Содержание

1. Основные понятия

1.1 Постановка задачи и треблвания к программе

2. Этапы разработки программы «Информационная система ГИБДД»

2.1 Анализ предметной области и выбор метода решения

2.2 Анализ предметной области и выбор метода решения

2.3 Разработка алгоритма программы

2.3.1 Понятие алгоритма, виды алгоритмов

2.3.2 Алгоритм работы программы

2.3.3 Блок - схема программы

2.4 Инструментальные средства разработки

2.4.1 Анализ инструментальных средств

2.5 Разработка интерфейса программы

2.5.1 Понятие интерфейса

2.5.2 Виды интерфейса

2.5.3 Разработка интерфейса программы «Информационная система ГИБДД»

2.6 Тестирование программы

2.6.1 Понятие тестирования программных средств

2.6.2 Виды и методы тестирования

2.6.3 Процесс и результат тестирования

2.7 Программная документация

2.7.1 Руководство программиста

2.7.2 Руководство оператора

3. Эксплуатация программных средств

3.1 Эксплуатация программного обеспечения

4. Экономический расчет программного продукта

4.1 Расчет себестоимости программного продукта «Информационная система ГИБДД»

4.2 Экономический эффект

5. Техника безопасности

5.1 Общие требования по техники безопасности

5.2 Техника безопасности при работе на компьютере

5.2.1 Требования безопасности перед началом работы на ПК

5.2.2 Требования безопасности во время работы с ПК

# 5.2.3 Требования безопасности после окончания работы с ПК

Заключение

Литература

1. Основные понятия

База данных. СУБД.

Четкого и однозначного определения базы данных не существует. Тем не менее, можно определить базу данных как физическое пространство (место на внешнем носителе компьютера), на котором в строго определенном порядке записываются и хранятся конкретные значения реквизитов (данные) первичных информационных форм, относящиеся к одному роду объектов учета.

Другое более общее определение определяет базу данных как хранилище данных, предназначенных для совместного использования. Здесь особо выделяется возможность совместного использования информации хранящейся в базе данных многими пользователями одновременно. Многопользовательский режим доступа к данным, хранящимся в базе, может быть организован различными способами:

1. использование в сети файл-серверов, на которых физически хранятся вся база данных. При такой организации каждый пользователь работает с базой данных так, как будто она размещена на собственном компьютере. Недостатком такой организации многопользовательской БД является большая нагрузка на каналы связи т. к. при обслуживании запроса порции данных считываются с диска центрального компьютера (файл-сервера) небольшими порциями и передаются на рабочее место, где происходит их обработка;
2. технология клиент/сервер, при которой вся обработка информации, все транзакции, задаваемые пользователями, выполняется самим сервером, а на рабочие станции передаются только результаты. К пропускной способности каналов связи в этом случае предъявляются значительно меньшие требования, однако центральный компьютер, обслуживающий базу данных, должен быть значительно мощнее, чем в первом варианте. При этом, рабочие станции могут быть бездисковыми терминалами
3. распределенная структура – когда части (фрагменты) одной базы данных физически находятся на различных серверах сети. По такому принципу строятся глобальные информационные системы типа Internet.

Для создания и обслуживания таких систем используется специализированное программное обеспечение - системы управления базами данных СУБД.

К основным функциям СУБД любого типа можно отнести:

1. создание (конструирование) базы данных путем описания структуры хранимой информации и взаимосвязей между ее частями;
2. занесение, хранение и удаление информации из базы данных - т. е. поддержание БД в актуальном состоянии;
3. обслуживание выборок и запросов пользователей

Различные СУБД позволяют создавать и обслуживать базы данных различной структуры: иерархические, сетевые и т. д. Наибольшее распространение получили так называемые реляционные БД. Реляционные базы данных представляют собой набор связанных таблиц и ничего кроме них. Термин «реляционная» указывает на то, что между таблицами базы данных могут быть установлены различные отношения. РСУБД составляют один из крупных сегментов рынка баз данных: они включают все от систем клиент/сервер до настольных систем.

Как отмечалось выше, реляционная модель БД рассматривает все данные как группы таблиц или отношений, которые содержат фиксированные количества рядов и столбцов. Иными словами многие объекты, используемые в реляционной базе данных, аналогичны объектам электронных таблиц. Рассмотрим основные термины и определения связанные с РСУБД.

Поле – базовый элемент любой базы данных, не обязательно реляционной. Поля это элементарный информационный объект базы данных. В данном случае, «элементарный», означает, что поле не может быть разбито на более мелкие порции информации. Кроме того, в каждом поле может храниться только строго определенный тип информации (текстовые поля, поля типа дата/время, числовые поля и т. п.). Большинство СУБД поддерживают возможность создания полей следующих типов:

* текстовые (для хранения строк размером до 255 символов);
* числовые (целочисленное, с плавающей точкой и т. п.);
* memo поля – поля для хранения тестовых фрагментов любого размера;
* дата/время – поля, в которых могут храниться даты и (или) время в национальном формате;
* логические – поля для хранения утверждений типа ДА/НЕТ, ВКЛЮЧЕНО/ВЫКЛЮЧЕНО, ИСТИНА/ЛОЖЬ и т. п.;

Запись – набор данных специфицирующих некоторый объект. Например в БД автотранспортных средств каждая запись содержит сведения о транспортном средстве (госномер, марку, год выпуска, № кузова и т. п.). Каждая запись БД содержит уникальный набор информации – в нашем примере, каждая запись представляет данные о конкретном транспортном средстве. В РСУБД записи не хранятся в каком либо порядке набора. Иными словами в концепции РСУБД вообще не существует номера записи, как в системах другого типа.

Таблица – это набор полей. Данные, содержащиеся в таблице, хранятся в виде записей. Каждая таблица базы данных представляет некоторый тип хранящихся в ней объектов. В БД может быть любое количество таблиц, между которыми могут быть установлены различные отношения. Тот факт, что таблица представляет только один тип объекта, отнюдь не является недостатком. Наоборот, это один из ключей к созданию эффективной базы данных.

Ключевое поле – это поле, которое используется для связи между двумя и более таблицами. Ключи – это поля, которые являются общими для связываемых таблиц. При этом значение этих полей в связанных таблицах дублируется. Ключи могут быть первичными, внешними или составными. Позже мы рассмотрим эти типы ключей.

Отношение – это связь, устанавливаемая между двумя и более таблицами посредством ключевого поля. Принципиально возможны три типа отношений: один к одному, один к многим и многие к многим.

Соединение – виртуальная таблица, создаваемая, когда пользователь запрашивает информацию из различных таблиц связанных отношением. Ключевые поля в этом случае используются для поиска соответствующих записей в различных таблицах, из которых формируется соединение.

Первичный ключ – уникально идентифицирует каждую запись в таблице и не имеет повторяющихся значений. Выбор поля в качестве первичного ключа – одно из важнейших решений принимаемых при проектировании БД.

Если запись в таблице не может быть однозначно идентифицирована каким-либо одним полем, то можно использовать составной ключ – группу полей. Составные ключи используются значительно реже первичных.

Внешний ключ – это поле (или группа полей) одной таблицы, для которого имеется дублированное значение в другой, связанной таблице. В отличие от первичных ключей, внешние ключи зачастую многократно повторяются при установлении отношения один к многим.

1.1 Постановка задачи и треблвания к программе

У ГИБДД есть четыре наиболее важные функциональные задачи:

* Регистрация автотранспортных средств, при совершении сделки купли-продажи;
* Разработка мер, повышающих безопасность дорожного движения и выполнение всех мер при совершении ДТП (дорожно-транспортное происшествие) на улицах города (регистрация, разбор, выявление виновных, автоэкспертиза и т.п.);
* Борьба с угоном автотранспортных средств;
* Оперативный поиск угнанных машин и задержание преступников.

ГИБДД занимается выделением учетом номерных знаков на автотранспорт. К автотранспортным средствам относятся: легковые, грузовые автомобили, прицепы, полуприцепы, мотоциклы, тракторы, автобусы, микроавтобусы. На разные виды транспорта выдаются разные виды номеров и в базу данных заносятся разные характеристики. Номера могут выделяться как частным владельцам, так и организациям. В справочнике номеров, выданных организации, дополнительно фиксируется: номер, ФИО владельца, его адрес, марка автомобиля, дата выпуска, объем двигателя, номера двигателя, шасси и кузова, цвет и т.п. В справочнике номеров, выданных организации, дополнительно фиксируется: название организации, район, адрес, руководитель. Существует справочник свободных номеров (серия, диапазон номеров). ГИБДД периодически проводит технический осмотр (ТО) машин. Для прохождения техосмотра необходима квитанция об оплате налогов, сумма оплаты зависит от объема двигателя. Периодичность прохождения зависит от года выпуска и вида транспортного средства. Технические характеристики, проверяемые на ТО и допуски, также зависят от вида транспортного средства.

ГИБДД занимается учетом и анализом ДТП (дорожно-транспортное происшествие). При регистрации ДТП фиксируется: дата, тип происшествия (наезд на пешехода, наезд на ограждение либо столб, лобовое столкновение, наезд на впереди стоящий транспорт, боковое столкновение на перекрестке и т.п.), место происшествия, марки пострадавших автомобилей, государственный номер, тип машины (легковая, грузовая, специальная), краткое содержание, число пострадавших, сумма ущерба, причина, дорожные условия и т.п. Анализ накопленной по ДТП статистике поможет правильно расставить запрещающие и предупреждающие знаки на улицах города, а так же спланировать местонахождение постов патрульных.

Угон либо исчезновение виновника ДТП с места происшествия требует оперативного вмешательства всех постов ГИБДД и патрульных машин. Для информации о разыскиваемой машине ее данные (включая номера двигателя и кузова) извлекаются из базы зарегистрированных номеров и передаются по рации всем постам. Ведение статистики угонов, ее анализ и опубликование результатов в СМИ поможет снизить количество угонов, а хозяевам машин принять необходимые меры (самые угоняемые марки, самый популярный способ вскрытия, самые надежные сигнализации и т.п.).

