Безруков А.И.

Основы метрологии, стандартизации и сертификации

Курс лекций

Аннотация

Курс лекций «Основы метрологии, стандартизации и сертификации» предназначен для студентов второго курса специальности «Программирование вычислительных систем». Лекции знакомят будущих программистов с основными понятиями:

метрологии и обеспечения единства измерений,
классификации и обеспечения единства терминологии,
стандартизации и нормативными методами управления производством,
сертификации.

Особое внимание уделено вопросам применения методов стандартизации в программировании. Отдельные главы посвящены постановке задачи на программирование и разработке комплекта программной документации. Приведены примеры оформления программной документации в соответствии с требованиями Единой системы программной документации (ЕСПД).

Благодарности

Автор признателен своим первым студентам, освоившим этот курс в 2003 году. Ваши вопросы, ваше желание понять и самим во всем разобраться помогли мне построить этот курс таким, каким он получился. Особую благодарность хочется выразить студентам: Пирожкову Артему, Барковой Натальи, Пугачевой Светлане, Улдаровой Кадрии, Черпак Ольге.

Оглавление

Введение

Предмет, который мы будем изучать, называется «Метрология, стандартизация и сертификация». Это очень короткий, ознакомительный курс. Его цель - дать Вам основные представления о проблемах и методах стандартизации.

- Мы познакомимся с понятием «Единство измерений» и методами его обеспечения.
- Рассмотрим проблемы обеспечения единства терминологии, стандартные приемы организации данных и программирования, обеспечивающие единство терминологии в информационных системах.
- Обсудим проблемы обеспечения совместимости и на примере модульного принципа разработки программ рассмотрим основные проблемы и приемы обеспечения совместимости изделий.
- Рассмотрим проблему подтверждения безопасности и качества продукции и изучим основные виды сертификации.
- Познакомимся с нормативными методами управления, назначением и способами применения стандартов и других нормативных документов. Научимся читать стандарты, познакомимся с Государственной системой стандартизации.

У вас сразу возникает вопрос: «Зачем программисту нужно разбираться во всех этих премудростях?» Программирование во всем мире относится к так называемым «высоким технологиям». Программный продукт является, пожалуй, самым современным видом продукции, вобравшим в себя труд высококвалифицированных разработчиков вычислительной техники, операционных систем, базового программного обеспечения и прикладных программистов. Для организации производства и использования программ, координации деятельности всех участников данного процесса широко применяются изучаемые нами методы. Таким образом, программист не знакомый хотя бы с основными понятиями метрологии, стандартизации и сертификации не может считаться квалифицированным специалистом.

Основы метрологии

Метрология – наука об измерениях. В ее задачи входят: поиск методов измерения интересующих нас характеристик, обеспечение сопоставимости и достоверности результатов измерений. В виду огромной практической значимости решаемой метрологией проблем, метрологические службы имеются практически на каждом предприятии. Для организации их взаимодействия в России действует ***Государственная система обеспечения единства измерений*** (ГСИ)

Зачем нужно единство измерений?

Рассмотрим конкретную задачу. Мы проектируем автоматизированную систему управления предприятием (АСУ). Первый этап – автоматизация склада готовой продукции. Программа должна позволять учитывать всю произведенную продукции, прослеживать пути ее распределения и использования (продажи, внутреннее использование, хранение и т.д.).

Одна из главных задач АСУ – баланс изготовления и использования (вся изготовленная продукция должна быть как-то использована). Однако у нас постоянно получается несоответствие объемов изготовленной и использованной продукции (то одной больше, то другой).

В чем дело? При поступлении на склад мы фиксируем объем продукции (например, ее массу), по измерениям, сделанным в цехах. А при отпуске пользуемся весами, находящимися на складе. Если не принять специальных мер, результаты этих измерений могут оказаться несопоставимыми!

Определение.
Сопоставимыми *называются результаты измерений, с которыми можно обращаться как с числами (складывать, вычитать). При этом сумма частей должна равняться целому.*

Как обеспечить единство измерений?

Как обеспечить сопоставимость наших измерений?

Можно отказаться от применения разных весов и взвешивать все на одних. Но это ужасно не удобно и очень дорого!

Можно поставить весы рядышком и взвешивать одни и те же предметы на обоих весах. Потом составить таблицу соответствия и пользоваться ей для ввода поправок результатов измерений. Тоже дорого и есть опасность запутать все дело.

Можно купить еще одни, более точные весы и проверять с их помощью все весы предприятия. Это уже гораздо лучше.

Как видим, проверять настраивать весы, так чтобы они стали показывать верный вес надо производить периодически, чтобы учесть износ весов и возможность их сбоев.

Таким образом, мы обеспечим единство измерений всех весов предприятия. Способы такой проверки могут быть разные.

Определение.
Единство измерений *- состояние измерений, при котором их результаты выражены в узаконенных единицах величин и погрешности измерений не выходят за установленные границы с заданной вероятностью* [1]. Данное состояние обеспечивает сопоставимость результатов измерений.

Поверка средств измерений

Назовем средство измерения, предназначенное для проверок других средств измерений, эталоном. Чтобы характеристики точности эталона сохранялись как можно дольше, будем использовать его только для проверок. Таким образом, все средства измерений разбиваются на два класса: рабочие и эталонные.

