begin

if ssLeft in Shift then

begin

j := 0;

while (j < SelectedItems.Count) and (Sender as TProcess <> SelectedItems.Items[j]) do

Inc(j);

if j = SelectedItems.Count then

begin

if not (ssShift in Shift) then

SelectedItems.Clear;

SelectedItems.Add(Sender as TProcess);

(Sender as TProcess).Selected := True;

end

else

begin

if (ssShift in Shift) then

begin

SelectedItems.Delete(j);

(Sender as TProcess).Selected := False;

end;

end;

for j := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[j].Mode:=pmMove;

dx:=x;

dy:=y;

end;

if ssRight in Shift then

begin

if (Sender as TProcess).FileName.Count>0 then PopupMenu1.Items.Items[0].Enabled:=True

else PopupMenu1.Items.Items[0].Enabled:=False;

if (Sender as TProcess).FileName.IndexOf('TOOL')>=0 then

PopupMenu1.Items.Items[1].Enabled:=True

else

PopupMenu1.Items.Items[1].Enabled:=False;

PopupMenu1.Items.Items[1].Checked := (Sender as TProcess).OnlyTool;

for i:=1 to item\_index-1 do

if (Sender as TProcess)=item[i] then

begin

current\_item:=i;

break;

end;

PopupMenu1.Popup((Sender as TProcess).left+STDPanel1.Left+x+Form1.Left+ScrollBox1.Left,(Sender as TProcess).top+STDPanel1.Top+y+Form1.Top+ScrollBox1.Top+40);

end;

end;

procedure TForm1.STDPanel1Click(Sender: TObject);

label Ex;

var

i: Integer;

pic:TBitmap;

Res: TModalResult;

begin

pic:=TBitmap.Create;

for i:=0 to ToolBar1.ButtonCount-1 do

begin

if ToolBar1.Buttons[i].Down then

begin

ToolBar1.Buttons[i].Down:=False;

Item[Item\_index]:=TProcess.Create(STDPanel1);

pic.LoadFromFile(ToolBar1.Buttons[i].Caption);

Item[item\_index].PicPath:=ToolBar1.Buttons[i].Caption;

Item[item\_index].Width:=pic.Width+3;

Item[item\_index].Height:=pic.Height+3;

Item[item\_index].Mode:=pmNormal;

Item[item\_index].left:=mx;

Item[item\_index].top:=my;

Item[item\_index].FileName:=TStringList.Create;

first\_click:=False;

Item[item\_index].Parent:=STDPanel1;

Item[item\_index].Hint:=ToolBar1.Buttons[i].Hint;

Item[item\_index].ShowHint:=True;

Item[item\_index].Picture.LoadFromFile(ToolBar1.Buttons[i].Caption);

Item[item\_index].OnMouseDown:=ItemMouseDown;

Item[item\_index].OnMouseUp:=ItemMouseUp;

Item[item\_index].OnMouseMove:=ItemMouseMove;

Item[item\_index].OnDblClick:=ItemDblClick;

if Item[item\_index].Hint='=хяюфтшцэр \_юяюЁр' then

begin

massafm.Edit6.Visible:=false;

massafm.Edit7.Visible:=false;

massafm.label8.Visible:=false;

massafm.label9.Visible:=false;

massafm.Edit5.Text:='=хяюфтшцэр \_юяюЁр';

massafm.Caption:='¦рёёр '+inttostr(index\_item);

massafm.ShowModal;

Item[item\_index].FileName.Add('¦рёёр '+inttostr(index\_item));

Item[item\_index].FileName.Add('=хяюфтшцэр \_юяюЁр');

Item[item\_index].FileName.Add(massafm.Edit1.Text);

Item[item\_index].FileName.Add(massafm.Edit2.Text);

Item[item\_index].FileName.Add(massafm.Edit3.Text);

Item[item\_index].FileName.Add(massafm.Edit4.Text);

Item[item\_index].FileName.Add(massafm.Edit5.Text);

if massafm.CheckBox1.Checked=true then

Item[item\_index].FileName.Add('TRUE')

else Item[item\_index].FileName.Add('FALSE');

end;

if Item[item\_index].Hint='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

massafm.Edit6.Visible:=true;

massafm.Edit7.Visible:=true;

massafm.label8.Visible:=true;

massafm.label9.Visible:=true;

massafm.Edit5.Text:='TюёЁхфюЄюўхээр \_ьрёёр';

massafm.Caption:='¦рёёр '+inttostr(index\_item);

massafm.ShowModal;

Item[item\_index].FileName.Add('¦рёёр '+inttostr(index\_item));

Item[item\_index].FileName.Add('TюёЁхфюЄюўхээр \_ьрёёр');

Item[item\_index].FileName.Add(massafm.Edit1.Text);

Item[item\_index].FileName.Add(massafm.Edit2.Text);

Item[item\_index].FileName.Add(massafm.Edit3.Text);

Item[item\_index].FileName.Add(massafm.Edit4.Text);

Item[item\_index].FileName.Add(massafm.Edit5.Text);

Item[item\_index].FileName.Add(massafm.Edit6.Text);

Item[item\_index].FileName.Add(massafm.Edit7.Text);

if massafm.CheckBox1.Checked=true then

Item[item\_index].FileName.Add('TRUE')

else Item[item\_index].FileName.Add('FALSE');

end;

{ if AnsiUpperCase(Item[item\_index].Hint) = '¦¦+LLT-' then

try

CreateProfile(Item[item\_index]);

except

on E: Exception do

MessageDlg(E.Message, mtError,[mbOK],0);

end

else

SelectParams(Item[item\_index]);

if Item[item\_index].FileName.Text = '' then

begin

Item[item\_index].Free;

Pic.Free;

Exit;

end;}

inc(item\_index);

inc(index\_item);

break;

end;

end;

Pic.Free;

end;

procedure TForm1.ToolButtonClick(Sender: TObject);

var

i: Word;

begin

for i:=0 to ToolBar2.ButtonCount-1 do ToolBar2.Buttons[i].Down:=False;

if (Sender as TToolButton).Down then

begin

index:=(Sender as TToolButton).Index;

for i:=0 to ToolBar1.ButtonCount-1 do

if i<>index then ToolBar1.Buttons[i].Down:=False;

end;

end;

procedure TForm1.STDPanel1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if Button = mbLeft then

begin

mx:=x;

my:=y;

OldSelect.x := mx;

OldSelect.y := my;

if not SelectItems then

begin

SelectItems := True;

end;

if not (ssShift in Shift) then

begin

for i := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[i].Selected := False;

for i := 0 to SelectedLinks.Count - 1 do

SelectedLinks.Items[i].Selected := False;

SelectedItems.Clear;

SelectedLinks.Clear;

end;

end;

end;

procedure TForm1.ToolButton6Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.ToolButton7Click(Sender: TObject);

var

ii: Word;

begin

if item\_index<>1 then

begin

end;

for ii:=1 to link\_index-1 do links[ii].Destroy;

link\_index:=1;

for ii:=1 to item\_index-1 do item[ii].Destroy;

item\_index:=1;

end;

procedure TForm1.FormCreate(Sender: TObject);

var

mask,pic:TBitmap;

icons,icons1:TextFile;

s,icon\_name,hint\_string:string;

tool\_button:TToolButton;

idx: Word;

ii: Word;

szAcadCaption: PChar;

begin

kol:=0;

CommandEnded := True;

Successful := True;

CommandsCount := 0;

program\_dir:=application.ExeName;

for ii:=length(program\_dir) downto 1 do

if program\_dir[ii]='\' then break;

delete(program\_dir,ii,length(program\_dir)-1);

SaveDialog1.InitialDir:=program\_dir+'\save';

OpenDialog1.InitialDir:=program\_dir+'\save';

item\_index:=1;

index\_item:=1;

link\_index:=1;

index\_link:=1;

first\_click:=false;

second\_click:=false;

destroy\_flag:=false;

mask:=TBitmap.create;

mask.Width:=46;

mask.height:=46;

mask.Canvas.brush.color:=clBlack;

mask.Canvas.pen.color:=clBlack;

mask.Canvas.FillRect(rect(0,0,45,45));

AssignFile(icons,program\_dir+'\icons.lst');

Reset(icons);

idx:=0;

ImageList1.Clear;

While not eof(icons) do

begin

readln(icons,s);

if s = '' then break;

icon\_name:=copy(s,1,pos(' ',s)-1);

Delete(s,1,pos(' ',s));

if Pos(' ', S) > 0 then

s := Copy(s,1,Pos(' ', s) - 1);

hint\_string := s;

tool\_button:=TToolButton.Create(ToolBar1);

With tool\_button do

begin

pic:=TBitmap.create;

try

pic.LoadFromFile(program\_dir+'\images\'+icon\_name+'.bmp');

except

MessageDlg('=х єфрыюё№ юЄъЁvЄ№ шчюсЁрцхэшх фхЄрыш '+Hint\_string+#13#10+

'Lрщы \IMAGES\'+icon\_name+'.bmp',mtError,[mbOK],0);

tool\_button.Free;

Continue;

end;

ImageList1.Add(pic,mask);

ImageIndex:=idx;

inc(idx);

Parent:=ToolBar1;

delete(icon\_name,1,1);

Caption:=program\_dir+'\images\'+icon\_name+'.bmp';

hint:=hint\_string;

showhint:=true;

grouped:=false;

AllowAllUp:=false;

style:=tbsCheck;

Wrap:=False;

Height:=21;

Width:=21;

onClick:=ToolButtonClick;

end;

pic.Free;

end;

CloseFile(icons);

mask.free;

ToolBar1.Images:=ImageList1;

OpenFile := TStringList.Create;

SaveFile := TStringList.Create;

SelectedItems := TItemList.Create;

SelectedLinks := TLinkList.Create;

end;

procedure TForm1.AcadDocument1BeginCommand(Sender: TObject;

var CommandName: OleVariant);

begin

CommandEnded := False;

FormCommand.Show;

Form1.Enabled := False;

end;

procedure TForm1.AcadDocument1EndCommand(Sender: TObject;

var CommandName: OleVariant);

begin

CommandEnded := True;

FormCommand.Hide;

Form1.Enabled := True;

Inc(CommandsCount);

end;

procedure TForm1.ToolButton4Click(Sender: TObject);

begin

if ToolBar2.Buttons[4].Down then

begin

first\_click := False;

click\_arr[1] := 0;

click\_arr[2] := 0;

end;

end;

function TLinkList.Add(Item: TLink): Integer;

begin

Result := inherited Add(Item);

end;

procedure TLinkList.Clear;

var

i: Integer;

begin

for i := 0 to Count - 1 do

begin

try

Items[i].Selected := False;

Items[i].Refresh;

except

end;

end;

inherited Clear;

end;

function TLinkList.GetItem(Index: Integer): TLink;

begin

Result := TLink(inherited Items[Index]);

end;

procedure TLinkList.SetItem(Index: Integer; Value: TLink);

begin

inherited Items[Index] := Value;

end;

procedure TForm1.FormDestroy(Sender: TObject);

begin

OpenFile.Free;

SaveFile.Free;

end;

procedure TForm1.STDPanel1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if SelectItems then

begin

SelectItems := False;

STDPanel1.Refresh;

for i := 1 to item\_index - 1 do

if not Item[i].Selected then

begin

Item[i].Selected := False;

with Item[i] do

if (Left > mx) and (Left+Width < OldSelect.x) and (Top > my) and (Top+Height < OldSelect.y) then

begin

SelectedItems.Add(Item[i]);