2. Этапы разработки программы «Информационная система ГИБДД»

2.1 Техническое задание

Настоящее техническое задание распространяется на разработку информационной системы ГИБДД. Предполагается, что данную систему будут использовать сотрудники ГИБДД.

Основания для разработки

Информационная система разрабатывается на основании задания на курсовое проектирование, выданное Челябинским энергетическим колледжем имени С.М. Кирова 1 декабря 2007 года. Срок окончания разработки 17 марта 2008 года.

Назначение разработки

Информационная система ГИБДД предназначена для выделения и учета номерных знаков на автотранспорт, учета и анализа ДТП (дорожно-транспортного происшествия), угона либо исчезновения виновника ДТП с места происшествия.

Анализ накопленной по ДТП статистике поможет правильно расставить запрещающие и предупреждающие знаки на улицах города, а так же спланировать местонахождение постов патрульных. Ведение статистики угонов, ее анализ и опубликование результатов в СМИ поможет снизить количество угонов, а хозяевам машин принять необходимые меры (самые угоняемые марки, самый популярный способ вскрытия, самые надежные сигнализации и т.п.)

Требования к программе

Система должна обеспечивать выполнение следующих функций:

* + Регистрацию автотранспортных средств, при совершении сделки купли-продажи;
  + Меры, повышающие безопасность дорожного движения и выполнение всех мер при совершении ДТП на улицах города;
  + Борьба с угоном автотранспортных средств;
  + Оперативный поиск угнанных средств.

Исходные данные:

* + Перечень транспорта;
  + Номерные знаки на автотранспорт;
  + Данные владельца автотранспорта;
  + Справочник свободных номеров;
  + Типы происшествий.

Результат:

* + Перечень и общее число организаций, которым выделены номера либо с указанной серией, либо за указанный период;
  + Сведения о владельце автотранспортного средства по государственному номеру автомашины;
  + «Досье» на автомобиль по государственному номеру - номера двигателя, кузова, шасси, участвовал ли в ДТП, прошел ли техосмотр;
  + Статистика по любому типу ДТП;
  + Данные о количестве ДТП, совершаемых водителями в нетрезвом виде;
  + Список машин, отданных в розыск, будь то скрывшиеся с места ДТП или угнанные;
  + Данные об эффективности розыскной работы: количество найденных машин;
  + Перечень угонов за указанный период;

Требования к надежности

* + 1. Предусмотреть контроль вводимой информации.
    2. Обеспечить блокировку некорректных действий пользователя.
    3. Обеспечить ценность хранимой информации.

Требования к составу и параметрам технических средств

Минимальные требования к компьютеру и П.О:

Windows 2000 XP Professional

Servise Pack 1, 2

Windows XP Home Edition

Servise Pack 1, 2

Pentium 3, 866 Гц, 512 Мб, не менее 800 Мб на жестком диске.

Требования к программной совместимости

Система должна работать в ОС Windows.

Требования к программной документации

Программные модули должны быть самодокументированны. Программная система должна включать справочную информацию.

В состав сопровождающей документации необходимо включить:

1. Руководство системного программиста.

2. Руководство пользователя.

3. Пояснительная записка.

Технико-экономические показатели.

Программа должна быстро окупить затраченные на нее средства.

Стадии и этапы разработки

1. Постановка задачи.
2. Анализ предметной области.
3. Разработка алгоритма
4. Разработка интерфейса.
5. Кодирование.
6. Тестирование.
7. Оформление документации.
8. Порядок контроля и приемки

Программа должна быть представлена на отдельном диске в виде инсталляционного пакета, необходимо провести тесты на ввод неправильных данных и т.д.

Примечание.

2.2 Анализ предметной области и выбор метода решения

Современный мир информационных технологий трудно представить себе без возможности обработки больших объёмов информации. Такие объёмы информации удобно обрабатывать с помощью баз данных. Практически все системы в той или иной степени связаны с долговременным хранением и обработкой информации. Фактически, информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных. Большинство операций не может быть выполнено вручную. Любые административные решения требуют более чёткой и точной оценки текущей ситуации и возможных перспектив ёё изменения.

В данном дипломном проекте необходимо разработать «Информационную систему ГИБДД». Она должна обеспечивать просмотр, обработку, ввод новых данных, поиск данных по определённому критерию. Целью создания данной базы данных является:

1. повышение производительности работы;
2. более надёжное хранение информации.

Стадии и этапы разработки:

1. анализ;
2. проектирование;
3. программирование;
4. тестирование и отладка.

На этапе анализа производится описание задачи. Описание является полным, последовательным, доступным для чтения и обзора различных заинтересованных лиц. В ходе выполнения этого этапа получают следующее:

- уточняются требования, приведённые в задании на дипломное проектирование;

- разрабатывается спецификация на программный продукт.

Итог выполнения этапа анализа – спецификация, содержащая конкретное описание программного продукта.

На этапе проектирования происходит иерархическое разбиение задачи создания программного продукта на подзадачи. На этом этапе производится следующее:

- формируется структура программного продукта и разработка алгоритмов, задаваемых спецификацией;

- выбирается структура информации в базе данных.

Результат работы этапа проектирования – алгоритм решения задачи, структура программного продукта и базы данных.

Цель этапа программирования – создание программного продукта в выбранной среде программирования. Итогом выполнения этого этапа является рабочая программа, отвечающая всем требованиям, указанным в задании на дипломном проектирование.

На этапе тестирования и отладки выявляются ошибки в программном продукте, проверяется работоспособность программного продукта и его соответствие спецификации. В ходе выполнения этого этапа выполняются следующие работы:

1. подготавливаются данные для отладки;
2. планируется сам процесс отладки;
3. проводится испытание программного продукта.

Результат выполнения этого этапа – оттестированный и отлаженный программный продукт.

2.3 Разработка алгоритма программы

2.3.1 Понятие алгоритма, виды алгоритмов

Слово «Алгоритм» происходит от algorithmi - латинского написания имени аль-Хорезми, под которым в средневековой Европе знали величайшего математика из Хорезма (город в современном Узбекистане) Мухаммеда бен Мусу, жившего в 783-850 гг. В своей книге «Об индийском счете» он сформулировал правила записи натуральных чисел с помощью арабских цифр и правила действий над ними столбиком. В дальнейшем алгоритмом стали называть точное предписание, определяющее последовательность действий, обеспечивающую получение требуемого результата из исходных данных. Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством. Создание алгоритма, пусть даже самого простого, - процесс творческий. Он доступен исключительно живым существам, а долгое время считалось, что только человеку. Другое дело - реализация уже имеющегося алгоритма. Ее можно поручить субъекту или объекту, который не обязан вникать в существо дела, а возможно, и не способен его понять. Такой субъект или объект принято называть формальным исполнителем. Примером формального исполнителя может служить стиральная машина-автомат, которая неукоснительно исполняет предписанные ей действия, даже если вы забыли положить в нее порошок. Человек тоже может выступать в роли формального исполнителя, но в первую очередь формальными исполнителями являются различные автоматические устройства, и компьютер в том числе. Каждый алгоритм создается в расчете на вполне конкретного исполнителя. Те действия, которые может совершать исполнитель, называются его его допустимыми действиями. Совокупность допустимых действий образует систему команд исполнителя. Алгоритм должен содержать только те действия, которые допустимы для данного исполнителя.

Данное выше определение алгоритма нельзя считать строгим - не вполне ясно, что такое «точное предписание» или «последовательность действий, обеспечивающая получение требуемого результата». Поэтому обычно формулируют несколько общих свойств алгоритмов, позволяющих отличать алгоритмы от других инструкций.

Алгоритм – система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (входной информации) в желаемый результат (выходную информацию) за конечное число шагов.

Алгоритм решения задачи имеет ряд обязательных свойств:

1. Дискретность (прерывность, раздельность) - алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов. Каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего.
2. Определенность - каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.
3. Результативность (конечность) - алгоритм должен приводить к решению задачи за конечное число шагов.
4. Массовость - алгоритм решения задачи разрабатывается в общем виде, то есть, он должен быть применим для некоторого класса задач, различающихся только исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.
5. Формализованность – предписания алгоритма должны быть записаны на некотором формальном (искусственном) языке.

В алгоритме отражаются логика и способ формирования результатов решения с указанием необходимых расчетных формул, логических условий, соотношений для контроля достоверности выходных результатов. В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения комплекса задач.

Алгоритм решения комплекса задач и его программная реализация тесно взаимосвязаны. Специфика применяемых методов проектирования алгоритмов и используемых при этом инструментальных средств разработки программ может повлиять на форму представления и содержание алгоритма обработки данных.

Алгоритм применительно к вычислительной машине – точное предписание, т.е. набор операций и правил их чередования, при помощи которого, начиная с некоторых исходных данных, можно решить любую задачу фиксированного типа.

Виды алгоритмов как логико-математических средств отражают указанные компоненты человеческой деятельности и тенденции, а сами алгоритмы в зависимости от цели, начальных условий задачи, путей ее решения, определения действий исполнителя подразделяются следующим образом:

1) Механические алгоритмы, или иначе детерминированные, жесткие (например алгоритм работы машины, двигателя и т.п.);

2) Гибкие алгоритмы, например стохастические, т.е. вероятностные и эвристические.

Механический алгоритм задает определенные действия, обозначая их в единственной и достоверной последовательности, обеспечивая тем самым однозначный требуемый или искомый результат, если выполняются те условия процесса, задачи, для которых разработан алгоритм.

3) Вероятностный (стохастический) алгоритм дает программу решения задачи несколькими путями или способами, приводящими к вероятному достижению результата.