Определение.
Рабочее средство измерения *- прибор или оборудование используемый для измерений на производстве.*

Определение.
Эталонное средство измерения *- прибор или оборудование используемый только для поверки рабочих средств измерений и эталонов.*

Определение.
Поверка *- совокупность операций, выполняемых органами государственной метрологической службы (другими уполномоченными на то органами, организациями) с целью определения и подтверждения соответствия средства измерений установленным техническим требованиям* [1].

Эталоны так же нуждаются в периодической поверке. Для этого используются эталоны более высокого уровня. Для каждой основной величины существует государственный (национальный) эталон и специальная методика поверок, позволяющая «передать» характеристики измеряемой величины через цепочку промежуточных эталонов до рабочего средства измерения.

Чем занимается метрологическая служба предприятия

Основная задача ***метрологической службы предприятия*** – обеспечение единства измерения. В обязанности службы входят:

- Регистрация и отслеживание использования всего парка[1] технических средств предприятия. На крупных передовых предприятиях для этой цели организуются базы данных и информационные системы.
- Определение периодичности поверок и составление планов поверок технических средств.
- Проведение периодических плановых поверок, а также послеремонтных поверок. Результаты поверок протоколируются. Средства измерения признанные годными снабжаются соответствующим штампом (меткой). Указывается дата следующей поверки. Средствами измерения, не имеющими штампов о проведении поверки, или просроченной датой следующей поверки пользоваться нельзя.

[1] Парк технических средств – все приборы и механизмы, используемые на предприятии. Также, как и деревья парке они требуют присмотра и ухаживания.

- Методическая помощь в организации контроля и измерений на всех производственных участках предприятия.

Таким образом, метрологическая служба отвечает за все измерения, проводимые на предприятии.

Обеспечение единства измерения в стране

Для современного производства характерны широкая кооперация и интеграция. Например, предприятие, выпускающее автомобили сотрудничает с тысячами других предприятий. Продукция предприятий продается многим покупателям как в нашей стане, так и за рубеж. Очевидно, что такое взаимодействие не возможно без обеспечения единства измерений.

Чтобы обеспечить единство измерений всего народного хозяйства страны необходимо скоординировать работу всех метрологических служб, обеспечить передачу величин от государственных эталонов, осуществлять методическую и научную поддержку предприятий в области измерений.

Правовой основой обеспечения единства измерений в России является закон РФ "Об обеспечении единства измерений" [1] . Согласно этому закону (статья 4) Государственное управление деятельностью по обеспечению единства измерений в Российской Федерации осуществляет Комитет Российской Федерации по стандартизации, метрологии и сертификации (Госстандарт России). В его компетенцию входят следующие вопросы:

- межрегиональная и межотраслевая координация деятельности по обеспечению единства измерений в Российской Федерации; организация взаимодействия региональных и отраслевых служб с целью обеспечения единства измерений в стране в целом;
- научные исследования и обобщение практического опыта организации измерений, представление Правительству РФ предложений по единицам величин, допускаемым к применению;
- установление правил создания, утверждения, хранения и применения эталонов единиц величин;
- определение общих метрологических требований к средствам, методам и результатам измерений. Разработка стандартов на методики проведения измерений (МВИ);
- осуществление государственного метрологического контроля и надзора. Надзор осуществляется путем:
    - утверждение типа средств измерений. Вновь разработанный прибор должен пройти метрологическую экспертизу и быть зарегистрирован.
    - поверки и калибровки средств измерения, в том числе эталонов;
    - надзора за состоянием и применением средств измерений,
    - надзора за соблюдением метрологических правил и норм установленных в нормативных документах по обеспечению единства измерений;
- осуществление государственного контроля за соблюдением условий международных договоров Российской Федерации о признании результатов испытаний и поверки средств измерений;
- руководство деятельностью ***Государственной метрологической службы*** и иных государственных служб обеспечения единства измерений;
- участие в деятельности международных организаций по вопросам обеспечения единства измерений.

Непосредственное обеспечивает единство измерений Государственная метрологическая служба, находящаяся в ведении ***Госстандарта России***.

Она и включает:

- государственные научные метрологические центры;

- органы Государственной метрологической службы на территориях республик в составе Российской Федерации, автономной области, автономных округов, краев, областей, городов Москвы и Санкт-Петербурга.

В Саратовской области это Центр метрологии и стандартизации (ЦМС).

Для обеспечения единства отдельных видов измерений при Госстандарте РФ созданы:

- Государственная служба времени и частоты и определения параметров вращения Земли (ГСВЧ);
- Государственная служба стандартных образцов состава и свойств веществ и материалов (ГССО);
- Государственная служба стандартных справочных данных о физических константах и свойствах веществ и материалов (ГСССД).

Полный текст закона "Об обеспечении единства измерений" приведен в Приложении.

Виды и характеристики измерений

Более подробное рассмотрение измерений начнем с классификации. В таблице 1 приведена многоаспектная классификация видов измерений. Каждый столбец таблицы – способ классификации. В первой строке («шапке» таблицы) записан вопрос, отвечая на который мы будем классифицировать. Ниже – различные варианты ответов. Такая таблица разбивает все множество измерений на относительно однородные группы, соответствующие комбинациям ответов на все вопросы из «шапки».

