# ДИПЛОМНАЯ РАБОТА

# ТЕМА: ”Разработка базы данных для информатизации деятельности предприятия малого бизнеса Delphi 7.0”

# Оглавление

Введение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

**Глава 1. Аналитическая часть\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5**

#### 1.1. Структура ремонтно-строительной компании\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

#### 1.2. Постановка задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8

#### 1.3. Анализ необходимости внедрения автоматизированной

####  системы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

#### 1.4. Техническое задание на разработку.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_15

**Глава 2. Информационно-программная часть.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19**

#### Функциональные возможности системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

#### 2.2. Построение модели БД. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21

##### 2.3. Инфологическая модель БД\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22

##### 2.4. Выбор СУБД.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_28

##### 2.5. Построение датологической модели.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_33

**Глава 3.Организационно - технологическая часть \_\_\_\_\_\_\_\_\_\_\_\_\_43**

#### 3.1. Общая структура организации работ по

#### проектированию ПП.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_44

##### 3.1.1. Постановка задачи.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_44

##### Составление проекта.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_45

##### Алгоритмизация. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_46

##### 3.1.4. Программирование. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_49

##### 3.1.5. Препарацuя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_50

##### 3.1.6. Трансляция. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_50

##### 3.1.7. Отладка.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_51

##### 3.1.8. Оформление программы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_52

##### 3.1.9. Проверка правильности расчетов.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_53

##### Отчет о работе. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_54

#####  Модернизация. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_54

#### 3.2. Необходимость отладки разработанного программного продукта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_55

#### Методы и средства отладки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_58

##### Контроль программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_58

##### 3.3.2. Контроль результатов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_60

##### 3.3.3. Классификация методов контроля\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_63

##### 3.3.4. Локализация ошибок.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_64

##### 3.3.5. Технология отладки программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_67

# Введение.

 Постепенно с развитием программного обеспечения ЭВМ появились идеи создания управляющих систем, которые позволяли бы накапливать, хранить и обновлять взаимосвязанные данные по целому комплексу решаемых задач. Эти идеи нашли свое воплощение в системах управления базами данных (СУБД). СУБД взаимодействуют не с локальными, а взаимосвязанными по информации массивами, называемыми базами данных. С появлением персональных компьютеров СУБД становятся наиболее популярным средством обработки табличной информации. Они являются инструментальным средством проектирования банков данных при обработке больших объемов информации.

На сегодняшний день обработка заявок в ремонтно-строительной компании (РСК) производится «вручную», что приводит к большим затратам времени, что в большинстве случаев вызывает недовольство клиентов.

Данная система управления основана на взаимодействии нескольких баз данных, созданных с помощью среды ACCESS. Вышеуказанная система касается той части работы РСК, которая связана с поступлением и обработкой заявки на ремонт. Описываемый программный комплекс способствует сокращению временного интервала от обращения заказчика в РСК до заключения договора о проведении работ; снижает затраты на обработку первичной информации об объекте; содержит нормативно – справочную информацию; ведет учет клиентов.

Основное преимущество системы заключается в обработке заявок по структуре: First Input First Output (F.I.F.O.), т.к. при поступлении заявки, происходит ее обработка, и клиент сразу же получает ответ о стоимости проведения ремонта и список материалов, необходимых для выполнения ремонтных работ. Таким образом создается благоприятный психологический климат при работе менеджера РСК с клиентами.

Так же к достоинствам программы стоит отнести то, что она разработана в рамках дипломного проекта, что значительно снижает ее себестоимость.

**Глава 1.**

# Аналитическая часть.

Разработал Солнцев М. А.

Руководитель Гагарина Л. Г.

#### 1.1. Структура ремонтно-строительной компании

В данном дипломном проекте затрагивается деятельность фирмы, которая оказывает услуги в области ремонта и строительства жилых зданий и общественных сооружений.

На сегодняшний день, для нормального функционирования, любая ремонтно-строительная компания (РСК) обязательно должна иметь в своем составе хотя бы следующие подразделения:

1. Административный отдел;
2. Расчетно-экономический отдел;
3. Отдел маркетинга;
4. Отдел по работе с клиентами;
5. Отдел снабжения;
6. Производственный отдел.

Административный отдел, в лице директора, обеспечивает общее руководство коллективом и решение принципиальных технических задач, связанных с выбором портфеля заказов, обеспечением конструкторской и технологической документации, закупкой оборудования и стандартной оснастки и т.п.

Он же ведет оперативное управление делами предприятия. При решении этих задач руководитель при необходимости привлекает других специалистов, поручая им выполнение конкретных заданий.

Расчетно-экономический отдел осуществляет бухгалтерскую отчетность и технико-экономическое планирование. Бухгалтер выполняет расчетные работы, оформляет необходимую документацию. Следит за точностью и своевременностью расчетов с потребителями, поставщиками и органами налогового контроля. Составляет итоговые бюджетные отчеты для предоставления в налоговые органы. Принимает активное участие в планировании в области налоговой и ценовой политики предприятия.

Основной задачей отдела маркетинга является эффективное управление рекламной компанией фирмы. Сюда можно отнести анализ рекламных изданий и размещение рекламы в наиболее популярных из них, распространение листовок, участие в специализированных выставках и т.п.

 Отдел по работе с клиентами занимается приемом и обработкой заявок граждан и организаций в РСК на проведение ремонтно-строительных работ. Затем первичная информация передается прорабу фирмы (лицу, занимающемуся проведением работ на объектах и составлением сметной документации), который встречается с потенциальным заказчиком, делает необходимые замеры объекта, после чего производит расчет стоимости услуг на проведение работ. На основании этого расчета заказчик принимает решение о заключении договора с данной компанией.

Отдел снабжения занимается закупкой и доставкой материалов, необходимого инвентаря и инструмента на объекты во время проведения работ. Здесь так же производят подбор материала по качественным и стоимостным характеристикам. Поставка комплектующих осуществляется согласно составленному прорабом графику проведения работ на объекте.

Производственный отдел занимается непосредственно проведением работ на объектах, с которыми заключены договора подряда. За каждым объектом закрепляется лицо ответственное за проведение работ на данном объекте (прораб). Прораб занимается расчетом и подбором необходимого количества специалистов для проведения ремонта, составляет подробный поэтапный график проведения работ. Во время ремонта строго контролируется соблюдение графика, технологии и качества работ.

#### 1.2. Постановка задачи.

Выше были рассмотрены функции различных подразделений РСК. Несомненно, работа каждого из них очень важна, но хотелось бы уделить особое внимание отделу по работе с клиентами, от эффективности работы которого во многом зависит финансовое состояние всего предприятия в целом.

В данной дипломной работе была поставлена задача по разработке программного продукта (ПП), который позволит ремонтно-строительной компании (РСК) значительно сократить промежуток времени от обращения потенциального клиента фирмы до заключения договора о проведении работ. В то же время разрабатываемый ПП позволит сократить работу персонала компании по обработке «пустых», т.е. не проходящих заказов. Это, в свою очередь, ведет к значительной экономии средств, так как обработка одного заказа занимает не менее восьми часов высококвалифицированного специалиста. Отмечу также, что программа позволяет специалисту Отдела по работе с клиентами проводить учет обращений и квалифицированно вести диалог с заказчиками, представляя достаточно сложную специальную информацию.