4) Эвристический алгоритм (от греческого слова «эврика») – это такой алгоритм, в котором достижение конечного результата программы действий однозначно не предопределено, так же как не обозначена вся последовательность действий, не выявлены все действия исполнителя. К эвристическим алгоритмам относят, например, инструкции и предписания. В этих алгоритмах используются универсальные логические процедуры и способы принятия решений.

5) Линейный алгоритм – набор команд (указаний), выполняемых последовательно во времени друг за другом.

6) Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

7) Циклический алгоритм – алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.

Цикл программы – последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

Вспомогательный (подчиненный) алгоритм (процедура) – алгоритм, ранее разработанный и целиком используемый при алгоритмизации конкретной задачи. В некоторых случаях при наличии одинаковых последовательностей указаний (команд) для различных данных с целью сокращения записи также выделяют вспомогательный алгоритм.

2.3.2 Алгоритм работы программы

Программа «Информационная система ГИБДД» выполняется по следующему алгоритму:

* 1. Загрузка программы в оперативную память;
  2. Ожидание реакции пользователя;
  3. В начале загружается главная форма. В главной форме пользователь может выбрать нужную базу данных, просмотреть справку и информацию о программе.
  4. При выборе базы данных «Справочник частных владельцев» пользователь может просмотреть информацию о владельце, редактировать таблицу, добавить новые номера и закрыть эту форму.
  5. При выборе базы данных «Справочник организации» пользователь также сможет редактировать таблицу, просмотреть информацию или закрыть эту форму.
  6. При выборе базы данных «Регистрация ДТП» загружается форма базы данных ДТП, где пользователь может редактировать таблицу, просмотреть информацию или закрыть эту форму.
  7. При выборе базы данных «Угон автомобилей» загружается форма, в которой хранятся данные об угонах или скрывшихся с места ДТП автомобилях. Здесь пользователь также может редактировать таблицу, просмотреть информацию или закрыть эту форму.
  8. При выборе пункта меню «Справка » отобразится окно с подсказками по работе в программе.
  9. При выборе пункта «О программе» будет выведена информация о разработчике программы.
  10. При нажатии кнопки «Выход» загружается форма, предупреждающая о действительности выхода из программы. При нажатии кнопки «нет» пользователь вернется в главную форму, а при нажатии кнопки «да» пользователь выйдет из программы.

2.3.3 Блок - схема программы

Да

Да

Нет

Нет

Формирование запроса

БД

Конец

Результат запроса

Загрузка

Ожидание

действия

Запрос

Ввод данных

Начало

Выход

2.4 Инструментальные средства разработки

2.4.1 Анализ инструментальных средств

При выборе программных средств, для разработки программы «Информационная система ГИБДД», необходимо учитывать возможности описания структуры данных, определение модулей программы и связи между ними, оценки развитости аппарата структур и типов данных.

Учёт этих возможностей позволит сделать программу легкодоступной для использования, позволит предупредить возникновение логических ошибок, обеспечить надёжность программного обеспечения и его модифицируемость.

Для написания программы «Информационная система ГИБДД» была выбрана среда программирования Delphi.

Существует несколько способов решения поставленной задачи:

- Flash-программирование;

- языки программирования (Паскаль, Си, Visual Basic);

- среда объектного визуального программирования Delphi;

Flash – программирование

Macromedia Flash – очень мощное, при этом простое в использовании, средство создания анимированных проектов на основе векторной графики с встроенной поддержкой интерактивности.

Flash не требует ничего дополнительного для перехода со ссылке, открытия окна браузера или выполнения чего-либо посредством HTML.

Имеется возможность создавать кнопки, нажатие которых приводит к выдаче информации и воспроизведению звука или переносящее в другое место фильма проекта. Поэтому презентация в проекте может проходить в предопределённой последовательности или по пути указанному пользователем.

Для разработки интерактивных элементов во Flash используют три основных компонента: событие (event), порождающее определённое действие; действие (action), порождающее тем или иным образом событием; целевой объект (target), выполняющий действие или изменяемый событием.

Языки программирования

Прогресс компьютерных технологий определил процесс появления новых разнообразных звуковых систем для записи алгоритмов – языков программирования. Смысл появления такого языка – оснащённый набор вычислительных формул дополнительной информации, превращает данный набор в алгоритм.

Можно писать программы непосредственно на машинном языке, хотя это и сложно. В начале 1950-х г.г. машинный язык был единственным языком. Далее были созданы языки высокого уровня, работающие через трансляционные программы, которые вводят «исходный код» (гибрид английских слов и математических выражений, которые считает машина), и в конечном итоге заставляет компьютер выполнять соответствующие команды, которые даются на машинном языке.

Существуют также другие среды программирования, с помощью которых можно было бы разработать данную программу. Такие как:

- среда программирования С++ - это универсальный язык программирования, который позволяет разрабатывать программы в соответствии с разными типами программирования: процедурным, объектно-ориентированным, параметрическим.

- система программирования Microsoft Visual Basic for Windows, обладая простыми в обращении средствами визуального проектирования, позволяет в полной мере использовать преимущества графической системы Windows и быстро конструировать эффективные приложения.

2.4.2 Среда программирования Delphi

Для реализации решаемой задачи в дипломном проекте необходимо осуществление следующих операций:

1. Добавление и удаление в базе данных записей.
2. Изменение и обновление записей.
3. Быстрое осуществление поиска нужных записей по критериям.

Все эти операции должны осуществляться в удобной для пользователя форме и не вызывать затруднений при работе. Поэтому я выбрал среду программирования Borland Delphi 7.0.

Для хранения данных выбрана база данных Paradox т.к.:

1) Это достаточно известная и изученная хорошо зарекомендовавшая себя БД, особенно для хранения небольших объемов данных

2) БД Paradox поддерживает одновременную работу нескольких пользователей по сети.

3) БД Paradox поддерживает язык запросов SQL

Для разработки приложения БД была выбрана среда разработки Borland Delphi 7 и средство доступа к данным BDE (Borland Database Engine).

BDE было выбрано по следующим причинам:

1. этот продукт много лет присутствует на рынке, поэтому он отличается высокой устойчивостью и надежностью;
2. существует большое количество литературы по программированию BDE;
3. BDE отличается высокой скоростью работы и большим количеством сервисных функций;
4. некоторая сложность при установке программ, использующих BDE компенсируется всеми вышеперечисленными преимуществами.

Система разработки Borland Delphi присутствует на рынке программного обеспечения более 10 лет. За это время фирмой Borland были реализованы многие функции, которые используют программисты при разработке программного обеспечения. Borland Delphi 7 отличается высокой скоростью работы, надежностью самой системы и созданных программ, возможностью разработки новых компонентов (программных объектов, включающих в себя определенную функциональность), возможностью разрабатывать разные классы приложения, компоненты ActiveX, приложения для работы с базами данных (локальные и серверные), WEV приложения, сервисы Windows NT, консольные приложения Windows и приложения Windows с графическим интерфейсом. Также существует пакет Borland KYLIX, который позволяет разрабатывать приложения для операционной системы Linux и во многом совместим с Delphi на уровне исходных текстов. Компания Borland занимается постоянным совершенствованием Delphi, например последняя версия Delphi 8 и недавно вышедшая Delphi 2005 компилируют приложения под новую платформу Microsoft .Net. Для разработки выбран Delphi 7, так как программы, созданные в нем, могут работать на любых версиях Windows без установки дополнительных библиотек.

2.5 Разработка интерфейса программы

2.5.1 Понятие интерфейса

Термин «интерфейс» широко используется в областях, где человеку приходится иметь дело с обработкой информации на компьютере. В переводе с английского языка Interface означает внешнее лицо.

Интерфейс – совокупность средств и правил, которые обеспечивают взаимодействие устройств, программ и человека. Особенно важен интерфейс, обеспечивающий взаимодействие пользователя с персональным компьютером, называемый пользовательским интерфейсом. От удобства этого интерфейса во многом зависит успех нового программного продукта в конкурентной борьбе на рынке программных средств. Пользовательский интерфейс может быть символьным и графическим.

2.5.2 Виды интерфейса

В компьютерном мире известно множество разновидностей интерфейсов:

* интерфейс пользователя,
* графический интерфейс,
* интерфейс ввода-вывода,
* внешний или внутренний интерфейс,
* интеллектуальный интерфейс,
* человеко-машинный интерфейс,
* программный интерфейс и т.д.

Графический интерфейс, по сравнению с символьным, воспринимается как более понятный и интуитивно ясный.

Графический пользовательский интерфейс – интерфейс, где для взаимодействия человека и компьютера используются графические средства. Ярким примером графического пользовательского интерфейса служит интерфейс Windows. При разработке этой операционной системы специалисты широко использовали возможные графические средства: рисунки, специальные значки, цветовое оформление, разнообразные начертания шрифтов, дизайн экрана и др. Графический интерфейс Windows позволяет более оперативно задавать команды операционной системы, запускать программы, выбирать файлы, кнопки, пункты меню.

Графический интерфейс пользователя (ГИП) является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя. К ГИП предъявляются высокие требования как с чисто инженерной, так и с художественной стороны, при его разработке ориентируются на возможности человека.

Наиболее часто ГИП реализуется в интерактивном режиме работы пользователя для программных продуктов, функционирующих в среде Windows, и строится в виде системы ниспадающих меню с использованием в качестве средства манипуляции мыши и клавиатуры. Работа пользователя осуществляется с экранными формами, содержащими объекты, панели инструментов с пиктограммами режимов и команд обработки.

Стандартный ГИП должен отвечать ряду требований:

* Поддерживать информационную технологию работы пользователя с программным продуктом – содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки, расположенные в естественной последовательности использования.
* Ориентироваться на конечного пользователя, который общается с программой на внешнем уровне взаимодействия.
* Удовлетворять правилу «шести» – в одну линейку меню включать не более шести понятий, каждое из которых содержит не более шести действий.
* Графические объекты сохраняют своё стандартизированное назначение и по возможности своё местоположение на экране.