Таблица 1

<table>
<tr><th rowspan="2">Что измеряем? (вид измерений)</th><th rowspan="2">Сколько величин измеряется?</th><th colspan="2">Как влияет измерение на:</th><th rowspan="2">Все ли объекты измеряются?</th></tr>
<tr><th>наблюдаемый процесс?</th><th>объект измерения?</th></tr>
<tr><td>Прямые. Свойство объекта измеряется непосредственно</td><td>Индивидуальные Измеряется одна величина</td><td>Пассивно</td><td>Неразрушающее</td><td>Сплошной контроль</td></tr>
<tr><td rowspan="2">Косвенные. Для определения значения измеряемого свойства измеряются свойства, связанные с ним. Значение свойства вычисляется.</td><td>Совокупные Измеряются несколько однородных величин.</td><td rowspan="2">Активно</td><td rowspan="2">Разрушающее. В процессе измерения объект разрушается.</td><td rowspan="2">Выборочный контроль</td></tr>
<tr><td>Совместные Измеряются несколько разнородных величин.</td></tr>
</table>

Некоторые комбинации ответов недопустимы. Например, в результате сплошного разрушающего контроля мы уничтожим всю продукцию.

Характеристики измерений.

Какие характеристики мы должны указать, чтобы однозначно описать измерение? В таблице 2 приведен минимальный набор характеристик.

Таблица 2

Характеристика	Комментарий

Измеряемая величина	
Единица измерения	Единица измерения должна соответствовать измеряемой величине
Диапазон измерений	Диапазон измерений должен быть задан в выбранных единицах измерения
Характеристики точности	Существует несколько способов задания точности (см. ниже)
Характеристики достоверности	

Отсутствие какой либо из перечисленных характеристик делает оставшуюся информацию бессмысленной, так как не дает представления о том, какое именно измерение имеется в виду.

Точность измерений

Важнейшей метрологической характеристикой является точность измерения. Поэтому рассмотрим ее суть и проблемы определения точности более подробно.

Откуда возникают погрешности измерений

Начнем с философской проблемы познания реального мира (гносеологии). Когда исследователь изучает реальный мир он строит его модель. (Рис. 1).

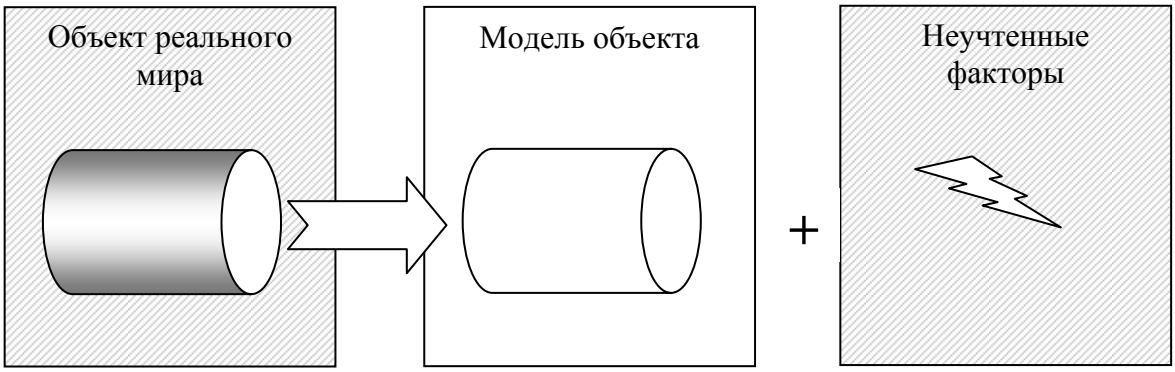


Рис. 1. Построение модели объекта реального мира

В зависимости от поставленной цели исследователь учитывает в модели одни особенности реального мира и отбрасывает как несущественные другие.

Например. Моделируя металлические валы, конструктор представляет их как правильные цилиндры. Отклонения вала от цилиндрической формы, шероховатость поверхности, цвет и т.д. в модели не учитываются. При прямом измерении характеристики вала (например, его длины) полученное значение будет зависеть от характеристики модели (точной длины) и неучтенных характеристик.

Обозначим результат измерения через X_e, а точное (модельное) значение той же характеристики через Xt . Тогда мы можем записать соотношение:

$$X_e = Xt + \delta \qquad (1)$$

Величина δ, получившая название ***погрешность измерения***, характеризует зависимость результата измерения от неучтенных факторов. Если модель построена удачно (в таких случаях принято говорить, что модель адекватна объекту реального мира), абсолютное значение величины δ существенно меньше модуля измеряемой величины:

$$|\delta| << |Xm| \sim |Xt| \qquad (2)$$

Систематическая и случайная погрешность

Давайте произведем несколько измерений одной и той же величины, теоретическое значение которой мы заранее знаем. Мы заметим, что значения δ концентрируются возле точки, в общем случае не равной нулю. Обозначив ее через δ_s, запишем:

$$\delta = \delta s + \delta r \qquad (3)$$

где δ_s - постоянная составляющая, называемая ***систематической погрешностью***

δ_r - переменная составляющая, среднее значение которой равно нулю. Эта величина называется ***случайной погрешностью***.

Проиллюстрируем различные виды погрешности на примере. Мои часы постоянно «убегают» на одну минуту в сутки. Но если учесть этот уход, по ним можно определить время с точностью до 0,5 сек.

Погрешности измерений и теория вероятности

Рассмотрим, как располагаются значения δ_r вокруг среднего значения. Для этого проведем серию замеров одной и той же величины и отобразим их результаты на таблице (Рис.2). Допустим, мы измеряем длину вала, номинальное (модельное) значение которой равно 100 мм. Возьмем листок бумаги «в клеточку» и в нижней его части проведем ось. Посередине поставим значение 100, а слева и справа последовательность значений с шагом 0,01 мм. Измерив значение величины, мы поставим крестик в клеточки над этим значением. Если результаты последующих измерений будут равняться тому же значению, крестик поставим клеточкой выше. Такой метод называется «Метод контрольных листков» [2] и широко применяется на практике при анализе причин появления некачественной продукции.