Item[i].Selected := True;

end;

end;

for i := 1 to link\_index - 1 do

if not Links[i].Selected then

begin

Links[i].Selected := False;

with Links[i] do

if (Left > mx) and (Left+Width < OldSelect.x) and (Top > my) and (Top+Height < OldSelect.y) then

begin

SelectedLinks.Add(Links[i]);

Links[i].Selected := True;

end;

end;

end;

end;

procedure TForm1.STDPanel1MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if SelectItems then

begin

STDPanel1.DrawRect(mx,my,OldSelect.x,OldSelect.y, clBlack);

STDPanel1.DrawRect(mx,my,x,y, clBlack);

OldSelect.x := x;

OldSelect.y := y;

end;

if Drag then

begin

// STDPanel1.DrawRect(mx, my, mx+STDPanel2.Width, my + STDPanel2.Height,STDPanel1.Color);

// STDPanel1.DrawRect(x, y, x+STDPanel2.Width, y + STDPanel2.Height,clBlack);

mx := x;

my := y;

end;

end;

function TItemList.Add(Item: TProcess): Integer;

begin

Result := inherited Add(Item);

end;

procedure TItemList.Clear;

var

i: Integer;

begin

for i := 0 to Count - 1 do

begin

try

Items[i].Selected := False;

Items[i].Refresh;

except

end;

end;

inherited Clear;

end;

function TItemList.GetItem(Index: Integer): TProcess;

begin

Result := TProcess(inherited Items[Index]);

end;

procedure TItemList.SetItem(Index: Integer; Value: TProcess);

begin

inherited Items[Index] := Value;

end;

procedure TForm1.STDPanel2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if (x in [0..10]) and (y in [0..10]) then

Drag := True;

end;

procedure TForm1.STDPanel2MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Drag := False;

end;

procedure TForm1.STDPanel2MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if Drag then

begin

// STDPanel2.Left := x + STDPanel2.Left;

// STDPanel2.Top := y + STDPanel2.Top;

end

end;

procedure TForm1.ToolButton20Click(Sender: TObject);

var

i: Integer;

// CopyOfItem: TProcess;

// CopyLink: TLink;

CopiedItems: TItemList;

function DoCopyItem(CopyItem: TProcess): TProcess;

begin

item[item\_index]:=TProcess.Create(STDPanel1);

item[item\_index].Parent:=STDPanel1;

item[item\_index].Id:= CopyItem.Id;

item[item\_index].Left:= CopyItem.Left + 10;

item[item\_index].Top := CopyItem.Top + 10;

item[item\_index].Height := CopyItem.Height;

item[item\_index].Width := CopyItem.Width;

item[item\_index].Hint := CopyItem.Hint;

item[item\_index].PicPath:= CopyItem.PicPath;

item[item\_index].FileName:=TStringList.Create;

item[item\_index].FileName.AddStrings(CopyItem.FileName);

item[item\_index].FileName.Strings[0]:='¦рёёр '+inttostr(index\_item);

item[item\_index].ShowHint:=True;

item[item\_index].OnDblClick:=ItemDblClick;

item[item\_index].OnMouseDown:=ItemMouseDown;

item[item\_index].OnMouseUp:=ItemMouseUp;

item[item\_index].OnMouseMove:=ItemMouseMove;

item[item\_index].Mode:=pmNormal;

CopyItem.Selected := False;

Result := item[item\_index];

Inc(item\_index);

Inc(index\_item);

end;

begin

CopiedItems := TItemList.Create;

//ъюяшЁютрэшх ¤ыхьхэЄют

i := 0;

while SelectedItems.Count > 0 do

begin

CopiedItems.Add(DoCopyItem(SelectedItems.Items[i]));

SelectedItems.Delete(0);

end;

for i := 0 to CopiedItems.Count - 1 do

begin

SelectedItems.Add(CopiedItems.Items[i]);

CopiedItems.Items[i].Selected := True;

end;

SelectedLinks.Clear;

CopiedItems.Free;

for i := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[i].Selected := True;

end;

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

Shift: TShiftState);

begin

if Key = VK\_DELETE then

ToolButton5.Click;

end;

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

var

Res: TModalResult;

begin

if Item\_index > 1 then

begin

Res := MessageDlg('TюїЁрэшЄ№ Єхъє•шщ яЁюхъЄ?',mtConfirmation,[mbYes,mbNo,mbCancel],0);

if Res = mrYes then

tbSave.Click

else

if Res = mrCancel then

CanClose := False;

end;

end;

procedure TForm1.ToolButton21Click(Sender: TObject);

begin

if ShellExecute(Handle,'open',PChar(Program\_dir+'\Help\TшёЄхьр +Єрья.htm'), '', '', 0) = 0 then

raise Exception.Create('');

end;

procedure TForm1.tbSaveClick(Sender: TObject);

var i,j,k,k1,l:integer;

f:TIniFile;

s:string;

begin

SaveDialog1.Execute;

form1.Caption:=SaveDialog1.FileName;

if SaveDialog1.FileName<>'' then

begin

f:= TIniFile.Create(SaveDialog1.FileName);

for i:=0 to 1000 do

f.EraseSection(inttostr(i));

f.WriteString('Model','Class','MDF');

f.WriteFloat('Model','T',t);

f.WriteFloat('Model','Ht',ht);

f.WriteFloat('Model','E',e);

f.WriteInteger('Model','Method',Method);

f.WriteString('Model','Description',desc);

for i:=1 to Item\_index-1 do

begin

//f.WriteString(inttostr(i-1),'Type','Massa');

if item[i].FileName.Strings[1]='TюёЁхфюЄюўхээр \_ьрёёр' then

f.WriteString(inttostr(i-1),'Class','TPointMassa')

else

f.WriteString(inttostr(i-1),'Class','TSupport');

f.WriteString(inttostr(i-1),'Name','¦рёёр '+inttostr(i));

f.WriteString(inttostr(i-1),'X0',item[i].FileName.Strings[2]);

f.WriteString(inttostr(i-1),'Y0',item[i].FileName.Strings[3]);

f.WriteString(inttostr(i-1),'W',item[i].FileName.Strings[4]);

f.WriteString(inttostr(i-1),'H',item[i].FileName.Strings[5]);

f.WriteString(inttostr(i-1),'Description',item[i].FileName.Strings[6]);

if item[i].FileName.Strings[1]='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

f.WriteString(inttostr(i-1),'M',item[i].FileName.Strings[7]);

f.WriteString(inttostr(i-1),'V0',item[i].FileName.Strings[8]);

if item[i].FileName.Strings[9]='TRUE' then

f.Writebool(inttostr(i-1),'GraphFlag',true)

else f.Writebool(inttostr(i-1),'GraphFlag',false);

end

else if item[i].FileName.Strings[7]='TRUE' then

f.Writebool(inttostr(i-1),'GraphFlag',true)

else f.Writebool(inttostr(i-1),'GraphFlag',false);

end;

k:=Item\_index-1;k1:=0;

for i:=1 to link\_index-1 do

for j:=0 to links[i].CountRelation-1 do

begin inc(k1);

f.WriteString(inttostr(k),'Type','Relation');

f.WriteString(inttostr(k),'Class',links[i].relations[j].classname);

f.WriteString(inttostr(k),'Name','Tт ч№ '+inttostr(k1));

for l:=1 to Item\_index-1 do

if links[i].First=Item[l] then

f.WriteInteger(inttostr(k),'M1',l);

for l:=1 to Item\_index-1 do

if links[i].Second=Item[l] then

f.WriteInteger(inttostr(k),'M2',l);

l:=0;

repeat

if links[i].relations[j].Param.strings[l]<>'GraphFlag' then

f.WriteString(inttostr(k),links[i].relations[j].Param.strings[l],links[i].relations[j].Param.strings[l+1])

else

if links[i].relations[j].Param.strings[l+1]='TRUE' then

f.WriteBool(inttostr(k),links[i].relations[j].Param.strings[l],true)

else f.WriteBool(inttostr(k),links[i].relations[j].Param.strings[l],false);

l:=l+2;

until links[i].relations[j].Param.Count=l;

k:=k+1;

end;

end;

end;

procedure TForm1.ToolButton5Click(Sender: TObject);

var

i, k, j: Integer;

idx, l1, l2: Integer;

Sel\_Item: Tprocess;

Sel\_Link: Tlink;

b16\_check,b2\_check: boolean;

begin

for i:=0 to ToolBar1.ButtonCount-1 do ToolBar1.Buttons[i].Down:=False;

if (Sender as TToolButton).Down then

begin

idx:=(Sender as TToolButton).Index;

for i:=0 to ToolBar2.ButtonCount-1 do

if i<>idx then ToolBar2.Buttons[i].Down:=False;

end;

if ((Sender as TToolButton).Hint = 'Lфрыхэшх') and

(SelectedItems.Count + SelectedLinks.Count > 0) then

if MessageDlg('LфрышЄ№ '+ IntToStr(SelectedItems.Count) + ' ¤ыхьхэЄ(ют) ш ' +

IntToStr(SelectedLinks.Count) + ' ёт чхщ?', mtWarning, [mbYes, mbNo], 0) = mrYes then

begin

for k := 0 to SelectedItems.Count - 1 do

begin

Sel\_Item := SelectedItems.Items[k];

l1:=1;

while l1<link\_index do

begin

if (links[l1].First=Sel\_Item) or

(links[l1].Second=Sel\_Item) then

begin

links[l1].Free;

links[l1] := nil;

for l2:=l1 to link\_index-2 do

begin

links[l2]:=links[l2+1];

// links[l2].Caption:=inttostr(l2);

end;

links[link\_index-1] := nil;

dec(link\_index);

dec(l1);

end;

inc(l1);

if l1<=0 then break;

end;

i := 1;

while (i < item\_index) and (Sel\_Item <> Item[i]) do

Inc(i);

FreeAndNil(item[i]);

destroy\_flag:=True;

for j:=i to item\_index-2 do item[j]:=item[j+1];

item\_index:=item\_index-1;

end;

for k := 0 to SelectedLinks.Count - 1 do

begin

Sel\_Link := SelectedLinks.Items[k];

i := 1;

while (i < Link\_index) and (links[i]<>Sel\_Link) do

Inc(i);

if i = Link\_Index then Continue;

try

links[i].Free;

except

end;

links[i] := nil;

for j:=i to link\_index-2 do

begin

links[j]:=links[j+1];

end;

links[link\_index-1] := nil;

link\_index:=link\_index-1;

end;

current\_item := 1;

SelectedItems.Clear;

SelectedLinks.Clear;

end;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

var form:TInputDlg;

begin

form:=TInputDlg.Create(form1);

form.ShowModal;

t:=StrToFloat(form.Edit1.Text);

ht:=StrToFloat(form.Edit2.Text);

e:=StrToFloat(form.Edit3.Text);

method:=form.RadioGroup1.ItemIndex;

desc:=form.Edit4.Text;

form.Destroy;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

//ShellExecute();

end;

end.

unit Unit1;

interface

uses

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

ImgList, ComCtrls, ToolWin, ExtCtrls, STDPanel, StdCtrls, Process, Link,

Menus, comobj,activeX,

OleServer, FileCtrl, ShellAPI, Variants, IniFiles, Buttons;

type

TLinkList = class(TList)

private

function GetItem(Index: Integer): TLink;

procedure SetItem(Index: Integer; Value: TLink);

public

property Items[Index: Integer]: TLink read GetItem write SetItem;

function Add(Item: TLink): Integer;

procedure Clear; override;

end;

TItemList = class(TList)

private

function GetItem(Index: Integer): TProcess;

procedure SetItem(Index: Integer; Value: TProcess);

public

property Items[Index: Integer]: TProcess read GetItem write SetItem;

function Add(Item: TProcess): Integer;

procedure Clear; override;

end;