2.5.3 Разработка интерфейса программы «Информационная система ГИБДД»

При разработке интерфейса программы «Информационная система ГИБДД» были учтены следующие показатели:

1. Интуитивность - управляющие элементы интерфейса должны быть удобными и заметными, вместе с тем они не должны отвлекать от основного содержания, за исключением случаев, когда управляющие элементы сами являются основным содержанием.
2. Цветовое оформление - используя цвет в передаче информации на дисплее, следует учитывать:

- Чувствительность глаза различна к разным участкам спектра. В условиях дневного освещения чувствительность глаза наиболее высока к желтым и зеленым лучам. По данным экспериментальных исследований, зеленый цвет на экране дает несколько лучшие результаты по скорости и точности чтения, чем оранжево-желтый;

- При длительном цветовом воздействии на глаз снижается его чувствительность к данному цвету. Наибольшее падение чувствительности наблюдается для сине-фиолетового цвета, наименьшее - для зеленого и желтого, т.е. синий цвет наиболее утомляет глаз;

- Светлые цвета на темном фоне кажутся приближенными к зрителю, а темные на светлом – удаленными.

В результате был разработан интерфейс, обладающий хорошим цветовым сочетанием, удобством расположения всех кнопок, выполняющих функции программы и позволяющий при запуске программы перейти на необходимый модуль программы, не прилагая к этому особых усилий.

2.6 Тестирование программы

2.6.1 Понятие тестирования программных средств

Многие организации, занимающиеся созданием программного обеспечения, до 50% средств, выделенных на разработку программ, тратят на тестирование, что составляет миллиарды долларов по всему миру в целом. И все же, несмотря на громадные капиталовложения, знаний о сути тестирования явно не хватает и большинство программных продуктов неприемлемо ненадежно даже после «основательного тестирования».

О состоянии дел лучше всего свидетельствует тот факт, что большинство людей, работающих в области обработки данных, даже не может правильно определить слово «тестирование», и это на самом деле главная причина неудач.

«Тестирование – процесс, подтверждающий правильность программы и демонстрирующий, что ошибок в программе нет. Основной недостаток подобного определения заключается в том, что оно совершенно неправильно; фактически это почти определение антонима слова «тестирование». Читатель с некоторым опытом программирования уже, вероятно, понимает, что невозможно продемонстрировать отсутствие ошибок в программе. Поэтому определение описывает невыполнимую задачу, а так как тестирование зачастую все же выполняется с успехом, по крайней мере с некоторым успехом, то такое определение логически некорректно. Правильное определение тестирования таково: Тестирование – процесс выполнения программы с намерением найти ошибки.

Невозможно гарантировать отсутствие ошибок в нетривиальной программе; в лучшем случае можно попытаться показать наличие ошибок. Если программа правильно ведет себя для солидного набора тестов, нет основании утверждать, что в ней нет ошибок; со всей определенностью можно лишь утверждать, что не известно, когда эта программа не работает. Конечно, если есть причины считать данный набор тестов способным с большой вероятностью обнаружить все возможные ошибки, то можно говорить о некотором уровне уверенности в правильности программы, устанавливаемом этими тестами.

Психологические эксперименты показывают, что большинство людей, поставив цель (например, показать, что ошибок нет), ориентируется в своей деятельности на достижение этой цели. Тестовик подсознательно не позволит себе действовать против цели, т. е. подготовить тест, который выявил бы одну из оставшихся в программе ошибок. Поскольку мы все признаем, что совершенство в проектировании и кодировании любой программы недостижимо и поэтому каждая программа содержит некоторое количество ошибок, самым плодотворным применением тестирования будет найти некоторые из них. Если мы хотим добиться этого и избежать психологического барьера, мешающего нам действовать против поставленной цели, наша цель должна состоять в том, чтобы найти как можно больше ошибок. Сформулируем основополагающий вывод:

Если ваша цель – показать отсутствие ошибок, вы. их найдете не слишком много. Если же ваша цель – показать наличие ошибок, вы найдете значительную их часть. Надежность невозможно внести в программу в результате тестирования, она определяется правильностью этапов проектирования. Наилучшее решение проблемы надежности – с самого начала не допускать ошибок в программе. Однако вероятность того, что удастся безупречно спроектировать большую программу, бесконечно мала. Роль тестирования состоит как раз в том, чтобы определить местонахождение немногочисленных ошибок, оставшихся в хорошо спроектированной программе. Попытки с помощью тестирования достичь надежности плохо спроектированной программы совершенно бесплодны.

2.6.2 Виды и методы тестирования

Тестирование программы методом «черного ящика».

Одним из способов изучения поставленного вопроса является исследование стратегии тестирования, называемой стратегией «черного ящика», тестированием с управлением по данным или тестированием с управлением по входу-выходу. При использовании этой стратегии программа рассматривается как «черный ящик». Иными словами, такое тестирование имеет целью выяснение обстоятельств, в которых поведение программы не соответствует спецификации. Тестовые же данные используются только в соответствии со спецификацией программы (т. е. без учета знаний о ее внутренней структуре). При таком подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных. Если такое испытание представляется сложным, то еще сложнее создать исчерпывающий тест для большой программы.

Построение исчерпывающего входного теста невозможно. Это подтверждается двумя аргументами: во-первых, нельзя создать тест, гарантирующий отсутствие ошибок; во-вторых, разработка таких тестов противоречит экономическим требованиям.

Тестирование программы методом «белого ящика»

Стратегия «белого ящика», или стратегия тестирования, управляемого логикой программы, позволяет исследовать внутреннюю структуру программы, В этом случае тестирующий продукт получает тестовые данные путем анализа логики программы. Сравним способ построения тестов при данной стратегии с исчерпывающим входным тестированием стратегии «черного ящика». Непосвященному может показаться, что достаточно построить такой набор тестов, в котором каждый оператор исполняется хотя бы один раз; нетрудно показать, что это неверно. Не вдаваясь в детали, укажем лишь, что исчерпывающему входному тестированию может быть поставлено в соответствие исчерпывающее тестирование маршрутов. Подразумевается, что программа проверена полностью, если с помощью тестов удается осуществить выполнение программы по всем возможным маршрутам ее потока (графа) передач управления. Последнее утверждение имеет два слабых пункта. Первый из них состоит в том, что число не повторяющих друг друга маршрутов в программе — астрономическое. Второй слабый пункт утверждения заключается в том, что, хотя исчерпывающее тестирование маршрутов является полным тестом и хотя каждый маршрут программы может быть проверен, сама программа будет содержать ошибки. Это объясняется следующим образом. Во-первых, исчерпывающее тестирование маршрутов не может дать гарантии того, что программа соответствует описанию. Например, вместо требуемой программы сортировки по возрастанию случайно была написана программа сортировки по убыванию. В этом случае ценность тестирования маршрутов невелика, поскольку после тестирования в программе окажется одна ошибка, т. е. программа неверна.

Во-вторых, программа может быть неверной в силу того, что пропущены некоторые маршруты. Исчерпывающее тестирование маршрутов не обнаружит их отсутствия.

Восходящее тестирование.

При восходящем подходе программа собирается и тестируется «снизу вверх». Только модули самого нижнего уровня тестируются изолированно, автономно. После того как тестирование этих модулей завершено, вызов их должен быть так же надежен, как вызов встроенной функции языка или оператор присваивания. Затем тестируются модули, непосредственно вызывающие уже проверенные. Эти модули более высокого уровня тестируются не автономно, а вместе с уже проверенными модулями более низкого уровня. Процесс повторяется до тех пор, пока не будет достигнута вершина. Здесь завершается и тестирование модулей, и тестирование сопряжений программы. При восходящем тестировании для каждого модуля необходим драйвер: нужно подавать тесты в соответствии с сопряжением тестируемого модуля. Одно из возможных решений — написать для каждого модуля небольшую ведущую программу. Тестовые данные представляются как «встроенные» непосредственно в эту программу переменные и структуры данных, и она многократно вызывает тестируемый модуль, с каждым вызовом передавая ему новые тестовые данные. Имеется и лучшее решение: воспользоваться программой тестирования модулей - это инструмент тестирования, позволяющий описывать тесты на специальном языке и избавляющий от необходимости писать драйверы. Здесь отсутствуют проблемы, связанные с невозможностью или трудностью создания всех тестовых ситуаций, характерные для нисходящего тестирования. Драйвер как средство тестирования применяется непосредственно к тому модулю, который тестируется, где нет промежуточных модулей, которые следует принимать во внимание. Не существует также и трудностей с незавершенностью тестирования одного модуля при переходе к тестированию другого, потому что при восходящем тестировании с применением нескольких версий заглушки нет сложностей с представлением тестовых данных.

Нисходящее тестирование.

Нисходящее тестирование (называемое также нисходящей разработкой) не является полной противоположностью восходящему, но в первом приближении может рассматриваться как таковое. При нисходящем подходе программа собирается и тестируется «сверху вниз». Изолированно тестируется только головной модуль. После того как тестирование этого модуля завершено, с ним соединяются (на пример, редактором связей) один за другим модули, непосредственно вызываемые им, и тестируется полученная комбинация. Процесс повторяется до тех пор, пока не будут собраны и проверены все модули. Нисходящий метод имеет как достоинства, так и недостатки по сравнению с восходящим. Самое значительное достоинство — то, что этот метод совмещает тестирование модуля, тестирование сопряжений и частично тестирование внешних функций. С этим же связано другое его достоинство: когда модули ввода-вывода уже подключены, тесты можно готовить в удобном виде. Нисходящий подход выгоден также в том случае, когда есть сомнения относительно осуществимости программы в целом или когда в проекте программы могут оказаться серьезные дефекты. Преимуществом нисходящего подхода очень часто считают отсутствие необходимости в драйверах.