				X						
				X	X					
				X	X					
				X	X	X				
			X	X	X	X				
		X	X	X	X	X	X			
X	X	X	X	X	X	X	X	X	X	
99.95	99.96	99.97	99.98	99.99	100	100,01	100,02	100,03	100,04	100,05

Рис. 2. Контрольный листок

В нашем примере значение 99.99 встретилось 7 раз, 100,02 – 2 раза, а 100,05 ни разу.

Проделав достаточно большое число замеров, мы получим «горку». Изучив, таким образом, множество измерений совершенно различных величин, мы заметим, что форма «горки» (Рис. 3) оказывается схожей для всех них. Впервые дал этому математическое объяснение немецкий математик Гаусс. Но прежде чем рассказывать о его исследованиях нам придется познакомиться с основными понятиями теории вероятности. Вы будете подробно изучать этот курс. Поэтому сейчас мы остановимся только на некоторых, нужных для понимания метрологии, понятиях и фактах.

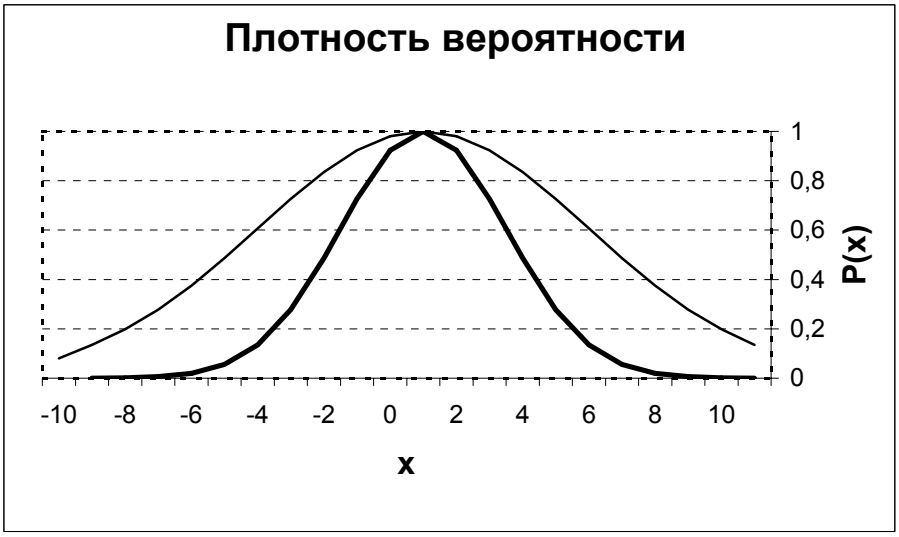


Рис. 3. Теоретические кривые Гаусса для различных распределений.

Назовем ***случайной величиной*** процесс, численную характеристику, которого нельзя предсказать заранее. Очевидно, многократное измерение одной величины является примером такого процесса. ***Частотой*** (**N**m) появления определенного значения (m) случайной величины назовем количество случаев, в которых случайная величина принимает данное значение. В нашем примере частота значения 99,99 равна 7. Если мы предполагаем, что частоты появления различных значений отражают свойства самой случайной величины, желательно выразить эти свойства вне зависимости от числа испытаний (в нашем примере, замеров). Назовем ***вероятностью*** того, что случайная величина примет данное значение, предел отношения:

$$\mathbf{P}m = \lim_{N \to \infty} (\mathbf{N}m/N) \qquad (4)$$

Где $\mathbf{N}_m$ – частота появления значения m;

N – общее число испытаний.

Равенство полученного значения случайной величины определенному значению это событие в нашем вероятностном мире. Если события не зависят друг от друга, то вероятность появления какого - либо из них равна сумме их вероятностей[2]. Вероятность появления хоть какого-то события равна единице.

События, изображенные на рис. 3 находятся так близко друг от друга, что их можно представить в виде непрерывной последовательности. Тогда кривую, похожую на шляпу гнома, можно интерпретировать как плотность вероятности, показывающую, насколько изменится вероятность события, если величина изменится на единицу.

Вероятность события $X < X_0$ будет равна площади под кривой, от линии $X = X_0$ до $-\infty$ (Рис. 4) .

[2]

Рис 4. Оценка вероятности событий по графику функции плотности вероятности

Вероятность события X< -5 равна заштрихованной площади в левой части рисунка, вероятность X > 5, 5 равна заштрихованной площади в правой части рисунка.

Вся площадь под кривой – вероятность того, что X примет хоть какое то значение равна единице. Тогда, вероятность того, что X находится в интервале: $X_1<X<X_2$ равна единице минус заштрихованная площадь слева и справа. Выбрав X_1 и X_2 такими, чтобы заштрихованная площадь была небольшой (например, 0,01), мы можем сказать: «С ***доверительной вероятностью*** 1- 0,01= 0,99 величина X находится в интервале (X_1; X_2).