TForm1 = class(TForm)

ImageList1: TImageList;

ToolBar2: TToolBar;

ImageList2: TImageList;

tbOpen: TToolButton;

tbSave: TToolButton;

ToolButton3: TToolButton;

ToolButton4: TToolButton;

ToolButton5: TToolButton;

ToolButton6: TToolButton;

OpenDialog1: TOpenDialog;

ToolButton7: TToolButton;

ToolButton8: TToolButton;

PopupMenu1: TPopupMenu;

N1: TMenuItem;

ToolButton13: TToolButton;

CoolBar1: TCoolBar;

ToolButton20: TToolButton;

N2: TMenuItem;

ToolButton21: TToolButton;

ToolButton22: TToolButton;

ScrollBox1: TScrollBox;

STDPanel1: TSTDPanel;

ToolBar1: TToolBar;

ImageList3: TImageList;

BitBtn1: TBitBtn;

SaveDialog1: TSaveDialog;

Button1: TButton;

procedure STDPanel1Click(Sender: TObject);

procedure ToolButtonClick(Sender: TObject);

procedure STDPanel1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemDblClick(Sender: TObject);

procedure LinkDblClick(Sender: TObject);

procedure ItemMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ItemMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure LinkMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure ToolButton6Click(Sender: TObject);

procedure ToolButton7Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure AcadDocument1BeginCommand(Sender: TObject;

var CommandName: OleVariant);

procedure AcadDocument1EndCommand(Sender: TObject;

var CommandName: OleVariant);

procedure ToolButton4Click(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure STDPanel1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel1MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure STDPanel2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel2MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure STDPanel2MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure ToolButton20Click(Sender: TObject);

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

Shift: TShiftState);

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

procedure ToolButton21Click(Sender: TObject);

procedure tbSaveClick(Sender: TObject);

procedure ToolButton5Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

Successful: Boolean;

CommandEnded: Boolean;

CommandsCount: Integer;

public

{ Public declarations }

Item:array [1..10000] of TProcess;

links:array [1..10000] of TLink;

SaveFile: TStringList;

OpenFile: TStringList;

SelectItems: Boolean;

Drag: Boolean;

OldSelect: TPoint;

SelectedItems: TItemList;

SelectedLinks: TLinkList;

// procedure DefineLinkInACAD(Link: TLink; NewParams: TStrings; IdentLinks: TLinkList);

// procedure SelectParams(Item: TProcess);

// function GetInput(Bits: Integer; KeyWordList, Prompt: OLEVariant): WideString;

// procedure CreateProfile(Item: TProcess); //тючтЁр•рхЄ False хёыш яюы№чютрЄхы№ юЄьхэшы

end;

type

item\_type=record

left,top,width,height:integer;

hint:string[80];

pic\_path:string[80];

kol: Word;

par:array [1..100] of string[80];

end;

link\_type=record

first,second: Word;

caption:string[80];

kol: Word;

par:array [1..100] of string[80];

end;

det=record

kol\_tool: Word;

end;

const

FlipInds: array[1..2] of Char = ('a','b');

var

Form1: TForm1;

program\_dir:string;

index: Word;

Item\_index,index\_item: Word;

click\_arr:array [1..2] of Word;

mx,my:integer;

link\_index,index\_link: Word;

first\_click,second\_click:boolean;

dx,dy: Word;

destroy\_flag:boolean;

rr:TRect;

p\_rr:PRect;

detal1,detal2:word;

current\_item:word;

current\_Link: Word;

flip\_flag:Boolean;

labs:array of TLabel;

kol:byte;

t,ht,e:real;

method:integer;

desc:string;

implementation

uses WaitCommand, LinkEdit, Variables,

Progress,Unit2, Unit3, relation,model\_param\_dlg;

{$R \*.DFM}

procedure TForm1.LinkDblClick(Sender: TObject);

var st:TStringList;

t:Tlink;

s:string;

begin

st:=(sender as Tlink).Param;

t:=sender as Tlink;

linksfm.showEdit(t);

end;

procedure TForm1.ItemDblClick(Sender: TObject);

begin

if (Sender as TProcess).Hint='=хяюфтшцэр \_юяюЁр' then

begin

massafm.Edit6.Visible:=false;

massafm.Edit7.Visible:=false;

massafm.label8.Visible:=false;

massafm.label9.Visible:=false;

massafm.Edit1.Text:=(Sender as TProcess).FileName.Strings[2];

massafm.Edit2.Text:=(Sender as TProcess).FileName.Strings[3];

massafm.Edit3.Text:=(Sender as TProcess).FileName.Strings[4];

massafm.Edit4.Text:=(Sender as TProcess).FileName.Strings[5];

massafm.Edit5.Text:=(Sender as TProcess).FileName.Strings[6];

if (Sender as TProcess).FileName.Strings[7]='TRUE' then

massafm.CheckBox1.Checked:=true

else massafm.CheckBox1.Checked:=false;

massafm.Caption:=(Sender as TProcess).FileName.Strings[0];

massafm.ShowModal;

(Sender as TProcess).FileName.Clear;

(Sender as TProcess).FileName.Add(massafm.Caption);

(Sender as TProcess).FileName.Add('notmov');

(Sender as TProcess).FileName.Add(massafm.Edit1.Text);

(Sender as TProcess).FileName.Add(massafm.Edit2.Text);

(Sender as TProcess).FileName.Add(massafm.Edit3.Text);

(Sender as TProcess).FileName.Add(massafm.Edit4.Text);

(Sender as TProcess).FileName.Add(massafm.Edit5.Text);

if massafm.CheckBox1.Checked=true then

(Sender as TProcess).FileName.Add('TRUE')

else (Sender as TProcess).FileName.Add('FALSE');

end;

if (Sender as TProcess).Hint='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

massafm.Edit6.Visible:=true;

massafm.Edit7.Visible:=true;

massafm.label8.Visible:=true;

massafm.label9.Visible:=true;

massafm.Edit1.Text:=(Sender as TProcess).FileName.Strings[2];

massafm.Edit2.Text:=(Sender as TProcess).FileName.Strings[3];

massafm.Edit3.Text:=(Sender as TProcess).FileName.Strings[4];

massafm.Edit4.Text:=(Sender as TProcess).FileName.Strings[5];

massafm.Edit5.Text:=(Sender as TProcess).FileName.Strings[6];

massafm.Edit6.Text:=(Sender as TProcess).FileName.Strings[7];

massafm.Edit7.Text:=(Sender as TProcess).FileName.Strings[8];

if (Sender as TProcess).FileName.Strings[9]='TRUE' then

massafm.CheckBox1.Checked:=true

else massafm.CheckBox1.Checked:=false;

massafm.Caption:=(Sender as TProcess).FileName.Strings[0];

massafm.ShowModal;

(Sender as TProcess).FileName.Clear;

(Sender as TProcess).FileName.Add(massafm.Caption);

(Sender as TProcess).FileName.Add('notmov');

(Sender as TProcess).FileName.Add(massafm.Edit1.Text);

(Sender as TProcess).FileName.Add(massafm.Edit2.Text);

(Sender as TProcess).FileName.Add(massafm.Edit3.Text);

(Sender as TProcess).FileName.Add(massafm.Edit4.Text);

(Sender as TProcess).FileName.Add(massafm.Edit5.Text);

(Sender as TProcess).FileName.Add(massafm.Edit6.Text);

(Sender as TProcess).FileName.Add(massafm.Edit7.Text);

if massafm.CheckBox1.Checked=true then

(Sender as TProcess).FileName.Add('TRUE')

else (Sender as TProcess).FileName.Add('FALSE');

end;

Abort;

end;

procedure TForm1.LinkMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i,j, k: Integer;

begin

j := 0;

while (j < SelectedLinks.Count) and (Sender as TLink <> SelectedLinks.Items[j]) do

Inc(j);

if j = SelectedLinks.Count then

begin

if not (ssShift in Shift) then

SelectedLinks.Clear;

SelectedLinks.Add(Sender as TLink);

(Sender as TLink).Selected := True;

end

else

begin

SelectedLinks.Delete(j);

(Sender as TLink).Selected := False;

end;

if ToolBar2.Buttons[5].Down then

begin

for k := 0 to SelectedLinks.Count - 1 do

begin

i := 1;

while (i < link\_index) and (links[i]<>SelectedLinks.Items[k]) do

Inc(i);

if i = link\_index then

Exit;

links[i].Free;

for j:=i to link\_index-2 do

begin

links[j]:=links[j+1];

end;

link\_index:=link\_index-1;

end;

ToolBar2.Buttons[5].Down:=False;

SelectedLinks.Free;

SelectedLinks := TLinkList.Create;

end;

end;

procedure TForm1.ItemMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

var

i: Integer;

j: Integer;

begin

if ssLeft in Shift then

begin

rr.Left:=STDPanel1.Left+Form1.Left+7;

rr.Top:=STDPanel1.Top+Form1.Top+25;

rr.Right:=STDPanel1.Left+STDPanel1.Width+Form1.Left;

rr.Bottom:=STDPanel1.Top+STDPanel1.Height+Form1.Top+20;

p\_rr:=@rr;

ClipCursor(p\_rr);

for j := 0 to SelectedItems.Count - 1 do

begin

if (SelectedItems.Items[j].Left+x-dx>0) and

(SelectedItems.Items[j].Left+x-dx+SelectedItems.Items[j].Width<STDPanel1.Width-5) then

begin

SelectedItems.Items[j].Left:=SelectedItems.Items[j].Left+x-dx;

end;

if (SelectedItems.Items[j].Top+y-dy>0) and

(SelectedItems.Items[j].Top+y-dy+SelectedItems.Items[j].Height<STDPanel1.Height-5) then

begin

SelectedItems.Items[j].Top:=SelectedItems.Items[j].Top+y-dy;

end;

end;

for i:=1 to link\_index-1 do

for j := 0 to SelectedItems.Count - 1 do

begin

if links[i].First= SelectedItems.Items[j] then

Links[i].First:=SelectedItems.Items[j];

if links[i].Second=SelectedItems.Items[j] then

Links[i].Second:=SelectedItems.Items[j];

end;

end;

end;

procedure TForm1.ItemMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

label Again;

var

i, j, k: Integer;

l1,l2:integer;

Sel\_Item: Tprocess;

Sel\_Link: Tlink;

IdentLinks: TLinkList;

str:TStringList;

begin

rr.Left:=Screen.DesktopLeft;

rr.Top:=Screen.DesktopTop;

rr.Right:=Screen.DesktopLeft+Screen.DesktopWidth;

rr.Bottom:=Screen.DesktopTop+Screen.DesktopHeight;

p\_rr:=@rr;

ClipCursor(p\_rr);

//яЁютхЁър эр єфрыхэшх

for k := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[k].Mode := pmNormal;

//яЁютхЁър эр єёЄрэютъє ёт чш

Sel\_Item := Sender as TProcess;

if ToolBar2.Buttons[4].Down then

begin

for i:=1 to item\_index-1 do

if Sel\_Item = item[i] then

Break;

if (not first\_click) then

begin

click\_arr[1]:=i;

first\_click:=True;

exit;

end;

if i<>click\_arr[1] then click\_arr[2]:=i;

for i:=1 to 2 do

if click\_arr[i]=0 then exit;