Нисходящий метод тестирования имеет, к сожалению, некоторые недостатки. Основным из них является то, что модуль редко тестируется досконально сразу после его подключения. Дело в том, что основательное тестирование некоторых модулей может потребовать крайне изощренных заглушек. Программист часто решает не тратить массу времени на их программирование, а вместо этого пишет простые заглушки и проверяет лишь часть условий в модуле. Второй тонкий недостаток нисходящего подхода состоит в том, что он может породить веру в возможность начать программирование и тестирование верхнего уровня программы до того, как вся программа будет полностью спроектирована. Эта идея на первый взгляд кажется экономичной, но обычно дело обстоит совсем наоборот. Большинство опытных проектировщиков признаёт, что проектирование программы — процесс итеративный. Редко первый проект оказывается совершенным. Нормальный стиль проектирования структуры программы предполагает по окончании проектирования нижних уровней вернуться назад и подправить верхний уровень, внеся в него некоторые усовершенствования или исправляя ошибки, либо иногда даже выбросить проект и начать все сначала, потому что разработчик внезапно увидел лучший подход. Если же головная часть программы уже запрограммирована и оттестирована, то возникает серьезное сопротивление любым улучшениям ее структуры.

2.6.3 Процесс и результат тестирования

Объектом испытаний в дипломном проекте является программа база данных «Информационная система ГИБДД».

Основной метод для обнаружения ошибок в программе – проведение испытаний. Испытание представляет собой процесс выполнения программы с целью проверки правильности результатов ее работы и соответствия заданным характеристикам. Процесс испытаний включает в себя преднамеренное конструирование сложных наборов входных данных, создающих наибольшие возможности для отказа программы.

Испытания следует проводить в программной среде предусмотренной программой, т. е. язык на котором написана программа с использованием сложных наборов входных данных, создающих наибольшие возможности для отказа программы.

Программа база данных «Информационная система ГИБДД» была протестирована методом «чёрного ящика».

Тест №1. Запуск программы.

Результат: Программа запускается без ошибок.

Тест №2. Проверка работы меню (выбирается любой пункт меню).

Результат: Открытие необходимого окна.

Тест №3. Проверка работы с базой данных (возможность просматривания базы данных, добавление, изменение и удаление записей).

Результат: Работа с базой данных выполняется правильно.

Тест №4. Проверка кнопки для выполнения запроса.

Результат: Запрос выполняется верно.

Тест №5. Проверка кнопки для открытия «Справки».

Результат: Открытие необходимого окна.

Тест №6. Проверка кнопки для просмотра информации «О программе»

Результат: Открытие необходимого окна.

Тест №7. Проверка пункта меню «Выход».

Результат: Пункт меню «Выход» работает без ошибок.

Все компоненты работают исправно. Все функции в программе

отвечают поставленным требованиям. В случае неправильных действий программа выдает предусмотренные в ней сообщения. Тест пройден успешно.

2.7 Программная документация

2.7.1 Руководство программиста

Назначение и условия применения программы:

1. приложение «Информационная система ГИБДД» представляет собой программу для управления базой данных;
2. программа полностью автоматизирует труд оператора ЭВМ: автоматически заносит вводимые пользователем данные в таблицы базы данных, изменяет структуру таблиц (добавление/удаление записей)
3. программа хранит все сведения об автомобилях и их владельцев: личные данные владельца и автомобиля.
4. также программа хранит сведения о дорожно-транспортном происшествии: место ДТП, ущерб, количество пострадавших и т. д.

Загрузка программы возможна с дискеты, жесткого диска, CD диска, открыванием файла с именем «БД ГИБДД.exe».

Программа использует БД Paradox, для доступа к данным используется BDE версии 5.

Для установки программы необходимо:

* 1. Скопировать каталог программы на жесткий диск.
  2. Установить BDE. Для этого нужно запустить BDE Administrator   
     (\Program Files\Common Files\Borland Shared\BDE\bdeadmin.exe). Затем создать псевдоним базы данных (Object\New). В выпадающем списке выбрать драйвер БД – STANDARD и переименовать его на STANDARD2. Убедиться, что Default Driver = PARADOX (при необходимости выбрать из выпадающего списка).
  3. Записать изменения (Object\Apply).
  4. После всего этого запустить файл «Информационная система ГИБДД.exe».

2.7.2 Руководство оператора

Для запуска программы, необходимо запустить файл с именем «Информационная система ГИБДД.exe».

После открытия этого файла появляется главная форма «Информационная система ГИБДД» (рисунок 1). Здесь оператор может выбрать базу данных, с которой хочет работать, вызвать справку и информацию «О программе», а также выйти из программы.

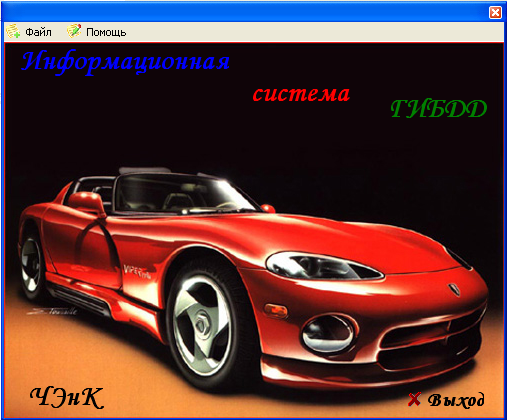


Рисунок 1 Окно главной формы.

Если оператор нажмет на вкладку «Справочник номеров частных владельцев» (которая находится в меню «Файл» на левой стороне окна), то появится окно (рисунок 2) в котором хранится база данных о номерах выданных частным владельцам. В этом окне имеется фильтрация по техосмотру, диапазон (поиск данных по дате) и при вводе номера автомобиля появляется вся информация о владельце и автомобиле (состоит из вкладок).

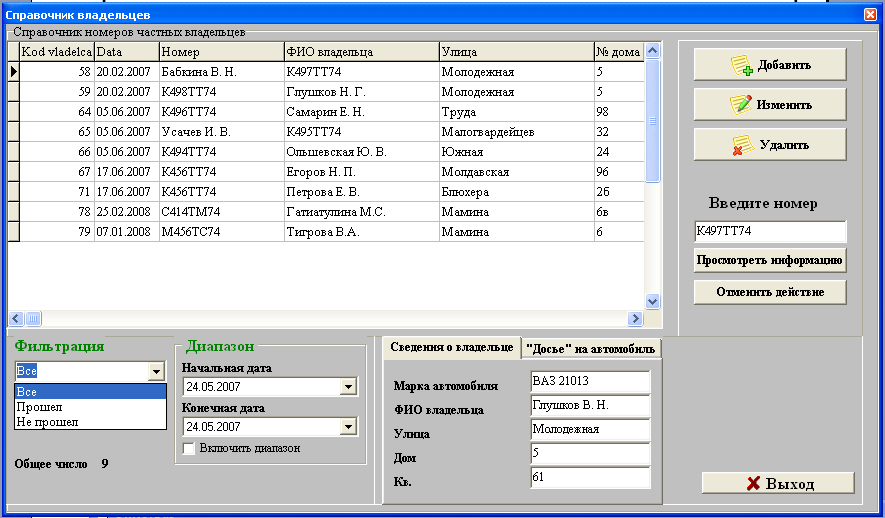


Рисунок 2. Справочника номеров частных владельцев.

Если оператор нажмет на вкладку «Справочник организации», то появится окно (рисунок 3), в котором хранится база данных о номерах выданных организациям. В этом окне так же имеется фильтрация по техосмотру, диапазон (поиск данных по дате) и при вводе номера автомобиля появляется вся информация о владельце и автомобиле (состоит из вкладок).

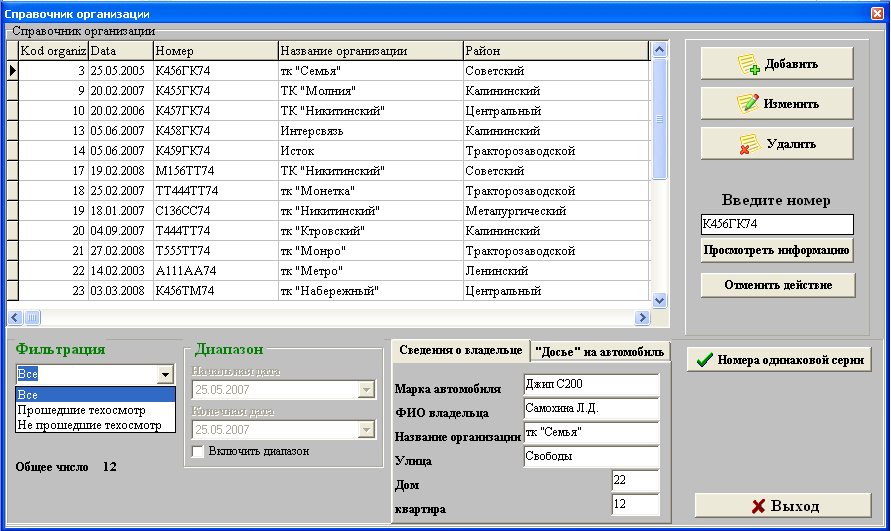


Рисунок 3. Справочник номеров организации.

При выборе вкладки «Регистрация ДТП» появится база данных регистрации ДТП (рисунок 4). В этом окне имеется краткое содержание ДТП, а так же имеется фильтрация и диапазон.