#### 1.3. Анализ необходимости внедрения автоматизированной системы.

В настоящее время в РСК обработка обращений граждан и организаций, по поводу проведения ремонтно-строительных работ, происходит «вручную», что отображено на рис.1.

Сведения о заказе

Предоставление сметной документации

Справочная информация

Менеджер по работе с клиентами

#### Директор

#### Прораб

# Заказчик

Заключение договора

Рис.1 Структурная схема неавтоматизированной системы обработки заявок

На вышеприведенной схеме прослеживается несколько этапов передачи информации:

1. Заказчик обращается на фирму к менеджеру по работе с клиентами, для подачи заявки на проведение ремонтно-строительных работ.
2. Менеджер по работе с клиентами отправляет сведения о заказе прорабу.
3. Прораб выезжает на объект, для уточнения сведений о ремонтных работах, производит необходимые замеры. После этого составляется сметная документация.
4. Составленные документы передаются менеджеру по работе с клиентами.
5. Менеджер представляет сметную документацию заказчику.
6. Заказчик, на основании предоставленного предложения, принимает решение о дальнейшем сотрудничестве с РСК.
7. Результаты принятого решения менеджер предоставляет директору фирмы.
8. При обоюдном согласии заключается договор между фирмой и заказчиком.

При таком положении дел на обработку одной заявки тратится несколько дней. Кроме того, менеджер вышеуказанного отдела должен владеть достаточно большим объемом специальной информации. Так же нельзя забывать, что в такой схеме обработки информации участвует еще одно лицо – прораб. От него зависит, насколько быстро он сможет связаться с заказчиком, произвести расчеты и составить необходимую документацию. При этом не следует забывать, что прораб так же бывает занят производством работ на других объектах. Предлагаемый ПП «Автоматизированная информационная система ремонтно-строительного предприятия (АИС РСК)» предполагает совсем иной подход к обработке заявок. Это можно увидеть на рис. 2:

Предоставление сметной документации

Справочная информация

Заключение договора

Менеджер по работе с клиентами

#### АИС

#### Заказчик

**Директор**

Рис.2. Структурная схема автоматизированной системы обработки заявок

На этой схеме прохождение информации осуществляется следующим образом:

1. Заказчик обращается на фирму к менеджеру по работе с клиентами для подачи заявки на проведение ремонтно-строительных работ.
2. Менеджер по работе с клиентами обращается к АИС РСК. Он заполняет карточку клиента, и система выдает все необходимые расчеты и информацию согласно запросам.
3. На основании полученной информации заказчик принимает решение о дальнейшем сотрудничестве с РСК
4. Результаты принятого решения менеджер предоставляет директору фирмы.
5. При обоюдном согласии заключается договор между фирмой и заказчиком.

На основании представленной схемы и описания можно увидеть, что процесс обработки заявок значительно упрощается. Благодаря АИС необходимую информацию о стоимости работ на объекте клиент получает в течение нескольких минут. Так же мы видим, что на этом этапе работы предприятия пропадает необходимость в каких-либо действиях со стороны прораба. Объем информации, которой должен обладать менеджер по работе с клиентами значительно уменьшается, так как всю необходимую специальную информацию содержит АИС.

Исходя из всего выше сказанного, польза от предлагаемой программы несомненна.

#### 1.4. Техническое задание на разработку.

**Автоматизированная информационная система учета и обработки заявок граждан в ремонтно-строительную компанию.**

#### *Введение.*

### Наименование

Автоматизированная информационная система ремонтно-строительной компании (АИС РСК) по учету и обработке заявок граждан.

* 1. Краткая характеристика и области применения.

АИС РСК представляет собой информационную систему для автоматизации ввода, регистрации и выдаче информации о стоимости проведения работ. АИС предназначена в основном, для использования в мелких и средних РСК.

1. ***Основание для разработки:***

Задание на дипломный проект.

1. ***Назначение и цели создания системы***
	1. . Назначение разработки:

Программный продукт (ПП) предназначен для учета и обработки заявок граждан и организаций в РСК, и представления отчетов в виде калькуляционных ведомостей.

3.2.Цели создания:

* Автоматизация учета и обработки заявок граждан в РСК;
* Сокращение временного интервала от обращения заказчика в РСК до заключения договора о проведении работ;
* Снижение затрат на обработку первичной информации об объекте;
* Создание благоприятного психологического климата при работе с клиентами.
1. ***Технические требования***
	1. Требования к функциональным характеристикам
		1. *Состав выполняемых функций:*
* предоставление готовых форм для ввода исходных данных;
* обработка исходных данных и составление смет;
* составление специальных форм отчетов.
	+ 1. *Организация входных и выходных данных:*

Входные данные - вносимые пользователем сведения о заказе на работы (реквизиты заказчика, виды работ, объем работ и т.п.). Выходные данные – отчеты о стоимости работ, а так же количество и стоимость материалов необходимых для проведения этих работ.

* + 1. *Временные характеристики:*

Машинная обработка данных составляет несколько секунд.

* 1. Требования к надежности

### В период опытной эксплуатации правильность работы системы проверяется тестовым вводом исходных данных.

* 1. Требования к условиям эксплуатации

Программа ориентирована на минимальные требования к компьютерной подготовке пользователей.

* 1. Требования к составу и параметрам технических средств

IBM PC 486/16 MB и выше.

* 1. Требования к информационной и программной совместимости
		1. *Информационные структуры на входе и выходе:*
* ОС Windows NT/9x/200x;
* Office 9x Pro.
	+ 1. *Методы решения:*

Построение СУБД на основе инфологической и датологической модели предметной области.

4*.5.3*. *Языки программирования и программные средства, используемые в программе*

Определяются на этапе эскизного проектирования.

4.6. Требования к упаковке, маркировке, транспортировке и хранению

 Отсутствуют.

1. ***Требования к программной документации***

Состав программной документации:

* руководство пользователя;
* листинг программы.
1. ***Технико-экономические показатели разрабатываются в*** экономической части проекта.
2. ***Стадии и этапы разработки***

определяются в соответствии с регламентом разработки дипломного проекта.

1. ***Порядок контроля и приемки***
	1. Общие требования к приемке работ:
* поэтапный контроль со стороны руководителя проекта;
* тестирование программных модулей;
* опытная эксплуатация.

Исполнитель \_\_\_\_\_\_\_\_\_\_\_/М.А. Солнцев/

2002 год

**Глава 2.**

# Информационно-программная часть.

Разработал Солнцев М. А.

Руководитель Гагарина Л. Г.

####  Функциональные возможности системы

Перед началом проектирования какой-либо системы необходимо в первую очередь определить состав тех операций, которые будут заложены в проектируемый комплекс программных средств, и проанализировать необходимость и возможность реализации функций средствами конкретной системы проектирования. АИС РСК по учету и обработке заявок граждан предназначена, как уже было сказано выше, в первую очередь для повышения эффективности работы менеджера по обработке заявок. Поэтому функциональные возможности программного комплекса должны быть направлены на решение конкретных задач возникающих в процессе работы.