ToolBar2.Buttons[4].Down:=False;

links[link\_index]:=TLink.Create(STDPanel1);

first\_click:=False;

links[link\_index].Parent:=STDPanel1;

links[link\_index].Caption:='';

links[link\_index].First:=item[click\_arr[1]];

links[link\_index].Second:=item[click\_arr[2]];

links[link\_index].Param:=TStringList.Create;

links[link\_index].OnMouseDown:=LinkMouseDown;

links[link\_index].OnDblClick:=LinkDblClick;

links[link\_index].NameBoxSize:=0;

links[link\_index].Param.Add('Tт ч№ '+inttostr(index\_link));

str:=links[link\_index].Param;

linksfm.showEdit(links[link\_index]);

links[link\_index].Param:=str;

click\_arr[1]:=0;

click\_arr[2]:=0;

{ if (AnsiUpperCase(links[link\_index].First.Hint) = '¦¦+LLT-') or

(AnsiUpperCase(links[link\_index].Second.Hint) = '¦¦+LLT-') then

// begin

links[link\_index].Kind := lkJoin;

{ IdentLinks := TLinkList.Create;

try

DefineLinkInACAD(links[link\_index], links[link\_index].Param, IdentLinks);

except

links[link\_index].Free;

Exit;

end;

end

else}

inc(link\_index);

inc(index\_link);

end;

end;

procedure TForm1.ItemMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

j: Integer;

begin

if ssLeft in Shift then

begin

j := 0;

while (j < SelectedItems.Count) and (Sender as TProcess <> SelectedItems.Items[j]) do

Inc(j);

if j = SelectedItems.Count then

begin

if not (ssShift in Shift) then

SelectedItems.Clear;

SelectedItems.Add(Sender as TProcess);

(Sender as TProcess).Selected := True;

end

else

begin

if (ssShift in Shift) then

begin

SelectedItems.Delete(j);

(Sender as TProcess).Selected := False;

end;

end;

for j := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[j].Mode:=pmMove;

dx:=x;

dy:=y;

end;

if ssRight in Shift then

begin

if (Sender as TProcess).FileName.Count>0 then PopupMenu1.Items.Items[0].Enabled:=True

else PopupMenu1.Items.Items[0].Enabled:=False;

if (Sender as TProcess).FileName.IndexOf('TOOL')>=0 then

PopupMenu1.Items.Items[1].Enabled:=True

else

PopupMenu1.Items.Items[1].Enabled:=False;

PopupMenu1.Items.Items[1].Checked := (Sender as TProcess).OnlyTool;

for i:=1 to item\_index-1 do

if (Sender as TProcess)=item[i] then

begin

current\_item:=i;

break;

end;

PopupMenu1.Popup((Sender as TProcess).left+STDPanel1.Left+x+Form1.Left+ScrollBox1.Left,(Sender as TProcess).top+STDPanel1.Top+y+Form1.Top+ScrollBox1.Top+40);

end;

end;

procedure TForm1.STDPanel1Click(Sender: TObject);

label Ex;

var

i: Integer;

pic:TBitmap;

Res: TModalResult;

begin

pic:=TBitmap.Create;

for i:=0 to ToolBar1.ButtonCount-1 do

begin

if ToolBar1.Buttons[i].Down then

begin

ToolBar1.Buttons[i].Down:=False;

Item[Item\_index]:=TProcess.Create(STDPanel1);

pic.LoadFromFile(ToolBar1.Buttons[i].Caption);

Item[item\_index].PicPath:=ToolBar1.Buttons[i].Caption;

Item[item\_index].Width:=pic.Width+3;

Item[item\_index].Height:=pic.Height+3;

Item[item\_index].Mode:=pmNormal;

Item[item\_index].left:=mx;

Item[item\_index].top:=my;

Item[item\_index].FileName:=TStringList.Create;

first\_click:=False;

Item[item\_index].Parent:=STDPanel1;

Item[item\_index].Hint:=ToolBar1.Buttons[i].Hint;

Item[item\_index].ShowHint:=True;

Item[item\_index].Picture.LoadFromFile(ToolBar1.Buttons[i].Caption);

Item[item\_index].OnMouseDown:=ItemMouseDown;

Item[item\_index].OnMouseUp:=ItemMouseUp;

Item[item\_index].OnMouseMove:=ItemMouseMove;

Item[item\_index].OnDblClick:=ItemDblClick;

if Item[item\_index].Hint='=хяюфтшцэр \_юяюЁр' then

begin

massafm.Edit6.Visible:=false;

massafm.Edit7.Visible:=false;

massafm.label8.Visible:=false;

massafm.label9.Visible:=false;

massafm.Edit5.Text:='=хяюфтшцэр \_юяюЁр';

massafm.Caption:='¦рёёр '+inttostr(index\_item);

massafm.ShowModal;

Item[item\_index].FileName.Add('¦рёёр '+inttostr(index\_item));

Item[item\_index].FileName.Add('=хяюфтшцэр \_юяюЁр');

Item[item\_index].FileName.Add(massafm.Edit1.Text);

Item[item\_index].FileName.Add(massafm.Edit2.Text);

Item[item\_index].FileName.Add(massafm.Edit3.Text);

Item[item\_index].FileName.Add(massafm.Edit4.Text);

Item[item\_index].FileName.Add(massafm.Edit5.Text);

if massafm.CheckBox1.Checked=true then

Item[item\_index].FileName.Add('TRUE')

else Item[item\_index].FileName.Add('FALSE');

end;

if Item[item\_index].Hint='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

massafm.Edit6.Visible:=true;

massafm.Edit7.Visible:=true;

massafm.label8.Visible:=true;

massafm.label9.Visible:=true;

massafm.Edit5.Text:='TюёЁхфюЄюўхээр \_ьрёёр';

massafm.Caption:='¦рёёр '+inttostr(index\_item);

massafm.ShowModal;

Item[item\_index].FileName.Add('¦рёёр '+inttostr(index\_item));

Item[item\_index].FileName.Add('TюёЁхфюЄюўхээр \_ьрёёр');

Item[item\_index].FileName.Add(massafm.Edit1.Text);

Item[item\_index].FileName.Add(massafm.Edit2.Text);

Item[item\_index].FileName.Add(massafm.Edit3.Text);

Item[item\_index].FileName.Add(massafm.Edit4.Text);

Item[item\_index].FileName.Add(massafm.Edit5.Text);

Item[item\_index].FileName.Add(massafm.Edit6.Text);

Item[item\_index].FileName.Add(massafm.Edit7.Text);

if massafm.CheckBox1.Checked=true then

Item[item\_index].FileName.Add('TRUE')

else Item[item\_index].FileName.Add('FALSE');

end;

{ if AnsiUpperCase(Item[item\_index].Hint) = '¦¦+LLT-' then

try

CreateProfile(Item[item\_index]);

except

on E: Exception do

MessageDlg(E.Message, mtError,[mbOK],0);

end

else

SelectParams(Item[item\_index]);

if Item[item\_index].FileName.Text = '' then

begin

Item[item\_index].Free;

Pic.Free;

Exit;

end;}

inc(item\_index);

inc(index\_item);

break;

end;

end;

Pic.Free;

end;

procedure TForm1.ToolButtonClick(Sender: TObject);

var

i: Word;

begin

for i:=0 to ToolBar2.ButtonCount-1 do ToolBar2.Buttons[i].Down:=False;

if (Sender as TToolButton).Down then

begin

index:=(Sender as TToolButton).Index;

for i:=0 to ToolBar1.ButtonCount-1 do

if i<>index then ToolBar1.Buttons[i].Down:=False;

end;

end;

procedure TForm1.STDPanel1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if Button = mbLeft then

begin

mx:=x;

my:=y;

OldSelect.x := mx;

OldSelect.y := my;

if not SelectItems then

begin

SelectItems := True;

end;

if not (ssShift in Shift) then

begin

for i := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[i].Selected := False;

for i := 0 to SelectedLinks.Count - 1 do

SelectedLinks.Items[i].Selected := False;

SelectedItems.Clear;

SelectedLinks.Clear;

end;

end;

end;

procedure TForm1.ToolButton6Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.ToolButton7Click(Sender: TObject);

var

ii: Word;

begin

if item\_index<>1 then

begin

end;

for ii:=1 to link\_index-1 do links[ii].Destroy;

link\_index:=1;

for ii:=1 to item\_index-1 do item[ii].Destroy;

item\_index:=1;

end;

procedure TForm1.FormCreate(Sender: TObject);

var

mask,pic:TBitmap;

icons,icons1:TextFile;

s,icon\_name,hint\_string:string;

tool\_button:TToolButton;

idx: Word;

ii: Word;

szAcadCaption: PChar;

begin

kol:=0;

CommandEnded := True;

Successful := True;

CommandsCount := 0;

program\_dir:=application.ExeName;

for ii:=length(program\_dir) downto 1 do

if program\_dir[ii]='\' then break;

delete(program\_dir,ii,length(program\_dir)-1);

SaveDialog1.InitialDir:=program\_dir+'\save';

OpenDialog1.InitialDir:=program\_dir+'\save';

item\_index:=1;

index\_item:=1;

link\_index:=1;

index\_link:=1;

first\_click:=false;

second\_click:=false;

destroy\_flag:=false;

mask:=TBitmap.create;

mask.Width:=46;

mask.height:=46;

mask.Canvas.brush.color:=clBlack;

mask.Canvas.pen.color:=clBlack;

mask.Canvas.FillRect(rect(0,0,45,45));

AssignFile(icons,program\_dir+'\icons.lst');

Reset(icons);

idx:=0;

ImageList1.Clear;

While not eof(icons) do

begin

readln(icons,s);

if s = '' then break;

icon\_name:=copy(s,1,pos(' ',s)-1);

Delete(s,1,pos(' ',s));

if Pos(' ', S) > 0 then

s := Copy(s,1,Pos(' ', s) - 1);

hint\_string := s;

tool\_button:=TToolButton.Create(ToolBar1);

With tool\_button do

begin

pic:=TBitmap.create;

try

pic.LoadFromFile(program\_dir+'\images\'+icon\_name+'.bmp');

except

MessageDlg('=х єфрыюё№ юЄъЁvЄ№ шчюсЁрцхэшх фхЄрыш '+Hint\_string+#13#10+

'Lрщы \IMAGES\'+icon\_name+'.bmp',mtError,[mbOK],0);

tool\_button.Free;

Continue;

end;

ImageList1.Add(pic,mask);

ImageIndex:=idx;

inc(idx);

Parent:=ToolBar1;

delete(icon\_name,1,1);

Caption:=program\_dir+'\images\'+icon\_name+'.bmp';

hint:=hint\_string;

showhint:=true;

grouped:=false;

AllowAllUp:=false;

style:=tbsCheck;

Wrap:=False;

Height:=21;

Width:=21;

onClick:=ToolButtonClick;

end;

pic.Free;

end;

CloseFile(icons);

mask.free;

ToolBar1.Images:=ImageList1;

OpenFile := TStringList.Create;

SaveFile := TStringList.Create;

SelectedItems := TItemList.Create;

SelectedLinks := TLinkList.Create;

end;

procedure TForm1.AcadDocument1BeginCommand(Sender: TObject;

var CommandName: OleVariant);

begin

CommandEnded := False;

FormCommand.Show;

Form1.Enabled := False;

end;

procedure TForm1.AcadDocument1EndCommand(Sender: TObject;

var CommandName: OleVariant);

begin

CommandEnded := True;

FormCommand.Hide;

Form1.Enabled := True;

Inc(CommandsCount);

end;

procedure TForm1.ToolButton4Click(Sender: TObject);

begin

if ToolBar2.Buttons[4].Down then

begin

first\_click := False;

click\_arr[1] := 0;

click\_arr[2] := 0;

end;

end;

function TLinkList.Add(Item: TLink): Integer;

begin

Result := inherited Add(Item);

end;

procedure TLinkList.Clear;

var

i: Integer;

begin

for i := 0 to Count - 1 do

begin

try

Items[i].Selected := False;