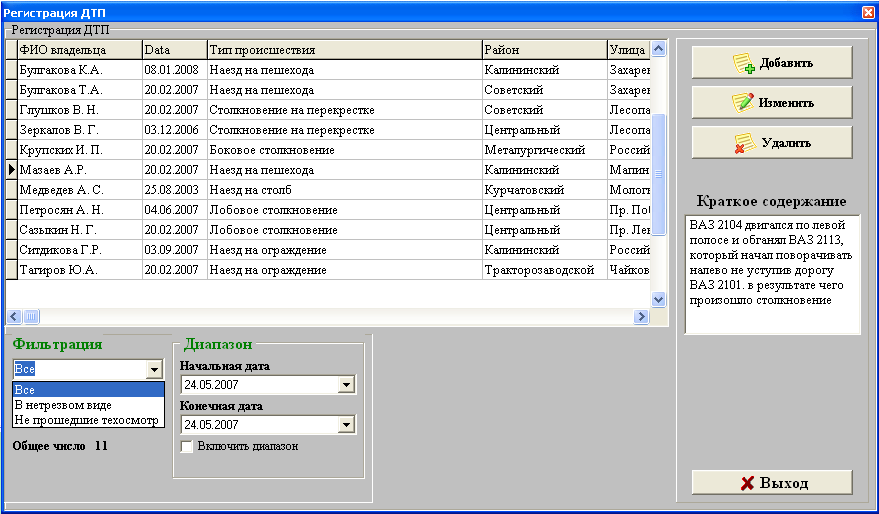


Рисунок 4. Регистрация ДТП.

При выборе вкладки «Угон автомобилей» появится окно базы данных угнанных автомобилей (рисунок 5). Здесь имеется фильтрация и диапазон.

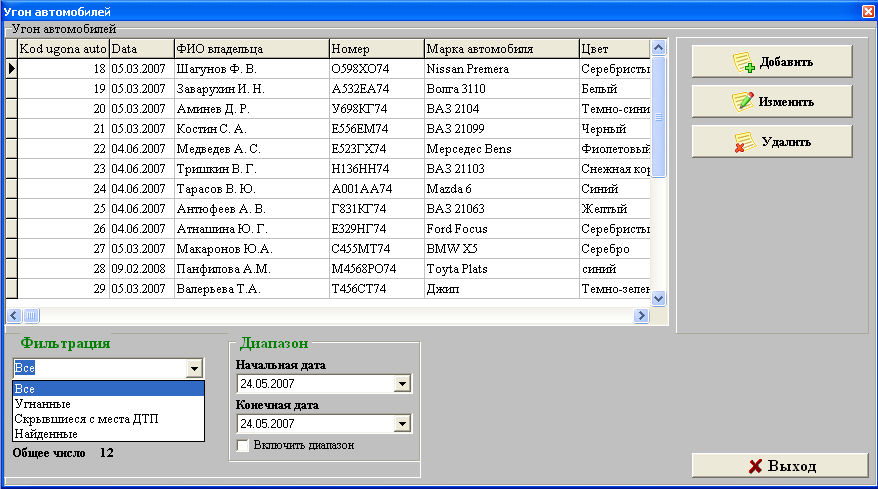


Рисунок 5. Угон автомобилей.

В каждой базе данных есть кнопки «Добавить» (добавление записи в таблицу), «Изменить» (изменение записи в таблице), «Удалить» (удалить запись в таблице), «Выход» (выйти из данной базы данных). При нажатии кнопки «Добавить» появится окно добавлении записи (рисунок 6). Оно будет пустым. Если пользователь начнет сохранять запись, пропустив хоть одну строчку, то вылезет окно «Заполните пустое поле» (рисунок 7). Пользователь должен заполнить его или выйти из этой формы, нажав выход. Аналогично кнопка «Изменить», только при появлении окна будут заполненные строчки (рисунок 7). При нажатии кнопки «Удалить» появится окошко действительно ли вы хотите удалить запись в таблице (рисунок 9).

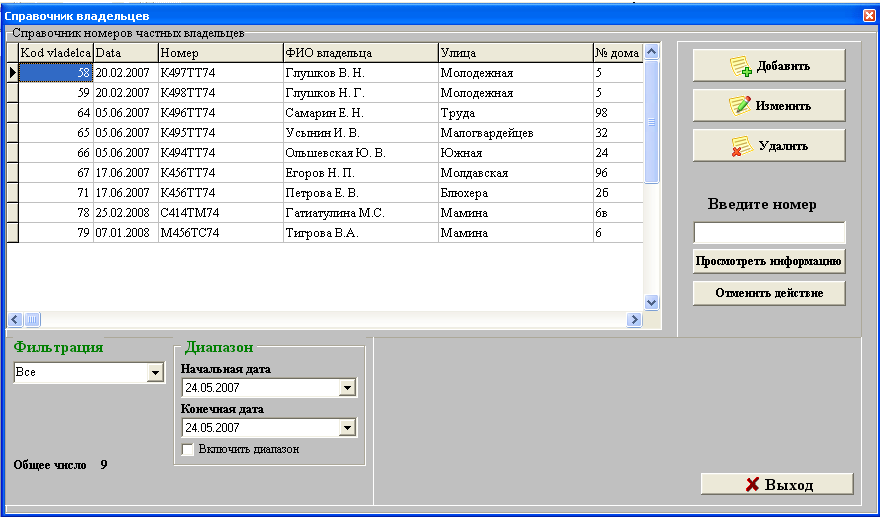


Рисунок 6. Заполнение таблицы.

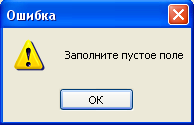


Рисунок 7. Предупреждение.

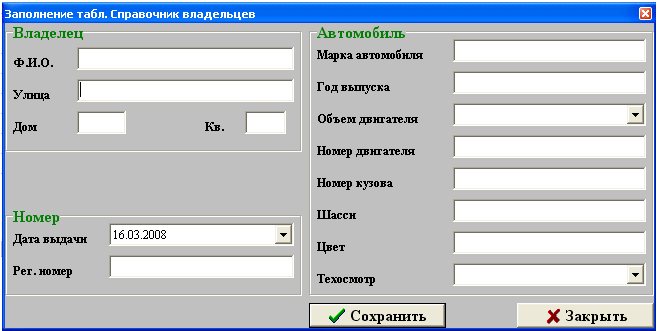


Рисунок 8. Изменение записи в таблице.

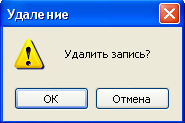


Рисунок 9. Предупреждение.

При нажатии на кнопку «О программе» в главной форме во вкладке «Помощь» (она находится на верхней части формы), появится окно, в которой указан разработчик программы «Информационная система ГИБДД» (рисунок 10).



Рисунок 10. О программе.

При нажатии на кнопку «Справка» в главной форме во вкладке «Помощь» (она находится на верхней части формы), появится окно, в которой указано как пользоваться программой «Информационная система ГИБДД» (рисунок 11).

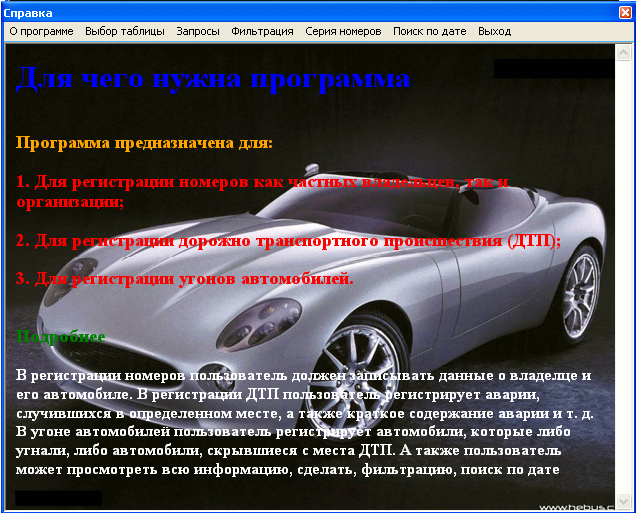


Рисунок 11. Справка.

При выходе из программы (при нажатии на кнопку «Выход») появится окно с подтверждением «Вы действительно хотите выйти из программы?» (рисунок 12).

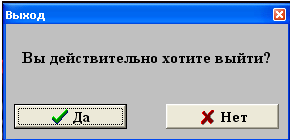


Рисунок 12. Подтверждение выхода из программы.

3. Эксплуатация программных средств

Рассмотрим структуру вычислительного центра на примере отдела автоматизированной системы учета ТГК №10

3.1 Эксплуатация программного обеспечения

Для поддержки информационной технологии выделяется три класса программных продуктов:

* Системное программное обеспечение;
* Пакеты прикладных программ;
* Инструментарий технологии программирования;

Системное программное обеспечение направлено на:

* Создание операционной среды функционирования других программ;
* Обеспечение надежной и эффективной работы самого компьютера и вычислительной сети;
* Проведение диагностики и профилактики аппаратуры компьютера и вычислительных сетей;
* Выполнение вспомогательных технологических процессов (копирование, архивирование, восстановление файлов программ и баз данных и т.д.).

Программные продукты данного класса носят общий характер применения, независимо от спецификации предметной области. К ним предъявляются высокие требования по надежности и технологичности работы, удобству и эффективности использования.

Пакеты прикладных программ служат программным инструментарием решения функциональных задач и являются самым многочисленным классом программных продуктов. В данный класс входят программные продукты, выполняющие обработку информации различных предметных областей.

Для IBM PC разработаны и используются сотни тысяч различных прикладных программ для различных применений. Наиболее широко применяются программы:

- подготовки тестов (документов) на компьютере – редакторы тестов;

- подготовки документов типографического качества – издательские системы;

- обработки табличных данных – табличные процессы;

- обработки массивов информации – системы управления базами данных.

Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированные программные продукты, которые являются инструментальными средствами разработчика. Программные продукты данного класса поддерживают все технологические этапы процесса проектирования, программирования (кодирования), отладки и тестирования создаваемых программ. Пользователями технологии программирования являются системные и прикладные программисты.

Данная программа база данных «Информационная система ГИБДД»» относится к прикладным программам.

Необходимость архивации файлов.