Характеристики распределения случайной величины

Как видно из рис. 3, различные случайные величины имеют различный разброс своих значений. Наглядно это показывает различная ширина «шляп». В теории вероятности рассматриваются несколько характеристик случайных величин:

Среднее значение - предел отношения суммы всех значений к общему числу наблюдений:

$$\mathbf{X}_{cp} = \lim_{N\to\infty} (\sum \boldsymbol{X}_m)/N \qquad (5)$$

Для оценки среднего значения используется допредельное выражение при достаточно большом числе испытаний

$$\mathbf{X}_{cp} \sim 1/N*\sum \boldsymbol{X}_m \qquad (6)$$

Если случайная величина может принимать только определенные значения: $\boldsymbol{X}_1$ $\boldsymbol{X}_2$ $\boldsymbol{X}_m$ и известны вероятности появления этих значений P_1 P_2 P_m, среднее значение может быть вычислено по формуле:

$$\mathbf{X}_{cp} = \sum \boldsymbol{X}_i * \mathbf{P}_i \qquad (7)$$

Дисперсия – мера разброса значений случайной величины, определяется как среднее значение квадрата отклонения случайной величины от ее среднего значения:

$$\mathbf{D}= \lim_{N\to\infty} (\sum (\boldsymbol{X}_m - \mathbf{X}_{cp})^2/N) \qquad (8)$$

Для оценки дисперсии при достаточно большом числе испытаний используется формула:

$$\mathbf{D} \sim 1/(N*(N-1))*\sum (\boldsymbol{X}_m - \mathbf{X}_{cp})^2 \qquad (9)$$

Если случайная величина может принимать только определенные значения с известными вероятностями, дисперсия вычисляется по формуле:

$$\mathbf{D} = 1/(M*(M-1))*\sum (\boldsymbol{X}_i - \mathbf{X}_{cp})^2 * \mathbf{P}_i \qquad (10)$$

Где M – число различных значений ***X***.

Величина **D** характеризует разброс значений вокруг среднего. В случае измерений, **D** может использоваться в качестве меры случайной погрешности измерения. Однако, применение **D** не удобно. Дело в том, что размерность **D** равна квадрату размерности измеряемой величины. Например, мы измеряем расстояние в метрах. Подставим в (10) результаты наших замеров. Размерность **D** получилась равной $м^2$. Поэтому вместо **D**, в качестве меры случайной ошибки, применяют величину **σ** – называемую ***среднеквадратическим отклонением***.

$$\boldsymbol{\sigma} = \sqrt{\mathbf{D}} = \sqrt{(1/(M*(M-1))*\sum (\boldsymbol{X}_i - \mathbf{X}_{cp})^2 * \mathbf{P}_i)} \qquad (11)$$

Для распределения Гаусса справедливо соотношение: «Вероятность, того, что значение величины отличается от среднего значения более чем на 3**σ** меньше 0,01. Таким образом, мы можем использовать **σ** как меру погрешности нашего измерения.

Отступление для программистов. **Как бороться с грубыми ошибками?**

При организации ввода данных в нашу информационную систему мы можем использовать вероятностный метод выявления грубых ошибок в данных. Как мы видели, отклонения значения величины за 3**σ** маловероятны. Организуем ввод данных так, чтобы сразу считать $\mathbf{X}_{cp}$ и **σ** для уже введенных данных. Тогда, при появлении значения отличающегося от среднего больше чем на 3**σ**, программа выдает сообщение: **«Введенное значение маловероятно! Пожалуйста, проверьте правильность ввода».** Вовсе не обязательно, что мы ошиблись. Но проверить стоит. Таким образом, компьютер обращает внимание оператора на маловероятную информацию, что позволяет сократить число грубых ошибок.

Что мы понимаем под погрешностью измерений

В таблице 3 сведены различные ситуации применения термина «погрешность измерения».

Таблица 3

Ситуация	Смысл термина «погрешность измерения».
Мы проводим научный эксперимент.	С доверительной вероятностью 99% истинное значение измеряемой величины $\boldsymbol{X_t}$ лежит в интервале: $\boldsymbol{X_e - 3\sigma < X < X_e + 3\sigma}$
Мы проектируем технологию	При нормальных технологических режимах значения параметра не должны уходить за граничные (критические) значения. Чтобы достоверно определить это, измеренные значения должны отстоять от критических не менее чем на 3**σ.** Это и есть допустимая погрешность.
Мы проводим технические измерения.	Нам заранее известна погрешность метода. Если мы точно воспроизводим метод измерения, можно считать, что погрешность равна погрешности метода. Например, измеряя длину стола линейкой с делением 1 мм, мы получим погрешность 1мм.

Способы представления погрешности

В зависимости от решаемых задач используются несколько способов представления погрешности:

- ***Абсолютная погрешность*** – измеряется в тех же единицах что и измеряемая величина. Характеризует величину возможного отклонения истинного значения измеряемой величины от измеренного.
- ***Относительная погрешность*** – отношение абсолютной погрешности к значению величины. Если мы хотим определить погрешность на всем интервале измерений, мы должны найти максимальное значение отношения на интервале. Измеряется в безразмерных единицах.
- ***Класс точности*** – относительная погрешность, выраженная в процентах. Обычно значения класса точности выбираются из ряда: 0,1; 0,5: 1,0; 1,5; 2,0; 2,5 …

Косвенные измерения

При косвенных измерениях мы замеряем одну или несколько величин, которые мы умеем мерить, и по их значениям вычисляем искомую величину. Правило вычисления можно записать в виде формулы:

$$Y=f(x_1,x_2,\dots x_n) \qquad (12)$$

Где Y – искомая величина,

$x_1,x_2,\dots x_n$ – величины, которые мы измеряем непосредственно,

f() – правило вычисления.