В проектируемой системе необходимо заложить возможности, обеспечивающие ниже перечисленные сервисные и информационно-расчетные функции:

1. автоматический учет обращений физических и юридических лиц;
2. возможность предоставления готовых форм для ввода исходных данных;
3. автоматическая обработка исходных данных;
4. выдача специальных форм отчетов согласно запросам;
5. возможность дополнения и изменения информации в базах данных (БД);

#### 2.2. Построение модели БД.

Для проектирования структуры БД необходима исходная информация о предметной области желательно в формализованном виде. Предметной областью (ПО) называется часть реальной системы, представляющая интерес для данного исследования.

При проектировании автоматизированных информационных систем ПО представляется в виде моделей данных нескольких уровней. Число используемых уровней зависит от сложности системы, но в любом случае включает логический и физический уровни. ПО может относиться к любому типу организации (институт, завод, малое предприятие и т.п.).

Необходимо различать полную ПО предприятия в целом и организационную единицу этой ПО. Организационная единица, в свою очередь, может представлять свою ПО (отделы).

Информация, необходимая для описания ПО, зависит от реальной модели и может включать сведения о персонале, заработной плате, товарах, то есть сведения о людях, местах, предметах, событиях и понятиях.

Описание ПО, без ориентации на используемые программные и технические средства, называется *инфологической моделью* (ИЛМ).

##### 2.3. Инфологическая модель БД

##### Компоненты инфологической модели

 Компоненты инфологической модели включают ряд описаний объектов ПО и связи между ними. Описание ПО всегда представлено в какой-то знаковой системе. Поэтому кроме отношений, присущих ПО, возникают еще и отношения, обусловленные особенностями отображения ПО в языковой среде.

В настоящее время не существует единого стандарта или общепринятого способа построения ИЛМ. Для описания ИЛМ используются как языки аналитического (описательного) типа, так и графические средства.

В процессе анализа в ПО выделяют классы объектов. *Классом объектов* называют совокупность объектов, обладающих одинаковым набором свойств. Например, если в качестве ПО рассмотреть вуз, то в ней можно выделить следующие классы объектов: учащиеся, преподаватели, аудитории и т. д. Объекты могут быть реальными или абстрактными, как, например, предметы, которые изучают студенты.

При отражении в ИС каждый объект представляется своим идентификатором, который отличает один объект класса от другого, а каждый класс объектов представляется именем этого класса. Так, для объектов класса «Изучаемые предметы» идентификатором каждого объекта будет «Название предмета». Идентификатор должен быть уникальным. Каждый объект обладает определенным набором свойств. Для объектов одного класса набор этих свойств одинаков, а их значения могут различаться.

Каждому классу объектов в ИЛМ присваивается уникальное имя.

При описании ПО надо отразить связи между объектом и характеризующими его свойствами. Это изображается в виде линии, соединяющей обозначение объекта и его свойств.

Связь между объектом и его свойством может быть различной. Объект может обладать только одним значением какого-то свойства. Например, каждый человек может иметь только одну дату рождения. Назовем такие свойства *единичными.* Для других свойств возможно существование одновременно нескольких значений у одного объекта. Пусть, например, при описании «Сотрудника» фиксируется в качестве его свойства «Иностранный язык», которым он владеет. Так как сотрудник может знать несколько иностранных языков, то такое свойство будем называть *множественным.* При изображении связи между объектом и его свойствами для единичных свойств будем использовать одинарную стрелку, а для множественных свойств – двойную. Кроме того, некоторые свойства являются постоянными, их значение не может измениться с течением времени. Назовем такие свойства *статическими,* а те свойства, значение которых может изменяться со временем, будем называть *динамическими.*

Кроме связи между объектом и его свойствами, в ИЛМ фиксируются связи между объектами разных классов. Различают связи типа *«один к одному»* (1:1), *«один ко многим»* (1:М), *«многие к одному»* (М:1) и *«многие ко многим»* (М:М). Иногда эти типы связей называются степенью связи. Кроме степени связи в ИЛМ для характеристики связи между разными сущностями надо указывать так называемый *класс принадлежности,* который показывает, может ли отсутствовать связь объекта данного класса с каким-либо объектом другого класса. Класс принадлежности сущности должен быть либо обязательным, либо необязательным.

С учетом вышеперечисленных особенностей построения ИЛМ были разработаны восемь объектов данных:

* Объект Карточка клиента содержит: данные о заказчике, дату поступления заказа и исходные данные об объекте для расчетов: площадь объекта, высота стен, окончательный материал пола, стен и потолка.
* Объект Материалы включает: наименование строительного материала, единицы измерения (м.кв., м.п., шт. и т.п.) и стоимость.
* Объект Нормы расхода содержит в себе информацию том, сколько затрачивается того или иного материала на единицу каждого типа работ: наименование или код работ, наименование материала, единицы измерения и количество.
* Объект Единицы измерения содержит вспомогательные сведения о всевозможных единицах измерения различных материалов.
* Объект Работы включает следующие аргументы: наименование, цену на единицу измерения, а так же тип выполняемых работ (штукатурные, малярные и т.п.).
* Объект Типы работ содержит вспомогательные сведения подстановки о всевозможных типах выполняемых работ.
* Объект Список работ предназначен для выборки всех работ, которые необходимо выполнить перед выполнением окончательной работы.
* Объект ЗаказыРаботы является вспомогательным и предназначен для связи объектов Карточка клиента и Список работ, так как между ними должна быть связь «многие ко многим».

На основании описанных свойств объектов и их назначения построим ИЛМ нашей базы данных (см. рис. 3.1).

**Работы**

**Карточка клиента**

КодРабот

КодЗаказа

ФИО (Наименование)

КодТипа

Работа

Телефон

ЕдИзмерения

Адрес

ДатаОбращения

Цена

Площадь

ВысотаСтен

##### Заказы Работы

Полы

КодЗаказа

Стены

КодОкончРаботы

Потолок

##### Типы Работ

**Единицы измерения**

КодЕдИзмерения

КодТипа

ЕдИзмерения

Тип

Рис. 2.1 Инфологическая модель базы данных.

##### Материалы

##### Нормы расхода

###### *КодМатериала*

КодНормы

КодРабот

Материал

КодМатериала

КодЕдИзмерения

ЕдИзмерения

Цена

Количество

##### Список работ

*КодОкончРаботт*

ОкончРабота

КодРабот

Рис. 2.1 Инфологическая модель базы данных (продолжение).

ИЛМ предметной области строится первой еще на предпроектной стадии и затем уточняется на более поздних стадиях. Затем на ее основе строится ДЛМ. Физическая и внешняя модели после этого могут строиться в любой последовательности, в том числе и параллельно. При проектировании БД возможен возврат на предыдущие уровни. При этом возможны два вида возвратов: первый вид обусловлен необходимостью пересмотра результата проектирования (например, для улучшения полученных характеристик, «обхода» ограничений и т. п.), второй вид вызван необходимостью уточнения предыдущей модели (обычно инфологической) с целью получения дополнительной информации для проектирования или при выявлении противоречий в модели.

##### 2.4. Выбор СУБД.

После построения ИЛМ необходимо выбрать СУБД, с помощью которой мы будем управлять нашими БД.