При эксплуатации компьютера по самым разным причинам возможны порча или потеря информации на магнитных дисках. Это может произойти из-за физической порчи магнитного диска, неправильной корректировки или случайного уничтожения файлов, разрушения информации компьютерным вирусом и т. д. Для этого, чтобы уменьшить потери в таких случаях, следует иметь архивные копии используемых файлов и систематически обновлять копии изменяемых файлов.

Для создания архивов употребляются специализированные программы. Их можно разделить на два класса: программы – упаковщики (архиваторы) и программы резервного копирования.

Программы – упаковщики позволяют за счет специальных методов сжатия информации создавать копии файлов меньшего размера и объединять копии нескольких файлов в один архивный файл. В большинстве случаев значительно удобнее хранить на дискетах, а иногда и на кассетах для стримера, файлы, предварительно сжатые программами – упаковщиками.

Программы резервного копирования предназначены для копирования информации с жесткого диска на кассеты стримера или дискеты. Из этих программ широко используются Norton Backup ( для DOS и для Windows), FastBack Plus и др.

Защита от компьютерных вирусов.

Компьютерный вирус – это специально написанная небольшая по размерам программа, которая может «приписывать» себя к другим программам (т. е. «заражать» их), а также выполнять различные нежелательные действия на компьютере. Программа, внутри которой находится вирус, называется «зараженной». Когда такая программа начинает работу, то сначала управление получает вирус. Вирус находит и «заражает» другие программы, а также выполняет какие-нибудь вредные действия (портит файлы или таблицу размещения файлов на диске, «засоряет» оперативную память и т. д.). После того как вирус выполняет нужные ему действия, он передает управление той программе, в которой он находится, и она работает так же, как обычно. Тем самым внешне работа зараженной программы выглядит так же, как и незараженной.

Таким образом, если не предпринимать мер по защите от вируса, то последствия заражения компьютера могут быть очень серьезными.

Виды вирусов.

- вирусы, меняющие файловую систему – обычно называются DIR и прячут свое тело в некоторый участок диска и помечают его в таблице размещения файлов как коней файла;

- «невидимые» самомодифицирующие вирусы – предотвращают свое обнаружение тем, что перехватывают обращения DOS к зараженным файлам и областям диска и выдают их в исходном (незараженном) виде;

- самомодифицирующие вирусы – для того, чтобы укрыться от обнаружения применяют модификацию своего тела. В теле подобного вируса не имеется ни одной постоянной цепочки байтов, по которой можно было бы идентифицировать вирус.

Основные методы защиты от компьютерных вирусов.

Для защиты от вирусов можно использовать:

- общие средства защиты информации;

- профилактические меры;

- специализированные программы для защиты от вирусов.

Общие средства защиты информации полезны не только для защиты от вируса. Имеются две основные разновидности этих средств:

- копирование информации – создание копий файлов и системных областей дисков;

- разграничение доступа – предотвращает несанкционированное использование информации.

Несмотря на то, что общие средства защиты информации очень важны для защиты от вирусов, все же их недостаточно. Необходимо и применение специализированных программ. Существуют следующие программы:

- программы-детекторы – позволяют обнаруживать файлы, зараженные одним из нескольких известных вирусов;

- программы-доктора, или фаги – «лечат» зараженные программы или диски, «выкусывая» из зараженных программ тело вируса;

- программы-ревизоры – сначала запоминают сведения о состоянии программ и системных областей дисков, а затем сравнивают их состояние с исходным. При выявлении несоответствий об этом сообщается пользователю;

- доктора-ревизоры – программы, которые не только обнаруживают изменения в файлах и системных областях дисков, но и могут в случае изменений автоматически вернуть их в исходное состояние;

- программы-фильтры – располагаются резидентно в оперативной памяти компьютера и перехватывают те обращения к операционной системе, которые используются вирусами для размножения и нанесения вреда, и сообщают о них пользователю;

- программы-вакцины, или иммунизаторы – модифицируют программы и диски таким образом, что это не отражается на роботе программ, но тот вирус, от которого производится вакцинация, считает эти программы или диски уже зараженными.

Копирование файлов с жесткого диска.

Перед тем как записывать на жесткий диск какие-то программы, желательно скопировать его содержимое на дискеты. Эти дискеты можно будет использовать при повреждении файлов операционной системы на жестком диске. Пред копированием файлов с жесткого диска на дискеты целесообразно включить режим проверки записи на диск. Это позволит избежать создания неправильных копий файлов на дискетах. После того как все файлы с жесткого диска будут скопированы на дискеты, следует заклеить на этих дискетах прорезь защиты от записи, чтобы скопированные файлы не смогли быть случайно изменены или испорчены. Целесообразно также сделать копии этих дискет и хранить вторые экземпляры дискет отдельно от первых экземпляров, чтобы уменьшить вероятность порчи дискет при каком-либо несчастном случае.

Обслуживание дисков.

В ходе эксплуатации жесткого диска на нем могут образовываться «потерянные» участки, не принадлежащие ни одному из файлов и списку свободных участков диска, дефектные участки, ненужные файлы, которые сохраняются на диске только потому, что их позабыли стереть, и т. д. Поэтому необходимо периодически проводить обслуживание жесткого диска.

Если компьютер используется интенсивно, то процедуру обслуживания жесткого диска следует выполнять раз в 1 – 2 недели, а при менее интенсивной загрузке – раз в 1 – 2 месяца.

4. Экономический расчет программного продукта

4.1 Расчет себестоимости программного продукта «Информационная система ГИБДД»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №п\п | Наименование | Количество | Цена за ед-цу, руб. | Сумма, руб. |
| 1 | Ручка | 1 шт. | 10 | 10 |
| 2 | Бумага | 1 шт. | 130 | 130 |
| 3 | Заправка картриджа для принтера | 1 шт. | 180 | 180 |
| 4 | Тетрадь | 1 шт. | 20 | 20 |
|  | Итого |  |  | 340 |

Исходные данные:

1. Количество компьютеров – 1.

2. Количество дней работы над программой –.30

3. Коэффициент использования ПК – 0,8.

4. Продолжительность рабочей смены – 6ч.

5. Тариф за потребление электроэнергию – 1,53 руб.\кВ.ч.

1 Полезный фонд времени работы ПК.

Фп = t\* nпк\* T \* Ku, час,

где t – продолжительность одной смены работы на ПК –6 ч.

nпк – количество компьютеров – 1.

T – количество дней работы над программой – 30.

Ku – коэффициент использования ПК – 0,8.

Фп = 6\*1\*30\*0,8 = 144 ч.

2 Стоимость потребляемой энергии.

Uэп=P\*Фп\*Цруб,

где P – мощность потребляемая ПК - 0,5.

Фп – полезный фонд времени работы ПК

Цруб – тариф электроэнергии 1,53 руб.

Uэп=0,5\*144\*1,53=110,16 руб.

3 Стоимость эксплуатационных материалов.

Материальные затраты – всего:

Uмз = Uэл + Uэм руб,

где Uэл – стоимость потребляемой электроэнергии –110,16 руб.

Uэм – стоимость эксплуатационных материалов – 340 руб.

Uмз = 110,16 + 340= 450,16 руб

4 Оплата труда.

Формируется из основной и дополнительной оплаты труда.

4.1 Основная оплата труда за время работы над программой

Uопл.осн = Т\*З руб,

Где Т – количество дней работы над программой – 30,

З – заработная плата программиста за 1 день – 78,09 руб.

Uопл.осн = 30\*78,09=2342,7 руб

Дополнительная зар. плата.

Uопл.д = 0.4 \* Uопл.осн руб, где

Uопл.осн – основная зар. плата за время работы над программой – 2342,7 руб.

Uопл.д = 0.4 \* 2342,7 = 937,08 руб,

Оплата труда – всего

Uопл.тр. = Uопл.осн + Uопл.д руб,

где Uопл.осн – основная зар. плата за время работы над программой – 2342,7 руб.

Uопл.д. – дополнительная заработная плата – 937,08 руб.

Uопл.тр. = 2342,7+937,08=3279,78 руб.

4.2 Отчисления на социальные нужды отражают обязательные отчисления по установленным законодательством нормам, органам социального страхования, пенсионного фонда, государственного фонда занятости и медицинского страхования

Uсн = (Uсн. / 100) \* Uопл.тр руб,

где Uсн. – норма отчисления на социальные нужды 26 %

Uопл.– оплата труда – 3279,78 руб.

Uсн. = (26 / 100) \* 3279,78= 852,7 руб.

4.3 Амортизационные отчисления:

Uа= (Ha / 100) \* Соф, руб,

где Ha – норма амортизационных отчислений на полное восстановление – 10,7%

Соф – стоимость основных фондов – 2500 руб.

Uа = (10,7 / 100) \* 2500 = 267,5 руб.

5 Прочие затраты

Прочих затрат нет.

6 Общие издержки работы над программой.

Uо = Uмз + Uопл.тр. + Uсн + Uа + Uпр руб

Uо = 450,16+3279,78+852,7+267,5+0=4850,14 руб.

7 Структура себестоимости программы.

Таблица 4.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование статей | Издержки за время работы, руб. | Структура себестоимости, % |
| 1 | Материальные затраты | 450,16 | 9,3 |
| 2 | Оплата труда | 3279,78 | 67,6 |
| 3 | Отчисления на социальные нужды | 852,7 | 17,6 |
| 4 | Амортизационные отчисления | 267,5 | 5,5 |
| 5 | Прочие затраты | 0 | 0 |
|  | Итого | 4850,14 | 100 |

8 Расчет структуры себестоимости:



Вывод: Полная себестоимость программы «Информационная система ГИБДД» составляет 4850,14 руб.

5. Техника безопасности

5.1 Общие требования по техники безопасности

При эксплуатации ПК на работника могут оказывать действия, следующие опасные и вредные производственные факторы:

* Повышенный уровень электромагнитных излучений;
* Повышенный уровень статического электричества;
* Повышенная ионизация воздуха;
* Статические физические перегрузки;
* Перенапряжение зрительных анализаторов;

Работник обязан:

- Выполнять только ту работу, которая определена его должностной инструкцией.