Как оценить погрешность Y если известны погрешности прямых измерений $x_1,x_2,\dots x_n$? Сначала рассмотрим проблему в случае одной переменной. На рис 5 приведен график зависимости Y от x. При изменении x на Δx величина Y меняется на ΔY. Если Δx мало ($\Delta x << x$), то значение ΔY мы можем найти с помощью производной f'(x):

$$\Delta Y = (f(x+\Delta x)- f(x)) \approx f'(x)*\Delta x \qquad (13)$$

Так как истинное значение x может отличаться от измеренного и в большую и в меньшую сторону, для определения ΔY нам придется взять не производную f'(x), а ее абсолютное значение |f'(x)|. Если мы хотим найти погрешность косвенного измерения на всем интервале измерений, заменим в формуле 13 величину f'(x) на ее максимальное значение в измеряемом интервале:

$$\Delta Y \approx \max|f'(x)|*\Delta x \qquad (13)$$

В случае нескольких измерений, вместо производной будем использовать частные производные по каждому x_i:

$$\Delta Y \approx \sum(\max|\partial f/\partial x_i|*\Delta x_i) \qquad (14)$$

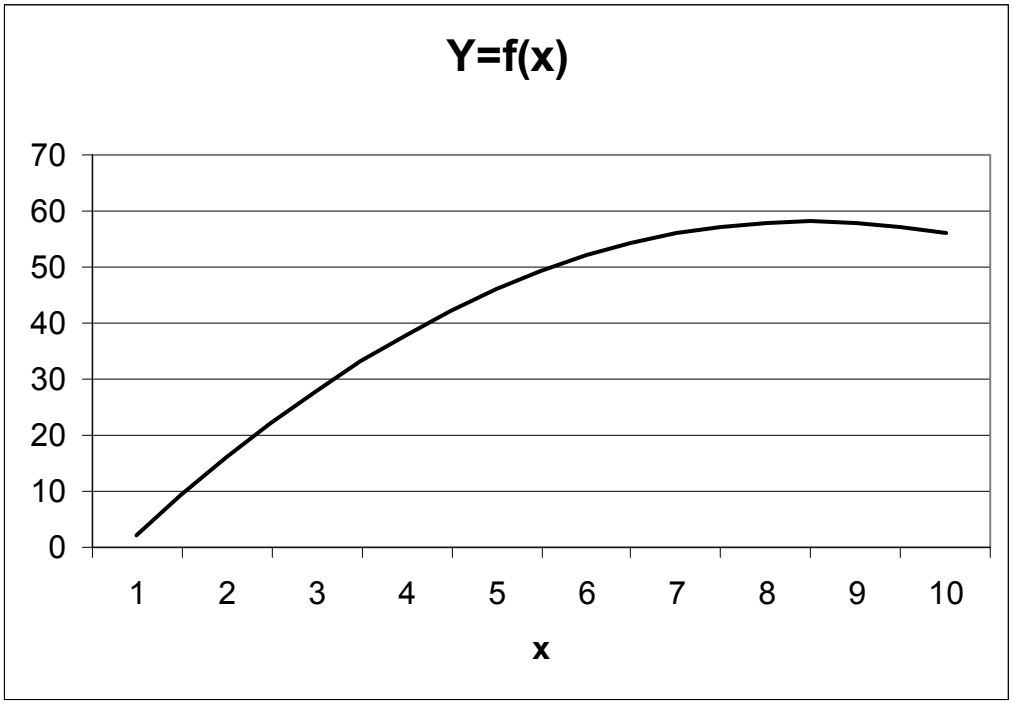


Рис 6. Оценка погрешности косвенного измерения

Выборочный контроль

Инструментальная и методическая ошибка измерения
Точность и достоверность результатов измерения
От чего зависит точность и достоверность выборочного контроля.
Использование планов контроля.

Классификаторы и кодификаторы

Пересортица на складе

Вернемся к нашему примеру со складом готовой продукции. Мы уже обеспечили единство измерений, но проблемы не кончаются. Продукция, поступающая на склад, весьма разнородная. У разных видов продукции разные, порой трудно запоминающиеся, названия, разная цена, и потребители. При ее регистрации легко можно ошибиться и тогда наша программа не сможет отследить правильность поступления и использования продукции. Как решить эту проблему?

Давайте создадим в нашей компьютерной системе справочник видов продукции. При вводе информации о каком либо виде, мы будем искать его название в справочнике и указывать компьютеру о какой продукции идет речь. Такая организация ввода информации исключит ошибки, связанные с вводом неверных наименований продукции.

При появлении нового вида продукции мы должны сначала убедится в том, что это действительно новая продукция, а не другое (неправильное) название продукции, уже имеющийся в нашем справочнике. Если это действительно новый продукт, занесем его название и всю имеющуюся о нем информацию в справочник. И только потом будем использовать новое название наравне с имеющимися.

Вопрос:
К чему приведет отказ от описанного способа ввода, например, если просто разрешить вводить название продукции с клавиатуры?

Терминологическое обеспечение системы управления

Очевидно, проблема правильного ввода и использования терминологии имеется не только на складе. Трудно представить себе последствия путаницы терминологии в системе управления большого предприятия. Чтобы такие сбойные ситуации не возникали, в АСУ предприятия ведутся специальные терминологические справочники. При необходимости, на предприятии выпускается специальный документ: стандарт предприятия (СТП) «Термины и определения».

Определение.
Единство терминологии *– состояние, при котором все участники процесса производства, распределения и потребления одинаково понимают и используют все термины, связанные с данным видом деятельности*.