На сегодняшний день существует много разнообразных систем управления базами данных. Это такие СУБД как Paradox, FoxPro, Clipper, Access и др. Для работы с большинством из них требуются достаточно глубокие знания данной СУБД и опыт программирования.

Успех Microsoft Access заключается в прекрасной реализации продукта, рассчитанного как на начинающего, так и квалифицированного пользователя. Microsoft Access – это самая популярная сегодня настольная система управления базами данных.

В Microsoft Access присутствует язык программирования Visual Basic, который позволяет создавать массивы, свои типы данных, контролировать работу приложений. MS Access имеет один из самых лучших наборов визуальных средств разработки и представления информации среди аналогичных программных продуктов.

Одно из основных преимуществ MS Access – интеграции с популярным офисным пакетом Microsoft Office.

Вся работа с базой данных осуществляется через окно контейнера базы данных. Отсюда осуществляется доступ ко всем объектам: таблицам, запросам, формам, отчетам, макросам, модулям.

Встроенный язык запросов SQL позволяет максимально гибко работать с данными и значительно ускоряет доступ к внешним данным.

Access воспринимает большое количество форматов данных, включая файловые структуры других СУБД. Поэтому приложение в Access может импортировать из текстовых файлов или электронных таблиц и экспорт в них: предоставлять прямой доступ и обновлять файлы Paradox, FoxPro и других БД. Можно также импортировать данные из этих файлов в таблицы Access.

Преимуществом Access является наличие средств проектирования приложения БД без знания языка программирования. Работа в Access начинается с определения реляционных таблиц и полей, предназначенных для хранения данных. Сразу после этого с помощью форм, отчетов, макросов и VBA можно определять действия над этими данными. Формы и отчеты используются для вывода на экран и дополнительных вычислений при работе с таблицами. В случае разработки более сложного приложения можно использовать язык Visual Basic.

Архитектура Access называет объектами все, что может иметь имя. В БД Access основными объектами являются таблицы, запросы, формы, отчеты, макросы и модули. Термин БД обычно относится только к файлам, в которых хранятся данные. В Access БД включает все объекты, связанные с хранимыми данными, в том числе и те, которые определяются для автоматизации работы (см. Табл. 2.1.).

Таблица 2.1.

Компоненты СУБД Access.

|  |  |
| --- | --- |
| Объект | Описание |
| Таблица | Содержит информацию об объектах. Поля (столбцы) хранят характеристики объектов, а каждая запись (строка) содержит сведения об объекте.  |
| Запрос | Фиксирует нужные данные из одной или нескольких таблиц. Для запроса можно использовать запрос по образцу или инструкцию SQL –запросы на выборку и обновление данных. |
| Форма  | Отражает требования к данным таблиц или запросов. Формы можно распечатать. С помощью формы можно запустить макрос или VBA. |
| Отчет | Объект форматирования, вычисления итогов и печати данных. |
| Макрос | Описание действий Access в ответ на событие. Макрос открывает другую форму, может проверять поля при изменении его содержимого, открывать таблицы, запросы, просмотр или печать, запустить другой макрос или процедуру VBA. |
| Модуль | Программа на языке Visual Basic для приложений, обнаружения ошибки, которые не обнаруживает макрос. Модули могут быть независимыми объектами, содержащими функции, вызываемые из любого приложения или отчета для реакции на события. |

В таблицах хранятся данные. Используя формы, можно выводить данные на экран или изменять их. Формы и отчеты получают данные как непосредственно из таблиц, так и через запросы. Для выполнения вычислений запросы могут использовать встроенные функции или функции, созданные с помощью Visual Basic для приложений.

События в формах или отчетах могут запускать макросы или процедуры VBA. Событие - любое изменение состояния объекта Access, например открытие формы, закрытие формы, ввод новой строки в форму, изменение содержимого текущей записи или элемента управления. Для обработки события можно создать макрос или процедуру VBA, с помощью которых можно предусмотреть реакцию на любое действие пользователя, вплоть до нажатия определенных клавиш во время ввода данных. С помощью макросов и модулей можно изменять ход выполнения приложения; открывать, фильтровать и изменять данные в формах и отчетах; выполнять запросы и создавать новые таблицы. Используя VBA, можно создавать, модифицировать и удалять любой объект Access, обрабатывать данные по строкам и по столбцам или каким-либо другим способом. Можно также вызывать процедуры из библиотек динамической компоновки Windows, чтобы использовать в приложении не только встроенные в Access функции, но и возможности Windows.

Учитывая все вышесказанное, мы остановимся на СУБД Access для разработки нашего программного продукта.

##### 2.5. Построение датологической модели.

С учетом построенной инфологической модели и зная ограничения, налагаемые на хранимые данные используемой системой управления базами данных, строится датологическая модель базы данных.

Датологическая модель строится в терминах базы данных. Так как в нашем случае используется СУБД ACCESS, то мы строим реляционную модель базы данных в реализации MS ACCESS.

Она позволяет организовывать описание объектов в виде таблиц. При этом можно задавать ограничения на типы хранимых данных в столбце, первичные ключи для задания связи нескольких таблиц. Наконец можно задавать ограничения целостности с помощью триггеров и процедур.

Кроме того таблицы поддерживают ограничения на непустое значение поля и уникальное поле. Возможно так же задание индексации полей для последующего ускорения поиска данных в таблицах.

Построенная датологическая модель БД, с учетом особенностей MS ACCESS, выглядит следующим образом:

Таблица 2.2.

# Таблица «Карточка клиента»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодЗаказа* | *Счетчик* | *Идентификатор* |
| ФИОНаименование | Текстовый | Имя заказчика |
| Телефон | Числовой  | Телефон заказчика |
| Адрес | Текстовый | Адрес заказчика |
| ДатаОбращения | Дата/время | Дата обращения |
| Площадь | Поле МЕМО | Площадь помещения |
| ВысотаСтен | Поле МЕМО | Высота стен |
| Полы | Текстовый | Окончательная отделка пола |
| Стены | Текстовый | Окончательная отделка стен |
| Потолок | Текстовый | Окончательная отделка потолка |
| Двери | Числовой | Количество дверей |
| Перегородки | Поле МЕМО | Периметр перегородок |

Таблица 2.3.

Таблица «Работы»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодРабот* | *Счетчик* | *Идентификатор* |
| КодТипа | Числовой | Тип работ |
| Работа | Текстовый | Наименование работы |
| ЕдИзм | Текстовый | Единицы измерения |
| Цена | Денежный | Цена единицы работы |

Таблица 2.4.

Таблица «Типы работ»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодТипа* | *Счетчик* | *Идентификатор* |
| Тип | Текстовый | Тип работ |

Таблица 2.5.

Таблица «Единицы измерения»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодЕдИзмерения* | *Счетчик* | *Идентификатор* |
| ЕдИзмерения | Текстовый | Единицы измерения |

Таблица 2.6.

Таблица «Материалы»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодМатериала* | *Счетчик* | *Идентификатор* |
| Материал | Текстовый | Наименование материала |
| КодЕдИзмерения | Числовой | Единицы измерения |
| Цена | Денежный | Цена материала |

Таблица 2.7.