- Содержать в чистоте рабочее место.

- Соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности.

- Соблюдать меры пожарной безопасности.

К работе с персональным компьютером допускаются лица: имеющие персональные навыки, изучившие руководство по эксплуатации персонального компьютера и знающие порядок включения и отключения электронных устройств; прошедшие вводный инструктаж, а также инструктаж по безопасности труда непосредственно на рабочем месте.

Профессиональные пользователи должны проходить обязательные предварительные (при поступлении на работу) и периодические медосмотры.

К непосредственной работе с персональным компьютером допускаются лица, не имеющие медицинских противопоказаний.

Следует иметь в виду, что:

* Расстояние между рабочими столами с компьютерами в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора.
* Площадь на одно рабочее место ПК должна составлять не менее 6,0 кв. м, а объем - не менее 20,0 куб. м.
* Верхний край экрана следует располагать на уровне глаз или чуть ниже.
* Оптимальное расстояние от глаз до экрана 600-700 мм, но не ближе 500 мм.
* Высота клавиатуры должна быть отрегулирована так, чтобы кисти рук держались прямо (возможно применение подставок для кистей рук).
* С целью снижения статического напряжения мышц шейно - плечевой области и спины для предупреждения развития утомления необходимо, чтобы рабочий стул кресло) позволял изменять позу, был подъемно - поворотным и регулируемым по высоте и углам наклона сидения и спинки от переднего сидения.
* В производственных помещениях, в которых работа на ПК является основной (диспетчерские, операторские, расчетные, кабинеты и посты управления, залы вычислительной техники и т.д.) должны обеспечиваться оптимальные параметры микроклимата. Для нормализации аэроионного фактора помещений с компьютерами необходимо использовать устройства автоматического регулирования ионного режима воздушной среды.
* Рабочие места с ПК при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, следует изолировать друг от друга перегородками высотой 1,5-2,0 м.
* В помещениях с ПК ежедневно должна проводится влажная уборка.
* Помещения с ПК должны быть оснащены аптечкой первой помощи и углекислотными огнетушителями.
* Кабели компьютера должны располагаться так, чтобы их нельзя было повредить неосторожным движением.
* Для предотвращения образовании и защиты от статического электричества, в помещении, где установлен компьютер, необходимо применять нейтрализаторы и увлажнители. Должна быть эффективная вентиляция и поддерживаться относительная влажность воздуха на уровне 40-60 % (можно разместить вблизи компьютера цветы или аквариум).

Для обеспечения оптимальной работоспособности и сохранения здоровья профессиональных пользователей, на протяжении рабочей смены должны устанавливаться регламентированные перерывы. Продолжительность непрерывной работы с персональным компьютером без регламентированного перерыва не должна превышать 2 часов.

При работе с персональным компьютером в ночную смену (c 22 до 6 часов), независимо от категории и вида трудовой деятельности, продолжительность регламентированных перерывов должна увеличиваться на 60 мин.

Невыполнение требований настоящей инструкции является нарушением производственной дисциплины. Виновные в этом несут ответственность, в порядке, установленном действующим законодательством.

5.2 Техника безопасности при работе на компьютере

5.2.1 Требования безопасности перед началом работы на ПК

Работник при работе с ПК обязан:

- Осмотреть и привести в порядок рабочее место.

- Отрегулировать освещенность на рабочем месте, убедиться в достаточности освещенности, отсутствии отражений на экране.

- Проверить правильность подключения оборудования в электросеть.

- Проверить исправность проводящих проводов и отсутствие оголённых участков проводов.

- Убедиться в наличии защитного заземления.

- Протереть салфеткой поверхность экрана и защитного фильтра.

- Убедиться в отсутствии дискет в дисководах ПК.

- Проверить правильность установки стола, стула, положения оборудования, угла наклона экрана, положение клавиатуры, положение мыши на специальном коврике, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения длительных напряжений тела.

Работнику запрещается приступать к работе с ПК при:

- Отсутствии защитного экранного фильтра класса «полная защита».

- Отсутствии специальной вилки с подключением заземления.

- Обнаружении неисправности оборудования.

- При размещении ПК в ряд на расстоянии менее 1,2 м, при расположении рабочих мест с компьютерами в колонку на расстоянии менее 2,0 м, при рядном расположении дисплеев экранами друг к другу.

- Работнику запрещается протирать влажной или мокрой салфеткой электрооборудование, которое находится под напряжением. Влажную или любую другую уборку производить только при отключенном оборудовании.

- Работник обязан сообщить руководителю об обнаруженной неисправности оборудования. Не пользоваться неисправным оборудованием. Приступить к работе только после устранения нарушений в работе или неисправностей оборудования.

- Работник не должен разрешать работать лицам, не имеющим допуска к работе с опасным оборудованием или с ПК.

5.2.2 Требования безопасности во время работы с ПК

Работник организации во время работы обязан:

* Выполнять ту работу, которая определена его должностной инструкцией. Которая ему была поручена, и по которой он был проинструктирован.

- В течение всего рабочего времени содержать в порядке и чистоте рабочее место.

* Держать открытыми вентиляционные отверстия, которыми оборудованы приборы и ПК.
* Не загромождать оборудование посторонними предметами, которые снижают теплоотдачу.
* При необходимости прекращения работы на некоторое время корректно закрыть все активные задачи.
* Выполнять санитарные нормы и соблюдать режимы работы и отдыха.
* Соблюдать правила эксплуатации электрооборудования или другого оборудования в соответствии с инструкциями по эксплуатации.
* При работе с текстовой информацией выбирать наиболее физиологичный режим представления черных символов на белом фоне.
* Соблюдать расстояние от глаз до экрана в пределах 60-70 см, но не ближе 50 см, с учетом размеров алфавитно-цифровых знаков и символов.
* Во всех случаях обнаружения обрыва проводов питания, неисправности заземления и других повреждений электрооборудования, появления запаха гари немедленно отключить питание и сообщать об аварийной ситуации руководителю и дежурному электромеханику.
* При любых случаях сбоя в работе технического оборудования или ПО немедленно вызвать специалиста.
* В случаях появления рези в глазах, резком ухудшении видимости – невозможности сфокусировать взгляд или навести его на резкость, появлении боли в пальцах и кистях рук, усилении сердцебиения немедленно покинуть рабочее место, сообщить руководителю.
* Не приступать к работе на ПК до устранения неисправности.
* При получении травм или при внезапном заболевании немедленно известить своего руководителя, организовать первую доврачебную помощь или вызвать скорую медицинскую помощь по телефону 03.
* При обнаружении человека, попавшего под напряжение, немедленно отключить электропитание и освободить его от действия тока, оказать доврачебную помощь или вызвать скорую медицинскую помощь по телефону 03.

Работнику при работе на ПК запрещается:

* Касаться одновременно экрана монитора и клавиатуры.
* Прикасаться к задней панели системного блока при включенном питании.
* Переключать разъёмы интерфейсных кабелей периферийных устройств при включенном питании.
* Допускать попадание влаги на поверхность системного блока, монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств.
* Производить самостоятельное вскрытие и ремонт оборудования.
* Работник обязан отключить ПК от электросети:

а) при обнаружении неисправности,

б) при внезапном снятии напряжения электросети,

в) во время чистки и уборки оборудования.

# 5.2.3 Требования безопасности после окончания работы с ПК

Работник обязан соблюдать следующую последовательность выключения ПК:

- Произвести закрытие всех активных задач.

- Выполнить парковку считывающей головки жесткого диска

- Убедиться, что в дисководе нет дискет.

- Выключить питание системного блока.

- Выключить питание всех периферийных устройств.

- Отключить блок питания.

- Работник обязан осмотреть и привести в порядок рабочее место и выполнить несколько упражнений для глаз и пальцев рук .

- Работник обязан по окончанию работы или, уходя с работы, вынуть исправную вилку из исправной розетки.

Заключение

В данном дипломном проекте была спроектирована, запрограммирована и протестирована программа «Информационная система ГИБДД». Сначала был проведен анализ предметной области, на основании чего был сделан вывод, что лучшим способом хранения данных в данном случае является база данных, затем был разработан алгоритм действия программы.

После этого был разработан пользовательский интерфейс, который позволит пользователям без труда работать с данной программой, поскольку при разработке были использованы общепринятые стандарты (например, стандартный для ОС Windows, оконный интерфейс).

Для программной реализации данного проекта были выбраны база данных Paradox (для хранения данных) и мощный язык программирования Delphi 7.

Данная программа снабжена всей необходимой для установки работы документацией.

Таким образом, поставленная передо мною задача дипломного проекта, полностью выполнена и соответствует техническому заданию.

Литература

1. Технологии разработки программного обеспечения: Учебник / С. Орлов. – СПб.: Питер, 2002. – 464 с.
2. Благодатских В. А. и др. Стандартизация разработки программных средств: Учеб. пособие / Благодатских В. А., Волнин В. А., Поскакалов К. Ф.; Под ред. Разумова О. С. – М.: Финансы и статистика, 2003. -288 с.
3. Программирование баз данных в Delphi 7. Учебный курс / Фаронов В. В.– СПб.: Питер, 2003. – 459 с.
4. Голицына О. Л., Максимов Н. В., Попов И. И. Базы данных: Учебное пособие. – М.: ФОРУМ: ИНФРА-М, 2003. 352 с.
5. С. И. Бобровский Delphi 7 учебный курс. – СПб.: Питер 2003-457с.;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

31

***63***

1. Рудаков А.В., «Технология разработки программных продуктов», Москва 2005г.;
2. Макарова Н. В.,«Информатика», Санкт-Петербург, Издательство «ПИТЕР», 2002 г.;