Items[i].Refresh;

except

end;

end;

inherited Clear;

end;

function TLinkList.GetItem(Index: Integer): TLink;

begin

Result := TLink(inherited Items[Index]);

end;

procedure TLinkList.SetItem(Index: Integer; Value: TLink);

begin

inherited Items[Index] := Value;

end;

procedure TForm1.FormDestroy(Sender: TObject);

begin

OpenFile.Free;

SaveFile.Free;

end;

procedure TForm1.STDPanel1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if SelectItems then

begin

SelectItems := False;

STDPanel1.Refresh;

for i := 1 to item\_index - 1 do

if not Item[i].Selected then

begin

Item[i].Selected := False;

with Item[i] do

if (Left > mx) and (Left+Width < OldSelect.x) and (Top > my) and (Top+Height < OldSelect.y) then

begin

SelectedItems.Add(Item[i]);

Item[i].Selected := True;

end;

end;

for i := 1 to link\_index - 1 do

if not Links[i].Selected then

begin

Links[i].Selected := False;

with Links[i] do

if (Left > mx) and (Left+Width < OldSelect.x) and (Top > my) and (Top+Height < OldSelect.y) then

begin

SelectedLinks.Add(Links[i]);

Links[i].Selected := True;

end;

end;

end;

end;

procedure TForm1.STDPanel1MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if SelectItems then

begin

STDPanel1.DrawRect(mx,my,OldSelect.x,OldSelect.y, clBlack);

STDPanel1.DrawRect(mx,my,x,y, clBlack);

OldSelect.x := x;

OldSelect.y := y;

end;

if Drag then

begin

// STDPanel1.DrawRect(mx, my, mx+STDPanel2.Width, my + STDPanel2.Height,STDPanel1.Color);

// STDPanel1.DrawRect(x, y, x+STDPanel2.Width, y + STDPanel2.Height,clBlack);

mx := x;

my := y;

end;

end;

function TItemList.Add(Item: TProcess): Integer;

begin

Result := inherited Add(Item);

end;

procedure TItemList.Clear;

var

i: Integer;

begin

for i := 0 to Count - 1 do

begin

try

Items[i].Selected := False;

Items[i].Refresh;

except

end;

end;

inherited Clear;

end;

function TItemList.GetItem(Index: Integer): TProcess;

begin

Result := TProcess(inherited Items[Index]);

end;

procedure TItemList.SetItem(Index: Integer; Value: TProcess);

begin

inherited Items[Index] := Value;

end;

procedure TForm1.STDPanel2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if (x in [0..10]) and (y in [0..10]) then

Drag := True;

end;

procedure TForm1.STDPanel2MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Drag := False;

end;

procedure TForm1.STDPanel2MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if Drag then

begin

// STDPanel2.Left := x + STDPanel2.Left;

// STDPanel2.Top := y + STDPanel2.Top;

end

end;

procedure TForm1.ToolButton20Click(Sender: TObject);

var

i: Integer;

// CopyOfItem: TProcess;

// CopyLink: TLink;

CopiedItems: TItemList;

function DoCopyItem(CopyItem: TProcess): TProcess;

begin

item[item\_index]:=TProcess.Create(STDPanel1);

item[item\_index].Parent:=STDPanel1;

item[item\_index].Id:= CopyItem.Id;

item[item\_index].Left:= CopyItem.Left + 10;

item[item\_index].Top := CopyItem.Top + 10;

item[item\_index].Height := CopyItem.Height;

item[item\_index].Width := CopyItem.Width;

item[item\_index].Hint := CopyItem.Hint;

item[item\_index].PicPath:= CopyItem.PicPath;

item[item\_index].FileName:=TStringList.Create;

item[item\_index].FileName.AddStrings(CopyItem.FileName);

item[item\_index].FileName.Strings[0]:='¦рёёр '+inttostr(index\_item);

item[item\_index].ShowHint:=True;

item[item\_index].OnDblClick:=ItemDblClick;

item[item\_index].OnMouseDown:=ItemMouseDown;

item[item\_index].OnMouseUp:=ItemMouseUp;

item[item\_index].OnMouseMove:=ItemMouseMove;

item[item\_index].Mode:=pmNormal;

CopyItem.Selected := False;

Result := item[item\_index];

Inc(item\_index);

Inc(index\_item);

end;

begin

CopiedItems := TItemList.Create;

//ъюяшЁютрэшх ¤ыхьхэЄют

i := 0;

while SelectedItems.Count > 0 do

begin

CopiedItems.Add(DoCopyItem(SelectedItems.Items[i]));

SelectedItems.Delete(0);

end;

for i := 0 to CopiedItems.Count - 1 do

begin

SelectedItems.Add(CopiedItems.Items[i]);

CopiedItems.Items[i].Selected := True;

end;

SelectedLinks.Clear;

CopiedItems.Free;

for i := 0 to SelectedItems.Count - 1 do

SelectedItems.Items[i].Selected := True;

end;

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

Shift: TShiftState);

begin

if Key = VK\_DELETE then

ToolButton5.Click;

end;

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

var

Res: TModalResult;

begin

if Item\_index > 1 then

begin

Res := MessageDlg('TюїЁрэшЄ№ Єхъє•шщ яЁюхъЄ?',mtConfirmation,[mbYes,mbNo,mbCancel],0);

if Res = mrYes then

tbSave.Click

else

if Res = mrCancel then

CanClose := False;

end;

end;

procedure TForm1.ToolButton21Click(Sender: TObject);

begin

if ShellExecute(Handle,'open',PChar(Program\_dir+'\Help\TшёЄхьр +Єрья.htm'), '', '', 0) = 0 then

raise Exception.Create('');

end;

procedure TForm1.tbSaveClick(Sender: TObject);

var i,j,k,k1,l:integer;

f:TIniFile;

s:string;

begin

SaveDialog1.Execute;

form1.Caption:=SaveDialog1.FileName;

if SaveDialog1.FileName<>'' then

begin

f:= TIniFile.Create(SaveDialog1.FileName);

for i:=0 to 1000 do

f.EraseSection(inttostr(i));

f.WriteString('Model','Class','MDF');

f.WriteFloat('Model','T',t);

f.WriteFloat('Model','Ht',ht);

f.WriteFloat('Model','E',e);

f.WriteInteger('Model','Method',Method);

f.WriteString('Model','Description',desc);

for i:=1 to Item\_index-1 do

begin

//f.WriteString(inttostr(i-1),'Type','Massa');

if item[i].FileName.Strings[1]='TюёЁхфюЄюўхээр \_ьрёёр' then

f.WriteString(inttostr(i-1),'Class','TPointMassa')

else

f.WriteString(inttostr(i-1),'Class','TSupport');

f.WriteString(inttostr(i-1),'Name','¦рёёр '+inttostr(i));

f.WriteString(inttostr(i-1),'X0',item[i].FileName.Strings[2]);

f.WriteString(inttostr(i-1),'Y0',item[i].FileName.Strings[3]);

f.WriteString(inttostr(i-1),'W',item[i].FileName.Strings[4]);

f.WriteString(inttostr(i-1),'H',item[i].FileName.Strings[5]);

f.WriteString(inttostr(i-1),'Description',item[i].FileName.Strings[6]);

if item[i].FileName.Strings[1]='TюёЁхфюЄюўхээр \_ьрёёр' then

begin

f.WriteString(inttostr(i-1),'M',item[i].FileName.Strings[7]);

f.WriteString(inttostr(i-1),'V0',item[i].FileName.Strings[8]);

if item[i].FileName.Strings[9]='TRUE' then

f.Writebool(inttostr(i-1),'GraphFlag',true)

else f.Writebool(inttostr(i-1),'GraphFlag',false);

end

else if item[i].FileName.Strings[7]='TRUE' then

f.Writebool(inttostr(i-1),'GraphFlag',true)

else f.Writebool(inttostr(i-1),'GraphFlag',false);

end;

k:=Item\_index-1;k1:=0;

for i:=1 to link\_index-1 do

for j:=0 to links[i].CountRelation-1 do

begin inc(k1);

f.WriteString(inttostr(k),'Type','Relation');

f.WriteString(inttostr(k),'Class',links[i].relations[j].classname);

f.WriteString(inttostr(k),'Name','Tт ч№ '+inttostr(k1));

for l:=1 to Item\_index-1 do

if links[i].First=Item[l] then

f.WriteInteger(inttostr(k),'M1',l);

for l:=1 to Item\_index-1 do

if links[i].Second=Item[l] then

f.WriteInteger(inttostr(k),'M2',l);

l:=0;

repeat

if links[i].relations[j].Param.strings[l]<>'GraphFlag' then

f.WriteString(inttostr(k),links[i].relations[j].Param.strings[l],links[i].relations[j].Param.strings[l+1])

else

if links[i].relations[j].Param.strings[l+1]='TRUE' then

f.WriteBool(inttostr(k),links[i].relations[j].Param.strings[l],true)

else f.WriteBool(inttostr(k),links[i].relations[j].Param.strings[l],false);

l:=l+2;

until links[i].relations[j].Param.Count=l;

k:=k+1;

end;

end;

end;

procedure TForm1.ToolButton5Click(Sender: TObject);

var

i, k, j: Integer;

idx, l1, l2: Integer;

Sel\_Item: Tprocess;

Sel\_Link: Tlink;

b16\_check,b2\_check: boolean;

begin

for i:=0 to ToolBar1.ButtonCount-1 do ToolBar1.Buttons[i].Down:=False;

if (Sender as TToolButton).Down then

begin

idx:=(Sender as TToolButton).Index;

for i:=0 to ToolBar2.ButtonCount-1 do

if i<>idx then ToolBar2.Buttons[i].Down:=False;

end;

if ((Sender as TToolButton).Hint = 'Lфрыхэшх') and

(SelectedItems.Count + SelectedLinks.Count > 0) then

if MessageDlg('LфрышЄ№ '+ IntToStr(SelectedItems.Count) + ' ¤ыхьхэЄ(ют) ш ' +

IntToStr(SelectedLinks.Count) + ' ёт чхщ?', mtWarning, [mbYes, mbNo], 0) = mrYes then

begin

for k := 0 to SelectedItems.Count - 1 do

begin

Sel\_Item := SelectedItems.Items[k];

l1:=1;

while l1<link\_index do

begin

if (links[l1].First=Sel\_Item) or

(links[l1].Second=Sel\_Item) then

begin

links[l1].Free;

links[l1] := nil;

for l2:=l1 to link\_index-2 do

begin

links[l2]:=links[l2+1];

// links[l2].Caption:=inttostr(l2);

end;

links[link\_index-1] := nil;

dec(link\_index);

dec(l1);

end;

inc(l1);

if l1<=0 then break;

end;

i := 1;

while (i < item\_index) and (Sel\_Item <> Item[i]) do

Inc(i);

FreeAndNil(item[i]);

destroy\_flag:=True;

for j:=i to item\_index-2 do item[j]:=item[j+1];

item\_index:=item\_index-1;

end;

for k := 0 to SelectedLinks.Count - 1 do

begin

Sel\_Link := SelectedLinks.Items[k];

i := 1;

while (i < Link\_index) and (links[i]<>Sel\_Link) do

Inc(i);

if i = Link\_Index then Continue;

try

links[i].Free;

except

end;

links[i] := nil;

for j:=i to link\_index-2 do

begin

links[j]:=links[j+1];

end;

links[link\_index-1] := nil;

link\_index:=link\_index-1;

end;

current\_item := 1;

SelectedItems.Clear;

SelectedLinks.Clear;

end;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

var form:TInputDlg;

begin

form:=TInputDlg.Create(form1);

form.ShowModal;

t:=StrToFloat(form.Edit1.Text);

ht:=StrToFloat(form.Edit2.Text);

e:=StrToFloat(form.Edit3.Text);

method:=form.RadioGroup1.ItemIndex;

desc:=form.Edit4.Text;

form.Destroy;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

//ShellExecute();

end;

end.