Итак, на предприятии мы обеспечили одинаковое понимание всеми сотрудниками всех используемых терминов. Но предприятие сотрудничает со многими другими предприятиями в России и за рубежом. Путаница в терминологии может привести к серьезным ошибкам при составлении и выполнении договоров, организации взаимодействия предприятий и т.д.

Еще одна проблема- исполнение законов. Законы, действующие на территории России должны одинаково исполняться всеми ее гражданами. Значит, и пониматься все законы должны всеми одинаково.

Задачи статистики и анализа

Наблюдаемость

Агрегирование информации

Советы молодому программисту
Организация справочников информационной системы

1. Если Вы хотите, чтобы:
a. данные, хранящиеся и используемые в Вашей системе, были сопоставимы;
b. поиск нужной информации не превращался в трудную проблему;
c. ввод типовых данных не был трудоемким и не порождал множество ошибок;
d. информация, хранящаяся в Вашей системе, была согласована с внешними источниками данных;

используйте терминологические справочники!
Для этого:
I. Организуйте таблицу вида:

Код	Термин	Примечание

Заполните вторую колонку всеми терминами, используемыми в Вашей системе. В первой колонке для каждого термина запишите его уникальный (не повторяющийся) код. В третью колонку, при необходимости, внесите комментарии (например, определения терминов).

Если у Вас используются термины, отражающие объекты разной природы, заведите несколько таких таблиц. Например. Вы организуете базу данных по музыкальным CD. В ней будет храниться информация о самих дисках, их хозяевах и людях, которым они дали диски послушать. Заведите две таблицы терминов (справочников наименований):

Код_CD	Название_CD	Примечание
1	Классическая музыка	
2	Популярные песни	
3	Тяжелый рок	

Код_Персон	Персона	Примечание
1	Иванов Петр	Уважает тяжелый рок
2	Петров Иван	Любитель классики

II. Используйте вместо терминов их коды.
Например. В нашем случае, для того чтобы описать кому принадлежит данный CD и у кого он сейчас находится достаточно одной таблицы:

Код_CD	Код_хозяина	Код_держателя
1	2	2
2	1	1
3	1	2

Из таблицы видно, что Петру принадлежит диск классики, который он никому не дал; Ивану принадлежат популярные песни и рок, Диск с роком он дал послушать любопытствующему Петру.
III. При вводе и выводе информации расшифровывайте коды их значениями из соответствующих справочников.
Такой способ организации данных получил название «четвертая нормальная форма».
Именно на такой способ ориентированы современные реляционные системы управления базами данных (СУБД).

2. При вводе информации пользуйтесь терминологическими справочниками.
Например. Если Вам нужно ввести фамилию и номер группы студента создайте форму:

Организуйте программу так, чтобы при нажатии кнопки «V» вызывался соответствующий список. Выбирайте значения только из списка. Если в списке отсутствует нужное значение (например, Вы описываете нового студента), сначала введите его в список (и присвойте ему уникальный код) и только затем, выбрав из списка, используйте это значение.

Обеспечение совместимости

Введение в проблему. Дисковод из Гонконга

Представьте себе, что вам потребовалось обновить Ваш домашний компьютер. Вы покупаете «карту», изготовленную где-то в Гонконге, вставляете ее в свой компьютер и она (о чудо!) прекрасно работает. Чтобы понять, что это действительно чудо, вспомним, по скольким параметрам надо согласовать электронную схему, чтобы она могла правильно работать в компьютере. В таблице приведены основные виды совместимости, необходимой для нормальной работы дисковода и компьютера.

Вид совместимости	Пример
Конструкционная	Дисковод должен влезть в предусмотренное для него место в компьютере и быть надежно закреплен
Электрическая	
разъемы	Входные и выходные разъемы компьютера и дисковода должны соответствовать друг другу.
питание	Напряжение питания, подаваемое на дисковод, а также его качество, должны удовлетворять требованиям дисковода
распайка разъемов	Сигналы, приходящие на ножки выходных разъемов и уходящие с ножек входных разъемов дисковода и компьютера должны соответствовать друг другу.
форма сигналов	Уровень сигналов, которыми обменивается дисковод и компьютер, их продолжительность и форма должны соответствовать друг другу.
Информационная	
код	Способы кодирования информации в устройствах должны соответствовать друг другу.
семантика	Смысл информации, которой обмениваются устройства должен совпадать

Методы обеспечения совместимости

Впервые проблема совместимости возникла при переходе от ремесленного производства к промышленному. Ремесленник, самостоятельно изготавливает изделие от начала до конца. Он может позволить себе использовать уникальные детали, не подходящие ни к какому другому изделию. Но как быть с массовым производством?

Одним из первых упоминаний использования методов совместимости является пример вооружения армии США времен Гражданской войны[3]. Армия остро нуждалась в мушкетах. Но, ни один ремесленник не мог изготовить нужное количество оружия в требуемые сроки. Раздавать заказ по многим ремесленникам командование не хотело, справедливо полагая, что собрать и испытать массу разнородного оружия будет очень трудно.

За выполнение заказа взялся один смелый бизнесмен. Он выбрал самую простую, но надежную конструкцию мушкета, тщательно зарисовал все его детали, сформулировал требования к каждой их них и раздал в виде субподряда другим ремесленникам и фирмам. У себя он оставил только участки контроля и сборки. В договоре о субподряде он заранее оговорил, что оплачивать будет только те детали, которые полностью удовлетворяют преъявленным к ним требованиям. Таким образом, заказ был выполнен в полном объеме и в срок, армия получила добротное и однотипное оружие, а человечество сделало еще один шаг в направлении кооперации и повышения эффективности производства.