Таблица «Нормы расхода»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодНормы* | *Счетчик* | *Идентификатор* |
| КодРабот | Числовой | Наименование работ |
| КодМатериала | Числовой | Наименование материала |
| Единицы | Числовой | Единицы измерения |
| Количество | Поле МЕМО | Количество |

Таблица 2.8.

#### Таблица «Список работ»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодОкончРаботы* | *Счетчик* | *Идентификатор* |
| ОкончатРабота | Текстовый | Окончательная работа |
| КодРабот | Числовой | Наименование работ |

Таблица 2.9.

Таблица «ЗакзыРаботы»

|  |  |  |
| --- | --- | --- |
| Имя поля | **Тип данных** | **Описание** |
| *КодЗаказа* | *Числовой* | *Код заказа* |
| *КодОкончРаботы* | *Числовой* | *Окончательная работа* |

Курсивом в таблицах выделен ключевой столбец.

Связи между таблицами выглядят следующим образом:

Рис. 2.2. Связывание таблиц

На рисунке показана организация связей между таблицами. Связи между таблицами объединены общей тематикой.

# Начало

Заполнить карточку клиента?

Подпрограмма заполнения карточки клиента

Нет

Запрос на смету

Подпрограмма запроса на смету

Да

Да

Нет

Да

Да

Нет

Запрос на список материалов

Подпрограмма запроса на список материалов

Редактирование таблиц

Подпрограмма редактирования таблиц

# Выход

Рис.2.3. Общий алгоритм работы программы.

При проектировании рабочей модели системы, с учетом информационных потребностей пользователя, был разработан общий алгоритм работы программы, который показан на рис.2.3. . Из этого рисунка хорошо просматриваются функциональные возможности системы. Эти возможности реализуются через отдельные блоки подпрограмм. При входе в главное меню системы, пользователь выбирает один из пунктов меню, что и является в конечном итоге выбором конкретной подпрограммы.

Функциональные особенности подпрограмм заключаются в следующем:

* Подпрограмма заполнения карточки клиента предоставляет пользователю готовые формы для ввода данных (реквизиты заказчика, виды работ, параметры объекта), которые служат базой для проведения расчетов. Так же здесь ведется учет обращений юридических и физических лиц в РСК.
* Подпрограмма запроса на смету предназначена для произведения расчетов и выдачи готовых результатов, в виде ремонтно-строительных смет, на основании данных содержащихся в карточке клиента.
* Подпрограмма запроса на список материалов предназначена для произведения расчетов и выдачи готовых результатов, в виде перечня ремонтно-строительных материалов и их стоимости на конкретный объект. Эти расчеты, так же, производятся на основании данных содержащихся в карточке клиента.
* Подпрограмма редактирования таблиц служит для изменения данных в таблицах о стоимости на производство работ и цен на материалы. С помощью этой подпрограммы можно вносить дополнения ко всем базам данных содержащимся в разработке. Алгоритм работы этой подпрограммы показан на рис. 2.4.

**Начало**

Редактировать таблицу «Работы»?

Даа

Даа

Изменение цен на работы

Корректировка расценок

Нет

Нет

Нет

Нет

Даа

Даа

Даа

Редактировать таблицу «Типы работ»?

Добавление новых видов работ

Добавление новых типов работ

Изменение цен на материалы

Редактировать таблицу «Материалы»?

Корректировка цен на материалы

Нет

Даа

Редактировать таблицу «Нормы расхода»?

Добавление новых норм расхода

**Выход**

Рис. 2.4. Алгоритм работы подпрограммы редактирования таблиц

**Глава 3.**

# Организационно - технологическая часть

Разработал Солнцев М. А.

Руководитель Гагарина Л. Г.

#### 3.1. Общая структура организации работ по проектированию ПП.

##### 3.1.1. Постановка задачи.

Задача, которую предстоит решить программисту на ЭВМ, формулируется им самим или выдается ему в виде специального задания на разработку программы. Задание содержит формулировку задачи, необходимые характеристики разрабатываемой программы, требования к взаимодействию с ней. Выдаче такого задания для крупных задач может предшествовать большая работа научно-исследовательского характера.

Задание на разработку программы по форме и характеру должно быть аналогично техническому заданию (ТЗ) на разработку какого-либо технического продукта (см., например, ГОСТ 19.201-78 Единой системы программной документации).

Техническое задание полезно и в том случае, когда заказчик и исполнитель работают в одной и той же комнате или даже являются одним и тем же лицом. Наличие четкой письменной формулировки будет препятствовать подмене или отходу в процессе разработки программы от сформулированных в ТЗ требований в угоду каким-то другим побочным целям. Кроме того, письменно сформулированное задание делает возможным обсуждение, оценку или согласованную с заказчиками (пользователями) корректировку отдельных требований ТЗ в ходе разработки программы. ТЗ препятствует проникновению в программу таких ошибок и противоречий, которые могут быть обнаружены только после разработки большей части программы или уже на стадии анализа полученных результатов счета. Чем более формализованным по характеру будет техническое задание, тем больше шансов, что разрабатываемая программа будет решать именно ту задачу, которую имел ввиду заказчик.

Техническое задание должно содержать также требования или указания, касающиеся принципов проверки и испытаний готовой программы.

##### 3.1.2. Составление проекта.

 На основании анализа технического задания программист выбирает основной метод решения задачи, составляет общий проект программы. Выбранный подход к решению задачи должен обеспечивать правильные результаты для тех условий функционирования программы, которые определены ТЗ, гарантировать требуемую скорость работы, предусматривать удобство использования программы и т. п.

В проекте, помимо формулировки выбранного общего метода решения задачи, характеризуются основные части проектируемой программы, их функции, взаимосвязь и последовательность выполнения, а также точно определяются входные данные и выдаваемые результаты, как всей программы, так и основных ее частей. Поскольку каждая разрабатываемая программа, как правило, используется в дальнейшем не только ее автором, но и другими программистами. Составляется и проект инструкции для пользователей, в которой фиксируется (и, таким образом, может быть заранее оценен и исправлен) предполагаемый режим общения пользователя (и оператора) с программой.

Для очень простых задач проектирование может производиться программистом мысленно, без фиксации на бумаге. Наоборот, очень сложные задачи могут потребовать нескольких этапов проектирования: пред эскизное, эскизное, техническое - по аналогии с инженерным проектированием.

##### 3.1.3. Алгоритмизация.

На этот этап иногда смотрят как на вспомогательный, подготовительный к выполнению следующего этапа считающегося основным (см. 3), на котором производится написание программы на выбранном языке программирования. Введение такого "промежуточного" этапа преследует цель облегчить выполнение этапа 3 и тем самым предотвратить возникновение многих ошибок при его осуществлении для сложных задач. Формулировка алгоритма закрепляет последовательность основных шагов выполнения программы, четко фиксирует функциональное содержание ее частей, позволяет уделить необходимое внимание простоте логической структуры разрабатываемой программы. Этап алгоритмизации является совершенно необходимым также, в случае, если язык, на котором предстоит программировать, не вполне освоен программистом-разработчиком и предвидятся трудности при его использовании на следующем этапе (3).