**ПРИЛОЖЕНИЕ В**

Главный модуль прграммы расчётов.

unit modell;

interface

uses classes, stdctrls, graphics, extctrls, QUICKRPT, QRCTRLS, qrprntr,

ShellAPI, Windows, ComObj, inifiles,

types, model\_edit\_dlg,

model\_param\_dlg, visual\_dlg, report\_dlg, waitreportcreate\_dlg;

type

TCount=record

Masses, Relations: integer;

end;

TMethod=record

code : TFloat;

digree: integer;

end;

TModel = class(TAbstractObject)

public

Count: TCount;

flIsSaved : boolean; // flag of is model saved

flIsModelling: boolean;

constructor Create;

destructor Destroy; override;

procedure AddElement; virtual;

function ShowDialog(wind: integer): integer;

procedure SaveModel(FileName: string); virtual;

function LoadModel(FileName: string):boolean; virtual;

procedure SetParamsInZero; virtual;

procedure Visio; virtual;

procedure ShowReport(FName: string); virtual;

procedure MakeReport(FName: string); virtual;

protected

t: TFloat; //time

ht: TFloat; // time step

e: TFloat; // inaccuracy

CalcMethod: TMethod; // digital modelling method

WDoc: variant; //for winword report automation;

NewElement: TObject; //current created element

flReadFromFile: boolean; // flag - true if now data read from file

Editor: TEditDlg;

VisualDlg: TVisualDlg;

ReportDlg: TReportDlg;

WaitReportCreateDlg: TWaitReportCreateDlg;

procedure InitInputParamDlg; override;

procedure InitParameters; override;

procedure DeleteElement; virtual;

function EditElementParams: integer; virtual;

procedure ClearData; virtual;

procedure SaveToFile(Stream: TStream);override;

procedure SaveToIniFile(iniF: TIniFile; Section: string);override;

procedure LoadFromFile(Stream: TStream); override;

procedure LoadFromIniFile(IniF: TIniFile; Section: string); override;

procedure EditorOnHide(Sender: TObject); virtual;

procedure ListBox1Click(Sender: TObject); virtual;

procedure ListBox1DblClick(Sender: TObject); virtual;

procedure SpeedButton1Click(Sender: TObject); virtual;

procedure SpeedButton2Click(Sender: TObject); virtual;

procedure OnChangeModelName(Sender: TObject); virtual;

procedure OnClickOkButtonOnParamDlg(Sender: TObject); virtual;

procedure OnClickOkButtonOnEditorDlg(Sender: TObject); virtual;

procedure OnClickToolButton(Sender: TObject); virtual;

procedure CheckState;

function ElementPos(elType: boolean; var elPos: integer):integer; virtual;

function CheckModel: boolean;

procedure Calculate(Sender: TObject); virtual;

procedure ShowGraphs(Sender: TObject); virtual;

procedure HideGraphs(Sender: TObject); virtual;

procedure ModelInformation(Sender: TObject); virtual;

function GetModelInformation:string; virtual;

function MakeReportDlg(where: integer): integer;

function CreateReport(FName: string): boolean;

procedure OnClickSeriesCheckBox(Sender: TObject); virtual;

procedure OnClickSaveButtonOnReportDlg(Sender: TObject); virtual;

procedure WaitReportCreateDlgOnShow(Sender: TObject); virtual;

private

TempFName: string;

end;

var

Model : TModel;

implementation

uses SysUtils, Dialogs, Controls, Comctrls, clipbrd, Chart, TeEngine, Series,

Imglist,

main, massa, relation;

constructor TModel.Create;

var

i: integer;

ResType: TResType;

flType: boolean;

s: shortstring;

begin

inherited Create;

Count.Masses:=0;

Count.Relations:=0;

flIsSaved:=true;

VisualDlg:=TVisualDlg.CreateParented(0);

VisualDlg.Timer1.OnTimer:=Calculate;

VisualDlg.ToolButton3.OnClick:=ShowGraphs;

VisualDlg.ToolButton4.OnClick:=ModelInformation;

VisualDlg.OnHide:=HideGraphs;

with Editor do

begin

OnHide:=EditorOnHide;

ListBox1.OnClick:=ListBox1Click;

ListBox1.OnDblClick:=ListBox1DblClick;

SpeedButton1.OnClick:=SpeedButton1Click;

SpeedButton2.OnClick:=SpeedButton2Click;

Edit1.OnChange:=OnChangeModelName;

if ClassList.Count>0 then

for i:=0 to ClassList.Count-1 do begin

if StrUpper(PChar(ExtractFileExt(TToolButton(ClassList.Items[i]).Caption)))='.BMP'

then ResType:=rtBitmap

else ResType:=rtIcon;

if not ImageList1.FileLoad(ResType, TToolButton(ClassList.Items[i]).Caption, clBtnFace)

then ImageList1.Add(nil,nil);

TToolButton(ClassList.Items[i]).ImageIndex:=i;

TToolButton(ClassList.Items[i]).OnClick:=OnClickToolButton;

flType:=TToolButton(ClassList.Items[i]).Wrap;

TToolButton(ClassList.Items[i]).Wrap:=false;

if flType then

ToolBar1.InsertControl(TToolButton(ClassList.Items[i]))

else

ToolBar2.InsertControl(TToolButton(ClassList.Items[i]));

end;

ToolBar2.Enabled:=false;

end;

WaitReportCreateDlg:=TWaitReportCreateDlg.CreateParented(0);

WaitReportCreateDlg.OnActivate:=WaitReportCreateDlgOnShow;

NewElement:=nil;

flReadFromFile:=false;

flIsModelling:=false;

i:=GetEnvironmentVariable('TEMP',@s[1],255);

SetLength(s,i);

TempFName:=s+'\$$$\_mdl\_$$$.';

end;

destructor TModel.Destroy;

begin

inherited Destroy;

ClearData;

Editor.Destroy;

VisualDlg.Destroy;

ClassList.Destroy;

WaitReportCreateDlg.Destroy;

end;

procedure TModel.InitInputParamDlg;

begin

TInputDlg(InputParamDlg):=TInputDlg.CreateParented(0);

TInputDlg(InputParamDlg).OKBtn.OnClick:=OnClickOkButtonOnParamDlg;

TInputDlg(InputParamDlg).OnHide:=OnHideParamDlg;

Editor:=TEditDlg.CreateParented(0);

Editor.OKBtn.OnClick:=OnClickOkButtonOnEditorDlg;

end;

procedure TModel.InitParameters;

begin

t:=10;

Params.Add(TParam.Create(@t, varDouble, 'T', 0.001, 1e20, TInputDlg(InputParamDlg).Edit1)); //t

ht:=0.01;

Params.Add(TParam.Create(@ht, varDouble, 'Ht', 1e-10, 10, TInputDlg(InputParamDlg).Edit2)); //ht

e:=1;

Params.Add(TParam.Create(@e, varDouble, 'E', 0.00001, 1, TInputDlg(InputParamDlg).Edit3)); //e

CalcMethod.code:=0;

Params.Add(TParam.Create(@CalcMethod.code, varDouble,'Method', 0, 3, nil)); //CalcMethod

ObjName:='Колебательная система';

Params.Add(TParam.Create(@ObjName, varString, 'Description', '', '', Editor.Edit1)); //string description

end;

procedure TModel.OnClickOkButtonOnEditorDlg(Sender: TObject);

begin

Model.GetParams;

end;

procedure TModel.OnClickOkButtonOnParamDlg(Sender: TObject);

begin

inherited OnClickOkButtonOnParamDlg(Sender);

if Editor.ListBox1.Items.Count > 0 then flIsSaved:=false;

CalcMethod.code:=TInputDlg(InputParamDlg).RadioGroup1.ItemIndex;

end;

procedure TModel.ClearData;

var i: integer;

begin

if Editor.ListBox1.Items.Count > 0 then

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).Destroy;

flIsSaved:=true;

Count.Masses:=0;

Count.Relations:=0;

with Editor do ListBox1.Clear;

flIsModelling:=false;

end;

function TModel.ShowDialog(wind: integer) :integer;

begin

case wind of

0: Result:=Editor.ShowModal;

1: Result:=InputParamDlg.ShowModal;

2: Result:=MakeReportDlg(1);

end;

end;

procedure TModel.EditorOnHide(Sender: TObject);

begin

if not flIssaved then

flIsSaved:= (Editor.ListBox1.Items.Count=0);

end;

procedure TModel.OnChangeModelName(Sender: TObject);

begin

flIsSaved:= false;

end;

procedure TModel.ListBox1Click(Sender: TObject);

begin

with Editor do

begin

Memo1.Lines.Clear;

Memo1.Lines.Append(TAbstractElement(ListBox1.Items.Objects[ListBox1.ItemIndex]).AboutStr);

end;

end;

procedure TModel.ListBox1DblClick(Sender: TObject);

begin

SpeedButton2Click(Sender);

end;

procedure TModel.SpeedButton1Click(Sender: TObject);

var

index: integer;

s:string;

begin

with Editor do

if ListBox1.ItemIndex>=0 then

begin

if MessageDlg('Удалить элемент'#10+ListBox1.Items.Strings[ListBox1.ItemIndex],

mtConfirmation, [mbYes, mbNo], 0) <> mrYes then exit;

index:=ListBox1.ItemIndex;

s:=ListBox1.Items.Strings[index];

DeleteElement;

ListBox1.Items.Delete(ListBox1.ItemIndex);

Memo1.Lines.Clear;

CheckState;

end

else ListBox1.SetFocus;

end;

procedure TModel.SpeedButton2Click(Sender: TObject);

begin

with Editor do

if ListBox1.ItemIndex>=0 then begin

if EditElementParams =1 then begin

ListBox1Click(Sender);

flIsSaved:=false;

end;

end

else ListBox1.SetFocus;

end;

procedure TModel.OnClickToolButton(Sender: TObject);

begin

try

NewElement:=TNewObject(TToolButton(Sender).Tag);

if not flReadFromFile then AddElement;

except raise ERangeError.Create('Критическая ошибка. Недостаточно памяти');

end;

end;

procedure TModel.CheckState;

begin

with Editor do begin

ToolBar2.Enabled:= Count.Masses>1;

SpeedButton1.Enabled:= ListBox1.Items.Count <>0;

SpeedButton2.Enabled:= SpeedButton1.Enabled;

end;

end;

function TModel.ElementPos(elType: boolean; var elPos: integer):integer;

var

i,n1,n2: integer;

begin

elPos:=0;

Result:=1;

if Editor.ListBox1.Items.Count=0 then exit;

n1:=0;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType = elType then

begin

n2:=StrToInt(Copy(Editor.ListBox1.Items.Strings[i],Pos(' ',Editor.ListBox1.Items.Strings[i]),

length(Editor.ListBox1.Items.Strings[i])));

if (n2-n1)>1 then

begin

break;

end

else n1:=n2;

end;

elPos:=i;

Result:=n1+1;

end;

procedure TModel.AddElement;

var

ps,i: integer;

begin

i:=ElementPos(TAbstractElement(NewElement).flType,ps);

if TAbstractElement(NewElement).flType then

TAbstractElement(NewElement).InputParamDlg.Caption:='Масса '+IntToStr(i)

else

TAbstractElement(NewElement).InputParamDlg.Caption:='Связь '+IntToStr(i);

if (TAbstractElement(NewElement).InputParamDlg.ShowModal=mrOk) then begin

TAbstractElement(NewElement).ObjName:=TAbstractElement(NewElement).InputParamDlg.Caption;

TAbstractElement(NewElement).GraphDlg.Caption:=TAbstractElement(NewElement).ObjName;

Editor.ListBox1.Items.InsertObject(ps,TAbstractElement(NewElement).ObjName, NewElement);

if TAbstractElement(NewElement).flType then inc(Count.Masses)

else inc(Count.Relations);

NewElement:=nil;

flIsSaved:=false;

CheckState;

end

else TAbstractElement(NewElement).Destroy;

end;

function TModel.EditElementParams: integer;

begin

with Editor do

Result:=TAbstractElement(ListBox1.Items.Objects[ListBox1.ItemIndex]).InputParamDlg.ShowModal;

end;

procedure TModel.DeleteElement;

var

num, code, i, index: integer;

flag: boolean;

s: string;

begin

with Editor do

begin

index:=ListBox1.ItemIndex;

s:=ListBox1.Items.Strings[index];

end;

val(Copy(s,Pos(' ', s)+1, length(s)), num, code);

flag:=TAbstractElement(Editor.ListBox1.Items.Objects[Editor.ListBox1.ItemIndex]).flType;

if flag then begin

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

if (TAbstractRelation(Editor.ListBox1.Items.Objects[i]).m1=num) or

(TAbstractRelation(Editor.ListBox1.Items.Objects[i]).m2=num) then begin

MessageDlg(TAbstractRelation(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+

' была присоединена к удаляемой массе.'#10'Откорректируйте параметры элемента '+

TAbstractRelation(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption,

mtWarning, [mbOk], 0);

TAbstractRelation(Editor.ListBox1.Items.Objects[i]).InputParamDlg.ShowModal;

end;

dec(Count.Masses);

end

else dec(Count.Relations);

TAbstractElement(Editor.ListBox1.Items.Objects[Editor.ListBox1.ItemIndex]).Destroy;

flIsSaved:=false;

end;

procedure TModel.SaveToIniFile(iniF: TIniFile; Section: string);

var i: integer;

begin

inherited SaveToIniFile(iniF,Section);

try

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SaveToIniFile(iniF, IntToStr(i));

except

raise ERangeError.Create('Ошибка записи');

end;

end;

procedure TModel.SaveToFile(Stream: TStream);

var i: integer;

begin

inherited SaveToFile(Stream);

try

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SaveToFile(Stream);

except

raise ERangeError.Create('Ошибка записи');

end;

end;

procedure TModel.SaveModel(FileName: string);

var

Stream: TStream;

IniF: TIniFile;

Ext: string;

begin

Ext:=UpperCase(ExtractFileExt(FileName));

if Ext='.MDL' then

begin

try

Stream:= TFileStream.Create(FileName, fmCreate);

SaveToFile(Stream);

finally

Stream.Free;

end;

end

else begin

try

IniF:=TIniFile.Create(FileName);

SaveToIniFile(IniF,'Model');

finally

IniF.Free;

end;

end;

end;

procedure TModel.LoadFromIniFile(IniF: TIniFile; Section: string);

var

i: integer;

InstControl: TControl;

InstComponentName: string;

begin

inherited LoadFromIniFile(IniF, Section);

i:=0;

while IniF.SectionExists(IntToStr(i)) do

begin

Section:=IntToStr(i);

InstComponentName:=IniF.ReadString(Section,'Class','');

InstControl:=Editor.FindControl(InstComponentName);

if InstControl<>nil then

begin

TToolButton(InstControl).OnClick(InstControl);

TAbstractElement(NewElement).LoadFromIniFile(IniF, Section);

Editor.ListBox1.Items.AddObject(TAbstractElement(NewElement).ObjName, NewElement);

TAbstractElement(NewElement).InputParamDlg.Caption:=TAbstractElement(NewElement).ObjName;

TAbstractElement(NewElement).GraphDlg.Caption:=TAbstractElement(NewElement).ObjName;

if TAbstractElement(NewElement).flType then inc(Count.Masses)

else inc(Count.Relations);

end

else MessageDlg('Неизвестный компонент "'+InstComponentName+'"', mtError, [mbOk], 0 );

CheckState;

inc(i);

end;

end;

procedure TModel.LoadFromFile(Stream: TStream);

var

InstControl: TControl;

InstComponentName: string;

begin

try

InstComponentName:=LoadStrFromStream(Stream);

if InstComponentName<>InstName then

begin

MessageDlg('Файл не содержит необходимых данных', mtError, [mbOk], 0 );

exit;

end;

inherited LoadFromFile(Stream);

while Stream.Position < Stream.Size do

begin

InstComponentName:=LoadStrFromStream(Stream);

InstControl:=Editor.FindControl(InstComponentName);

if InstControl<>nil then

begin

TToolButton(InstControl).OnClick(InstControl);

TAbstractElement(NewElement).LoadFromFile(Stream);

Editor.ListBox1.Items.AddObject(TAbstractElement(NewElement).ObjName, NewElement);

TAbstractElement(NewElement).InputParamDlg.Caption:=TAbstractElement(NewElement).ObjName;

TAbstractElement(NewElement).GraphDlg.Caption:=TAbstractElement(NewElement).ObjName;

if TAbstractElement(NewElement).flType then inc(Count.Masses)

else inc(Count.Relations);

end

else begin

MessageDlg('Неизвестный компонент "'+InstComponentName+'"', mtError, [mbOk], 0 );

exit;

end;

end;

CheckState;

except

raise ERangeError.Create('Ошибка чтения');

end;

end;

function TModel.LoadModel(FileName: string):boolean;

var

i: integer;

Stream: TStream;

Ext: string;

IniF: TIniFile;

begin

Result:=false;

Ext:=UpperCase(ExtractFileExt(FileName));

if Ext='.MDL' then

try

Stream := TFileStream.Create(fileName, fmOpenRead);

ClearData;

flReadFromFile:=true;

LoadFromFile(Stream);

finally

Stream.Free;

Editor.Memo1.Lines.Clear;

TInputDlg(InputParamDlg).RadioGroup1.ItemIndex:=Round(CalcMethod.code);

flReadFromFile:=false;

flIsSaved:=true;

Result:=true;

{//Warning! Add new parameters !

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractElement(Editor.ListBox1.Items.Objects[i]).AddParam;

flIsSaved:=false;

}

end

else

try

IniF:=TIniFile.Create(FileName);

ClearData;

flReadFromFile:=true;

LoadFromIniFile(IniF,'Model');

finally

IniF.Free;

Editor.Memo1.Lines.Clear;

TInputDlg(InputParamDlg).RadioGroup1.ItemIndex:=Round(CalcMethod.code);

flReadFromFile:=false;

flIsSaved:=true;

Result:=true;

end;

end;

function TModel.CheckModel: boolean;

var i: integer;

begin

Result:=true;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractRelation(Editor.ListBox1.Items.Objects[i]).SetPointers(Editor.ListBox1.Items);

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

if not TAbstractMassa(Editor.ListBox1.Items.Objects[i]).flIsRelated then

begin

Result:=false;

MessageDlg(TAbstractMassa(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+' не связана',

mtError, [mbOk], 0);

exit;

end;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

if TAbstractRelation(Editor.ListBox1.Items.Objects[i]).CheckPointers then

begin

Result:=false;

MessageDlg(TAbstractRelation(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+

' не объединяет 2 массы', mtError, [mbOk], 0);

exit;

end;

end;

procedure TModel.SetParamsInZero;

var i: integer;

begin

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SetParamsInZero;

with VisualDlg do begin

Time:=0;

CountSteps:=0;

Caption:=Editor.Edit1.Text;

RealModelTime:=0;

StartT:=0;

end;

case Round(CalcMethod.code) of

0: CalcMethod.digree:=4;

1: CalcMethod.digree:=2;

2: CalcMethod.digree:=1;

end;//case

end;

procedure TModel.Visio;

var i,j: integer;

begin

SetParamsInZero;

if not CheckModel then exit;

with VisualDlg do begin

TopMargin:= 32000 div 2;

MaxX:=-1e20;

MaxY:=-1e20;

MinX:=1e20;

MinY:=1e20;

for i:=0 to Editor.ListBox1.Items.Count-1 do

begin

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecLeft<MinX then MinX:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecLeft;

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecRight>MaxX then MaxX:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecRight;

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecTop<MinY then MinY:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecTop;

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecBottom>MaxY then MaxY:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).RecBottom;

end;

try

ScaleX:= ScrollBox1.ClientWidth\*0.8 / abs(MaxX-MinX);

ScaleY:=ScrollBox1.ClientHeight\*0.95 / abs(MaxY-MinY);

except

raise ERangeError.Create('Деление на ноль'#10'Некорректные геометрические параметры модели');

exit;

end;

LeftMargin:=(ScrollBox1.ClientWidth - Round((MaxX-MinX)\*ScaleX)) div 2 ;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SetImageSize(MinX, MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

for i:=0 to Editor.ListBox1.Items.Count-1 do

begin

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SetImageSize(MinX, MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

ScrollBox1.InsertControl(TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image);

end;

//Sort relation by order

Editor.SortRelationListBox.Clear;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

Editor.SortRelationListBox.Items.AddObject(Editor.ListBox1.Items.Strings[i],

Editor.ListBox1.Items.Objects[i]);

for i:=0 to Editor.SortRelationListBox.Items.Count-2 do

for j:=i+1 to Editor.SortRelationListBox.Items.Count-1 do

if TAbstractRelation(Editor.SortRelationListBox.Items.Objects[i]).Order>

TAbstractRelation(Editor.SortRelationListBox.Items.Objects[j]).Order

then Editor.SortRelationListBox.Items.Exchange(i,j);

//Set range image

RangeImage:= TImage.Create(ScrollBox1);

RangeImage.Left:=LeftMargin;

RangeImage.Top:=TopMargin \*2;

RangeImage.Height:=1;

RangeImage.Width:=1;

ScrollBox1.InsertControl(RangeImage);

ScrollBox1.VertScrollBar.Position:=TopMargin;

VisualDlg.StatusBar1.SimpleText:=Format(' Итерация %d Время %-10.4f ( %d%% ) Текущий шаг %-10.4f',

[VisualDlg.CountSteps, VisualDlg.time, Round(100\*VisualDlg.time/t), VisualDlg.StepT] );

ShowModal;

flIsModelling:=true;

for i:=0 to Editor.ListBox1.Items.Count-1 do

ScrollBox1.RemoveControl(TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image);

ScrollBox1.RemoveControl(RangeImage);

RangeImage.Destroy;

end;

Editor.SortRelationListBox.Clear;

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).GetParams; //restore parameters

end;

procedure TModel.Calculate(Sender: TObject);

var

i, j, code: integer;

h: TFloat;

begin

if VisualDlg.Time < t then

begin

VisualDlg.Timer1.Enabled:=false;

inc(VisualDlg.CountSteps);

code:=Round(CalcMethod.code);

for j:=1 to CalcMethod.digree do

begin

for i:=0 to Editor.SortRelationListBox.Items.Count-1 do

begin

TAbstractRelation(Editor.SortRelationListBox.Items.Objects[i]).CalcForce([VisualDlg.Time, j]);

if j=1 then TAbstractRelation(Editor.SortRelationListBox.Items.Objects[i]).AddValueToChart(VisualDlg.Time);

end;