При разработке алгоритма необходимо учитывать ресурсы используемой ЭВМ (ее скорость, память) и возможности применяемой для решения задачи операционной системы. Алгоритмы для несложных задач, требования которых к ресурсам невелики, являются обычно машинно-независимыми.

В ходе разработки общего алгоритма используется некоторый специальный язык, который по своему характеру является промежуточным, переходным между неформальным, словесным способом изложения метода решения задачи на этапе 1 и формальным алгоритмическим языком для программирования на этапе 3. Промежуточный язык должен сочетать в себе, с одной стороны, наглядность для отображения содержания и смысла, выполняемых в алгоритме действий (что делается) и, с другой стороны, формализм для указания конкретных операций и последовательности их выполнения (как делается).

В качестве такого промежуточного языка обычно используют блок-схемы, которые позволяют наиболее наглядно представить логическую структуру разрабатываемой программы, взаимосвязь отдельных частей программы, условия или кратность выполнения таких частей. Для отображения вычислительной (арифметической) стороны программы используются обычные математические средства или элементы алгоритмических языков, а в самых общих блок-схемах - просто словесная формулировка; иногда используются и все эти способы вместе.

После последнего шага детализации алгоритма (а иногда и после отдельных крупных шагов) проводится проверка полученного алгоритма для выявления допущенных ошибок. Методы контроля алгоритма аналогичны некоторым методам контроля программы. В ходе разработки алгоритма, возможно, придется уточнять или изменять решения, принятые на этапе 2, и в этом случае такие изменения обязательно вносятся в проект, который всегда должен соответствовать разрабатываемому алгоритму.

##### 3.1.4. Программирование.

В случае, когда на предыдущем этапе был получен детально разработанный алгоритм, составление программы на выбранном для программирования языке сводится к переводу этого алгоритма на язык программирования.

 Основные трудности и, следовательно, причины ошибок на этом этапе заключаются, во-первых, в необходимости знания всех требований и ограничений выбранного языка программирования и, во-вторых, в необходимости постоянного внимания ко многим деталям языка, которые приходится учитывать в ходе написания программы. Если этап 3 был выполнен некачественно и алгоритм представлен недостаточно детально, то его доводку придется выполнять «на ходу», во время программирования. Это затруднит процесс программирования-перевода и поведет к возникновению дополнительных ошибок в программе. Чем более процесс программирования будет походить на перевод, чем более механическим будет такой перевод, тем более легким будет составление программы и тем меньше возникнет ошибок на этом этапе, самом щедром на ошибки.

После составления программы проводится ее проверка для обнаружения и исправления ошибок, внесенных на этом этапе. Если при проверке обнаруживаются ошибки, допущенные на предыдущем этапе (3), то соответствующие исправления вносятся и в алгоритм, поскольку к нему еще придется обращаться на следующих этапах, и тексты алгоритма и программы должны соответствовать друг другу.

##### 3.1.5. Препарация

После составления программы производится ее перенос на машинные носители, т. е. подготовка программы к выполнению ее на ЭВМ; будем называть этот этап препарацией. Для предупреждения и сокращения ошибок препарации текст программы должен быть написан ясно и четко: чем небрежнее будет написан текст программы, тем больше ошибок возникнет в препарированной программе. Проверка правильности препарации осуществляется распечаткой программы, введенной в ЭВМ с использованных носителей, и последующей сверкой с исходным текстом.

Как видим, рассматриваемый этап, в отличие от предыдущих, уже требует для своего осуществления непосредственного использования ЭВМ или, по крайней мере, ее внешних устройств.

##### 3.1.6. Трансляция.

Транслятор в ходе осуществления трансляции, наряду с печатью транслируемой программы, производит поиск синтаксических ошибок в программе и, в случае их обнаружения, печатает диагностику, помогающую последующей локализации ошибок. Трансляция, а вместе с ней и поиск синтаксических ошибок, могут быть прекращены, если найдена очень грубая (с точки зрения транслятора) ошибка. Отсутствие синтаксических ошибок не говорит о том, что в программе нет ошибок препарации (например, вместо знака \* был отперфорирован + или записана не та буква в т. п.). Поэтому тщательная сверка напечатанной программы с исходным текстом всегда необходима на данном или предыдущем этапе. Первые трансляции вновь составленной программы производятся обычно с включением таких режимов транслятора, которые позволяют получить текст программы вместе с дополнительной информацией о программе (например, с таблицей используемых в ней переменных) для наиболее полной ее проверки.

##### 3.1.7. Отладка.

На этапе отладки производится обнаружение с помощью ЭВМ ошибок в программе и их исправление. Этап отладки можно разделить на три подэтапа:

* *Контроль правильности программы (КПП).*
* *Локализация ошибок (ЛО).*
* *Исправление ошибок (ИО).*

На подэтапе (КПП) - *контроль программы* - путем пропуска на машине специальных контрольных примеров устанавливается факт отсутствия или, в противном случае, наличия ошибок в программе. Здесь речь идет о содержательных (семантических) ошибках, которые не проявляются при трансляции программы.

* На этапе (ЛО) - *локализация ошибок* - точно устанавливается место, где в программе допущена ошибка (ошибки), последствия которой проявились при выполнении этапа (КПП) .
* На этапе (ИО) производится исправление ошибок, выявленных на этапе (ЛО). Исправления вносятся как в программу, так и в алгоритм, если он затрагивается этими исправлениями.

Перечисленные подэтапы могут повторяться многократно (включая и этап трансляции, точнее перетрансляции), до тех пор, пока контроль покажет, что ошибок в программе, по-видимому, нет.

##### 3.1.8. Оформление программы

Для возможности эксплуатации программы кем-либо кроме автора она должна быть оформлена: составлено ее описание, изготовлены машинные носители для передачи программы пользователям. В описание включается инструкция по использованию программы, излагается примененный метод решения, приводятся алгоритмы (иногда и текст программы), а также контрольные примеры с эталонными результатами. Наличие описания программы позволяет не только успешно эксплуатировать ее длительное время, но и проводить ее модернизацию и использовать в дальнейших разработках. Основную часть описания составляют материалы, с которыми шла работа на предыдущих этапах разработки (проект разработки и описание метода решения, общая блок-схема, алгоритмы, проект инструкции для пользователя и т. п.). Поэтому для ускорения этапа оформления все перечисленные материалы всегда должны быть в рабочем состоянии и по содержанию вполне соответствовать друг другу и отлаживаемой программе, кроме того, уже на этапах разработки их нужно представлять в таком виде, чтобы они могли быть использованы для описания программы без дополнительных переделок.

В случае, когда программа проста и предназначена для эксплуатации только ее автором, оформление программы может производиться уже после проведения счета по ней, одновременно с изготовлением отчета.

##### 3.1.9. Проверка правильности расчетов.

По окончании отладки и оформления программы начинается ее эксплуатация: производится проверка правильности расчетов по ней, обычно многократная. Первые полученные результаты реальных расчетов подвергаются тщательному анализу, чтобы убедиться в пригодности использованного метода и установить согласованность полученных результатов с имеющимися данными и теорией. Если правильность получаемых результатов не вызывает сомнений и эффективность программы удовлетворительна, то ее эксплуатация продолжается по мере необходимости. Но случается и так, что приходится снова рассматривать вопросы правильности разработанного алгоритма или пригодности реализованного метода, и тогда вся работа может вернуться к началу.

##### 3.1.10. Отчет о работе.

На основании результатов, полученных в ходе эксплуатации программы, составляется отчет о проделанной работе, оценивается выбранный метод решения задачи и эффективность программы; публикуются научные выводы.

##### 3.1.11. Модернизация.

Если разработчик программы постоянно работает в некоторой области науки или техники, то обычно рано или поздно наступает такой момент, когда перед ним возникает вопрос о модернизации старой программы или о составлении новой, развивающей идеи, реализованной в прежней программе. Модернизация программы проходит те же этапы, что и разработка, и начинается с составления технического задания на модернизацию. Успешное осуществление модернизации зависит от того, насколько легко можно будет при разработке новой программы использовать блоки старой программы и вносить в них изменения. Быстрое выполнение такого рода работ зависит, в свою очередь, как от структуры модернизируемой программы, так и от качества ее оформления (наличие описания программы, подробных алгоритмов, пояснений к программе и т. п.).

#### 3.2. Необходимость отладки разработанного программного продукта

Оказывается, практически невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки.

При разработке алгоритма программы решаются тактические вопросы проведения отладки, намечаются способы контроля отдельных блоков и приемы предстоящей локализации ошибок в них. Для этого проектируются контрольные примеры, по алгоритмам (блок-схемам) намечаются места и моменты необходимой отладочной печати и выбираются выводимые на печать данные, которые должны обеспечить возможность быстрой локализации ошибок при отладке (на этапе (ЛО)). Разрабатывая алгоритм, следует, таким образом, учитывать, можно ли будет достаточно просто проконтролировать программу, составленную по выбранному алгоритму, и в случае, когда предвидятся большие затруднения, нужно отдать предпочтение другому, более выгодному для этапа отладки, алгоритму. Нужно всегда помнить, что главным критерием ценности программы является ее правильность, и для гарантирования такого свойства про граммы следует жертвовать другими показателями, такими, например, как скорость работы или требуемый объем памяти. Давно ушли в прошлое те времена, когда программу оценивали только по количеству команд в ней.

 У начинающих программистов изложенный плановый подход к проведению отладки вызывает вначале трудности, поскольку им приходится разрабатывать план отладки для несуществующей пока программы. Но нет другого пути освоения этого эффективного способа, кроме как развивать в себе навыки планирования своей работы и предвидения особенностей предстоящей отладки программы по ее проекту и общим алгоритмам. Чем на более ранней стадии разработки программист начинает заниматься вопросами отладки программы, тем меньше неприятных неожиданностей ожидает его в будущем. Надежды на то, что устранение ошибок из программы и получение правильных результатов произойдет как-то само собой, без затраты особых усилий, никогда не оправдываются. Вообще, оптимизм и самоуверенность для программиста на стадии разработки противопоказаны; они могут являться полезными только на стадии отладки при затяжной борьбе с очень глубоко скрытыми ошибками.

Примерное распределение между этапами общего времени, необходимого для разработки достаточно сложных программ, выглядит следующим образом :

1. Получение задания, составление проекта программы и общего плана отладки 10%

1. Разработка алгоритма (1.5%) и детального плана отладки 20%
2. Программирование (5%) и изготовление тестов 5%

4- 5. Препарация и первая трансляция 5%

6. Отладка 40%

7. Оформление программы 10%

Приведенные цифры отражают тот факт, что в процессе разработки программы работы по доказательству (демонстрации) правильности разрабатываемой программы равнозначны работам по ее изготовлению (проектированию, алгоритмизации и написанию), что можно выразить следующей формулой:

 *разработка программы = изготовление + доказательство.*

Конечно, для простых задач распределение времени между этапами будет несколько другим, за счет увеличения доли программирования по отношению к алгоритмизации и отладке. Но, как правило, время, затрачиваемое на работы, связанные с отладкой, составляет около половины всего времени, необходимого на разработку программы. Поэтому вопрос минимизации времени, необходимого на отладку, имеет особое значение. К его решению можно подойти с двух сторон:

 а) путем ускорения поиска и исправления ошибок, имеющихся в программе;

б) путем уменьшения количества ошибок, допускаемых при разработке алгоритма и составлении программы.

#### 3.3. Методы и средства отладки

##### 3.3.1. Контроль программы

 Под этап контроля программы характеризуется как этап решения задачи, целью которого является установление наличия ошибок в составленной программе или убедительная демонстрация их отсутствия. Если будет установлено, что ошибки в программе имеются, то на следующем этапе (этап локализации) будет производиться их поиск. Поэтому в задачу контроля входит также получение еще и такой информации о характере работы программы, которая могла бы помочь в дальнейшем при поиске ошибок.

 *Контроль текста*

Контроль текста программы можно производить как "вручную", так и с применением ЭВМ. Сначала рассмотрим "ручные" методы контроля текста программ (алгоритмов). Можно различать три способов контроля текста без применения ЭВМ: просмотр, проверка и прокрутка.

*Просмотр*

Текст составленной программы внимательно просматривается на предмет обнаружения ошибок, описок и смысловых расхождений с текстом алгоритма, по которому производилось программирование. Помимо сплошного просмотра применяется еще и выборочный просмотр некоторых фрагментов программы.

*Проверка*

При проверке программы программист по тексту программы мысленно старается восстановить тот вычислительный процесс, который определяет программа, после чего сверяет его с требуемым процессом, т. е. ТЗ, определенном в проекте.

*Прокрутка*

Другим способом контроля программ и алгоритмов за столом является прокрутка (иногда ее называют "сухой" прокруткой – ***dry running*** - для отличия от метода прокрутки, применяемого на этапе локализации и использующего ЭВМ). Основой прокрутки является имитация программистом выполнения программы на машине, с целью более конкретного и наглядного представления о процессе, определяемом текстом проверяемой программы. Прокрутка дает возможность приблизить последовательность проверки программы к последовательности ее выполнения, что позволяет проверять программу как бы в динамике ее работы, проверять элементы вычислительного процесса, задаваемого проверяемой программой а не только статичный текст программы. Для выполнения прокрутки обычно приходится задаваться какими-то исходными данными и производить над ними необходимые вычисления.

*Печать текста*

Для проверки правильности препарации программы (переноса текста на какие-либо машинные носители) после ввода в машину производится печать текста программы для последующей его сверки с исходным текстом.

##### 3.3.2. Контроль результатов

Как бы ни была тщательно проверена и прокручена программа за столом, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на ЭВМ. Наиболее универсальным методом проверки для всех классов задач является метод контрольных тестов или тестирование. Тестом будем называть информацию, состоящую из исходных данных, специально подобранных для отлаженной программы, и из соответствующих им эталонных результатов (не только окончательных, но и промежуточных), используемых в дальнейшем для контроля правильности работы программы.

*Тестирование*

Под тестированием следует понимать процесс исполнения программы с целью обнаружения ошибок, в качестве которых принимается любое отклонение от эталонов. Хорошим считается тест, который имеет высокую вероятность обнаружения еще не выявленных ошибок.