{

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractRelation(Editor.ListBox1.Items.Objects[i]).CalcForce([VisualDlg.Time]);

}

if j = 1 then begin //calculate min step

VisualDlg.StepT:=ht;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

begin

h:=TAbstractMassa(Editor.ListBox1.Items.Objects[i]).CalcStep(e, ht);

if h < VisualDlg.StepT then VisualDlg.StepT:=h;

if VisualDlg.Time = 0 then

TAbstractMassa(Editor.ListBox1.Items.Objects[i]).AddValueToChart(VisualDlg.Time);

end;

VisualDlg.Time:=VisualDlg.Time + VisualDlg.StepT;

end;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractMassa(Editor.ListBox1.Items.Objects[i]).Calculate(VisualDlg.StepT, VisualDlg.Time, Code, j);

end; //for j

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

with VisualDlg do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).MoveImage(MinX, MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

for i:=0 to Editor.ListBox1.Items.Count-1 do

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

with VisualDlg do

TAbstractElement(Editor.ListBox1.Items.Objects[i]).MoveImage(MinX, MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

VisualDlg.StatusBar1.SimpleText:=Format(' Итерация %d Время %-10.4f ( %d%% ) Текущий шаг %-8.6f',

[VisualDlg.CountSteps, VisualDlg.Time, Round(100\*VisualDlg.time/t), VisualDlg.StepT] );

VisualDlg.Timer1.Enabled:=true;

end

else begin

VisualDlg.ToolButton1Click(Sender);

VisualDlg.ToolButton1.Enabled:=false;

MessageDlg('Расчет окончен'#10+'Время моделирования '+TimeToStr(VisualDlg.RealModelTime)+

#10+format('Средний шаг интегрирования %-7.6f',[VisualDlg.Time/VisualDlg.CountSteps]),

mtInformation, [mbOk], 0);

end;

end;

procedure TModel.ShowGraphs;

var i: integer;

begin

{

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractMassa(Editor.ListBox1.Items.Objects[i]).ShowGraphs;

}

MakeReportDlg(1);

end;

procedure TModel.HideGraphs;

var i: integer;

begin

VisualDlg.Timer1.Enabled:=false;

for i:=0 to Editor.ListBox1.Items.Count-1 do

TAbstractMassa(Editor.ListBox1.Items.Objects[i]).HideGraphs;

end;

procedure TModel.ShowReport(FName: string);

var

FN: array[0..255] of Char;

begin

ShellExecute(Editor.Handle, nil,

StrPCopy(FN, FName), nil, nil, Sw\_Show);

end;

procedure TModel.MakeReport(FName: string);

begin

WaitReportCreateDlg.Hint:=FName;

WaitReportCreateDlg.ShowModal;

end;

procedure TModel.WaitReportCreateDlgOnShow(Sender: TObject);

begin

with TWaitReportCreateDlg(Sender) do

begin

StaticText1.Font.Color:=clBlue;

StaticText1.Caption:='Создаётся отчёт...';

WaitReportCreateDlg.Refresh;

Animate1.Active:=true;

Animate1.CommonAVI:=aviCopyFile;

if CreateReport(WaitReportCreateDlg.Hint) then

begin

StaticText1.Caption:='Отчёт создан';

ModalResult:=mrOk;

end

else

begin

StaticText1.Font.Color:=clRed;

StaticText1.Caption:='Ошибка создания документа WinWord';

ModalResult:=mrCancel;

end;

Animate1.Active:=false;

end;

end;

function TModel.CreateReport(FName: string): boolean;

var

qi: TQRImage;

i: integer;

ScaleX, ScaleY: TFloat;

TopMargin, LeftMargin: integer;

m: TMetafile;

Report: TQuickRep;

begin

//Result:=true;

WDoc := CreateOleObject('Word.Basic');

Result:= VarType(WDoc) = VarDispatch;

if Result then

try

WDoc.FileNew('Normal');

WDoc.FilePageSetup(PaperSize:='9', TopMargin:='2 см', BottomMargin:='2 см',

LeftMargin:='2 см', RightMargin:='2 см', PageWidth:='21 см',

PageHeight:='29,7 см', Orientation:=0);

WDoc.FormatFont(Points:=14, Font:='Times New Roman Cyr',Bold:=1);

WDoc.FormatParagraph(After:='12 пт',LineSpacingRule:=1,Alignment:=1);

WDoc.Insert('Результаты моделирования'+ #13);

WDoc.FormatFont(Points:=14, Font:='Times New Roman Cyr',Bold:=0);

WDoc.FormatParagraph(Alignment:=1);

WDoc.Insert('Модель: '+Editor.Edit1.Text+#13);

Report:= TQuickRep.CreateParented(0);

Report.Units:=pixels;//TQRUNIT(2);

TopMargin:=0;

ScaleX:= Report.Page.Width / abs(VisualDlg.MaxX-VisualDlg.MinX);

ScaleY:= Report.Page.Length / abs(VisualDlg.MaxY-VisualDlg.MinY);

LeftMargin:=Round(Report.Page.Width - (VisualDlg.MaxX-VisualDlg.MinX)\*ScaleX) div 2 ;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SetImageSize(VisualDlg.MinX,

VisualDlg.MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

for i:=0 to Editor.ListBox1.Items.Count-1 do

begin

if not TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

TAbstractElement(Editor.ListBox1.Items.Objects[i]).SetImageSize(VisualDlg.MinX,

VisualDlg.MinY, ScaleX, ScaleY, TopMargin, LeftMargin);

qi:= TQRImage.Create(Report);

qi.AutoSize:=false;

qi.Center:=true;

qi.Stretch:=true;

qi.Name:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Name;

qi.Left:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Left;

qi.Top:= TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Top;

qi.Height:= TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Height;

qi.Width:= TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Width;

qi.Picture:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Picture;

Report.InsertControl(qi);

qi:=nil;

end; //for

Report.Prepare;

m:=Report.QRPrinter.GetPage(Report.QRPrinter.PageNumber);

if m<>nil then begin

m.SaveToFile(TempFName+'wmf');

WDoc.InsertPicture(Name:=TempFName+'wmf',LinkToFile:='0');

{

clipboard.Assign(m);

WDoc.EditPaste;

}

end

else

Result:=false;

Report.Destroy;

WDoc.InsertPara;

for i:=0 to Editor.ListBox1.Items.Count-1 do

begin

WDoc.FormatFont(Points:=14);

WDoc.FormatParagraph(After:='0 пт',Alignment:=3);

WDoc.Insert(IntToStr(i+1)+' - '+TAbstractElement(Editor.ListBox1.Items.Objects[i]).AboutStr+#13);

end;

MakeReportDlg(0);

WaitReportCreateDlg.Refresh;

WDoc.FileSaveAs(ChangeFileExt(FName, '.doc'));

WDoc.FileClose;

except

Result:=false;

end;

WDoc:= UnAssigned;

end;

procedure TModel.OnClickSeriesCheckBox(Sender: TObject);

begin

ReportDlg.Chart1.Series[TCheckBox(Sender).Tag].Active:=TCheckBox(Sender).Checked;

//ReportDlg.Chart1.Series[TCheckBox(Sender).Tag].ShowInLegend:=TCheckBox(Sender).Checked;

end;

procedure TModel.OnClickSaveButtonOnReportDlg(Sender: TObject);

var

i: integer;

begin

try

if ReportDlg.Tag = 0 then

begin

WDoc.Insert(#13);

ReportDlg.Chart1.SaveToMetafile(TempFName+'wmf');

WDoc.InsertPicture(Name:=TempFName+'wmf',LinkToFile:='0');

WDoc.Insert(#13);

for i:=0 to ReportDlg.Chart1.SeriesCount-1 do

if ReportDlg.Chart1.Series[i].Active then

begin

WDoc.Insert(ReportDlg.Chart1.Series[i].Title+#13);

end;

end

else

ReportDlg.Chart1.CopyToClipboardMetafile(true);

except

MessageDlg('Сбой при создания отчета', mtError, [mbOk], 0 );

end;

end;

function TModel.GetModelInformation:string;

var

i: integer;

Mas, C: TFloat;

begin

Mas:=0;

C:=0;

for i:=0 to Editor.ListBox1.Items.Count-1 do

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

Mas:=Mas+TAbstractMassa(Editor.ListBox1.Items.Objects[i]).GetMassa

else

if TAbstractRelation(Editor.ListBox1.Items.Objects[i]).c<>0

then

C:=C+1/TAbstractRelation(Editor.ListBox1.Items.Objects[i]).c;

Result:=format('Масса колебательной системы = %-10.3f'#13+

'Жесткость системы %-10.3f',[Mas,1/C]);

end;

procedure TModel.ModelInformation(Sender: TObject);

begin

MessageDlg(GetModelInformation,

mtInformation, [mbOK], 0) ;

end;

function TModel.MakeReportDlg;

var

j,n,i,k: integer;

chart: TChart;

ser: TChartSeries;

st,bst: TStaticText;

ch: TCheckBox;

ss: string;

begin

ReportDlg:= TReportDlg.CreateParented(0);

ReportDlg.SaveBtn.OnClick:=OnClickSaveButtonOnReportDlg;

ReportDlg.Tag:=Where;

j:=0;

n:=-1;

for i:=0 to Editor.ListBox1.Items.Count-1 do

//if TAbstractElement(Editor.ListBox1.Items.Objects[i]).flType then

if TAbstractElement(Editor.ListBox1.Items.Objects[i]).GraphFlag then

begin

st := TStaticText.CreateParented(0);

st.Left:=3;

st.AutoSize:= true;

st.Top:=17\*j+2;

st.Caption:=Editor.ListBox1.Items.Strings[i];

st.ShowHint:=true;

st.Hint:=TAbstractElement(Editor.ListBox1.Items.Objects[i]).Image.Hint;

ReportDlg.ScrollBox1.InsertControl(st);

chart:=TAbstractMassa(Editor.ListBox1.Items.Objects[i]).GraphChart;

if chart<>nil then

for k:=1 to chart.SeriesCount do

begin

inc(n);

ch:= TCheckBox.CreateParented(0);

ch.Width:=17;

ch.Checked:=false;

ch.Top:=st.Top;

ch.Tag:=n;

ch.OnClick:=OnClickSeriesCheckBox;

ch.ShowHint:=true;

ch.Hint:=chart.Series[k-1].Title;

if chart.SeriesCount>1 then ch.Left:=64+24\*k

else ch.Left:=64+24\*3;

ReportDlg.ScrollBox1.InsertControl(ch);

ss:=ColorToString(ReportDlg.Chart1.GetFreeSeriesColor(true));

ser:=TLineSeries.Create(ReportDlg.Chart1);

ser.Active:=false;

ser.ShowInLegend:=false;

ser.Title:=chart.Series[k-1].Title+' ('+TAbstractElement(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+')';

{

case k of

1: if chart.SeriesCount>1 then ser.Title:='перемещение ('+TAbstractMassa(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+')'

else ser.Title:='сила ('+TAbstractElement(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+')';

2: ser.Title:='скорость ('+TAbstractElement(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+')';

3: ser.Title:='сила ('+TAbstractElement(Editor.ListBox1.Items.Objects[i]).InputParamDlg.Caption+')';

end; //case

}

ser.Title:=ser.Title+' (цвет - '+Copy(ss,3,length(ss)-2)+' )';

ser.AssignValues(chart.Series[k-1]);

ReportDlg.Chart1.AddSeries(Ser);

ch:=nil;

end;

st:=nil;

inc(j);

end;

Result:=ReportDlg.ShowModal;

ReportDlg.Destroy;

end;

end