Под отладкой понимается процесс, позволяющий получить программу, функционирующую с требуемыми характеристиками в заданной области входных данных. Таким образом, в результате отладки программа должна соответствовать некоторой фиксированной совокупности правил и показателей качества, принимаемой за эталонную для данной программы.

 Существует 3 основных способа тестирования: алгоритмическое, аналитическое, содержательное.

*Алгоритмическое тестирование*

 Алгоритмическое тестирование применяется программистом для контроля этапов алгоритмизации и программирования. Программисты проектируют тесты и начинают готовить эталонные результаты на этапе алгоритмизации, а используют их на этапе отладки.

*Функциональное или аналитическое тестирование*

Аналитическое тестирование служит для контроля выбранного метода решения задачи, правильности его работы в выбранных режимах и с установленными диапазонами данных. Тесты проектируют и начинают готовить сразу после выбора метода, а используют их на последнем этапе отладки или для анализа результатов пробного счета; в ходе тестирования, наряду со сверкой на совпадение, применяются и качественные оценки результатов.

*Содержательное тестирование*

Содержательное тестирование служит для проверки правильности постановки задачи. Для контроля при этом используются, как правило, качественные оценки и статистические характеристики программы, физический смысл полученных результатов и т.п. в проведении содержательного тестирования, принципы которого формулируются в техническом задании, самое активное участие должны принимать заказчики или идущие пользователи программы.

Содержательные и аналитические тесты проверяют правильность работы программы в целом или крупных ее частей, в то время как алгоритмические тесты в первую очередь должны проверять работу отдельных блоков или операторов программы.

##### 3.3.3. Классификация методов контроля

КОНТРОЛЬ

1. *По тексту.*
	1. Без ЭВМ.
		1. Просмотр.
		2. Проверка.
		3. Прокрутка.
	2. С ЭВМ.
		1. Печать.
		2. Трансляция (синтаксический контроль).
		3. Статический анализ.

*2. По результатам.*

* 1. Тестирование.
		1. Алгоритмическое.
		2. Функциональное.
		3. Содержательное.
	2. Специальные методы.

##### 3.3.4. Локализация ошибок.

***Способы локализации***

После того, как с помощью контрольных тестов (или каким либо другим путем) установлено, что в программе или в конкретном ее блоке имеется ошибка, возникает задача ее локализации, то есть установления точного места в программе, где находится ошибка.

 Процесс локализации ошибок состоит из следующих трех компонент:

1. *Получение на машине тестовых результатов.*
2. *. Анализ тестовых результатов и сверка их с эталонными.*
3. *. Выявление ошибки или формулировка предположения о характере и месте ошибки в программе.*

 По принципам работы средства локализации разделяются на 4типа:

1. *Аварийная печать.*
2. *Печать в узлах.*
3. *Слежение.*
4. *Прокрутка.*

*АВАРИЙНАЯ ПЕЧАТЬ* осуществляется один раз при работе отлаживаемой программы, в момент возникновения аварийной ситуации в программе, препятствующей ее нормальному выполнению. Тем самым, конкретное место включения в работу аварийной печати определяется автоматически без использования информации от программиста, который должен только определить список выдаваемых на печать переменных.

*ПЕЧАТЬ В УЗЛАХ* включается в работу в выбранных программистом местах программы; после осуществления печати значений данных переменных продолжается выполнение отлаживаемой программы.

*СЛЕЖЕНИЕ* производится или по всей программе, или на заданном про грамм истом участке. Причем слежение может осуществляться как за переменными (арифметическое слежение), так и за операторами (логическое слежение). Если обнаруживается, что происходит присваивание заданной переменной или выполнение оператора с заданной меткой, то производится печать имени переменной или метки и выполнение программы продолжается. Отличием от печати в узлах является то, что место печати может точно и не определяться программистом (для арифметического слежения); отличается также и содержание печати.

*ПРОКРУТКА* производится на заданных участках программы, и после выполнения каждого оператора заданного типа (например, присваивания или помеченного) происходит отладочная печать.

***Классификация средств локализации ошибок***

Ниже дана классификация средств локализации.

Средства локализации:

1. *Аварийная печать (арифметическая).*
	1. Специальные средства языка.
	2. Системные средства.
2. *Печать в узлах (арифметическая).*
	1. Обычные средства языка.
	2. Специальные средства языка.
3. *Слежение (специальные средства).*
	1. Арифметическое.
	2. Логическое.
4. Прокрутка (специальные средства).
	1. Арифметическая.
	2. Логическая.

##### 3.3.5. Технология отладки программы

Рассмотрим этапы создания рассматриваемой программы, основываясь на приведенных выше методах и приемах.

 Обычно рано или поздно наступает такой момент, когда возникает вопрос о необходимости создания программы. Перед нами стояла именно такая задача. То есть нам необходимо было, создать программу, рассчитанную на не специалистов в области компьютерной техники, т. е. программу с отлично проработанным интерфейсом для удобства и доступности пользователем. ТЗ выдавалось постепенно, уточнялось и изменялось. В зависимости от этого программа разрабатывалась поэтапно. Алгоритмизация в нашем случае охватывает, как программу целиком, так и поблочно, для процедур и функций.

При отладке программы использовались следующие методы контроля и локализации ошибок (обзор методов см. в теоретической части раздела):

1. Просмотр текста программы и прокрутка с целью обнаружения явных синтаксических и логических ошибок.
2. Трансляция программы (транслятор выдает сообщения об обнаруженных им ошибках в тексте программы).
3. При локализации ошибок преимущественно использовалась печать в узлах, которыми являлись в основном глобальные переменные, переменные, используемые при обмене данными основной про граммы с подпрограммами слежения. Также сверялись типы получаемых данных, что позволяет выявить ошибки несовпадения типов, которые не видны при компиляции.

 Известно, что существует проблема восприятия информации с экрана монитора. Чтобы избежать этого мы использовали распечатки программы на различных этапах разработки и отладки.

 В общем случае отладка программы производилась по следующему алгоритму:

1. Прогонка программы с набором тестовых входных данных и наличия ошибок.
2. Выделение области программы, в которой может находиться ошибка. Просмотр листинга программы с целью возможного обнаружения ошибок. В противном случае - установка контрольной точки примерно в середине выделенной области.
3. Новая прогонка программы. Если работа программы прервалась до обработки контрольной точки, значит, ошибка произошла раньше. Контрольная точка переносится, и процесс отладки возвращается к шагу 2.
4. Если контрольная точка программы была обработана, то далее следует изучение значений регистров, переменных и параметров программы с тем, чтобы убедиться в их правильности. При появлении ошибки - новый перенос контрольной точки и возврат к шагу 2
5. В случае не обнаружения ошибки продолжение выполнения программы покомандно, с контролем правильности выполнения переходов и содержимого регистров и памяти в контрольных точках. При локализации ошибки она исправляется, и процесс возвращается к шагу 1.

 Решающим этапом, устанавливающим пригодность программы для работы, является её контроль по результатам ее выполнения на ЭВМ. Наиболее универсальным методом проверки для всех классов задач является метод контрольных тестов или тестирование.

В процессе отладки созданной программы, был выявлен ряд ошибок. На основании результатов анализа проделанной работы по их устранению была разработана инструкция по пользованию созданной Автоматизированной системой и предложены методы устранения возможных ошибок.