[3] Пример взят и книги Дэниела Бурстина «Американцы: национальный опыт».(М.:Прогресс, 1993. 619 с.)

Роль стандартизации в обеспечении совместимости

Как видно из приведенного примера, основной метод обеспечения совместимости – разбиение общего изделия на относительно независимые части и формирование исчерпывающего набора требований к каждой из них. Этого будет достаточно, если изделие не будет совершенствоваться.

Настоящее время характерно частой сменой выпускаемых изделий. Подготовка к выпуску нового изделия связана с конструированием его деталей, разработкой требований к их производству, переоснасткой оборудования и т.д. Это очень сложные и дорогостоящие процессы. Как снизить затраты и время подготовки производства? Наверное, стоит подумать об использовании уже налаженного производства, максимально включив в новое изделие уже выпускаемые детали. Тогда, конструировать и налаживать выпуск придется только для новых деталей.

Такой подход называется «***унификация***». Вместо того чтобы каждый раз разрабатывать новое изделие от начала до конца, мы разрабатываем набор унифицированных деталей и узлов, пригодных для работы во многих уже выпускаемых и будущих изделиях. Время и затраты на разработку и выпуск сокращаются, а качество выпускаемых изделий возрастает. Это происходит потому, что большая часть деталей нового изделия уже проверена временем и выпускается по хорошо отлаженной технологии.

Унификация предъявляет дополнительные требования к конструкции деталей и узлов. Действительно: конструктор унифицированной детали должен не только учесть требования, предъявляемые к ней существующими изделиями, но и предвидеть требования будущих изделий.

Может ли производитель унифицированных изделий что-то изменить в их конструкции или технологии их производства? Может, если модернизированное изделие будет полностью удовлетворять требованиям, предъявленным к унифицированному изделию. Обратите внимание! Изменяется продукция, в которой используются унифицированные изделия. Изменяются сами эти изделия. Но, чтобы унификация приносила пользу, неизменной должна оставаться система требований к унифицированным изделиям.

Для обеспечения совместимости изделий используется метод стандартизации. Перечень требований к изделию (например, к унифицированному узлу) оформляют в виде специального документа: ***стандарта***. Изготовители, берущиеся за производства данного изделия, обязуются выпускать продукцию, полностью соответствующую требованиям стандартов. Разработчики и изготовители изделий, в которых применяются унифицированные изделия, имеют возможность подобрать нужное им изделие и приобрести его у любого изготовителя. При этом совместимость унифицированного изделия с выпускаемым, работоспособность выпускаемого изделия обеспечиваются за счет выполнения требований стандарта.

Пример совместимости: модульный принцип программирования

Рассмотрим, как решаются проблемы совместимости на примере, близком каждому программисту: обеспечения совместимости программных модулей.

Почему возникает такая проблема? Пока Вы пишете коротенькие и достаточно простые программки, Вы без труда можете проследить все связи и все изменения переменных величин. Но, время простых программ прошло. Современный программный комплекс оперирует с тысячами параметров и массивов. Размеры программного кода составляют десятки, а иногда и сотни мегабайт. Чтобы представить себе такие размеры вспомним, что страница текста в формате txt занимает приблизительно 2 кБт. Следовательно, один мегабайт – это книга в пятьсот страниц. Даже набрать на клавиатуре такую огромную программу сложно. А ее еще нужно отладить, заставить правильно

выполнять порученную ей работу и не создавать дополнительной головной боли у пользователя. Кроме того, программа должна быть написана в сжатые сроки, иначе конкуренты нас обгонят и «опоздавшая родиться» программа окажется ни кому не нужной.

Как решаются подобные задачи сейчас? Рассмотрим несколько правил быстрого и эффективного создания программ:

1) К разработке программ привлекается не один, а целая команда программистов.
2) Программа разбивается на относительно независимые части (модули), каждая из которых должна выполнять определенную функцию.
3) Обеспечивается совместимость программных модулей.

Как видим, разбиение программы на модули – типичная задача обеспечения совместимости.

Последовательность реализации модульного принципа программирования

1. Задача разбивается на относительно простые и самостоятельные фрагменты. Критериями качества разбиения являются:
    a. Законченность выделяемых фрагментов. Каждый из них должен до конца решать какую-то, пусть маленькую задачу.
    b. Посильность программирования. Размеры и сложность модуля должны быть такими, чтобы один программист в реальные сроки справился сего написанием и отладкой.
    c. Минимум связей с остальными модулями. Чем меньше зависит наш модуль от остальной части программы, тем меньше проблем возникнет при его совместной отладке.
    d. Проверяемость входных данных и результатов выполнения модуля. К правильности и полноте входных данных должны быть сформулированы четкие требования, выполнение которых должно быть проверено на входе модуля. Ситуации получения неверных данных должны быть учтены и обработаны. Решая определенную задачу до конца, модуль должен выдавать результаты, правильность которых можно логически проверить.
2. Формируются требования к модулям. Для каждой задачи, выполняемой программы должна существовать цепочка последовательно выполняемых модулей, полностью решающая данную задачу. В простейшем случае цепочка может состоять из одного модуля. Требования к модулям формируются по принципу «обратной волны»:
    a. Требования к результату решения задачи являются требованиями к последнему модулю в цепочке, решающей эту задачу.
    b. Анализируется, что (какая информация и инструментарий) необходимы модулю, чтобы обеспечить предъявленные требования.
    c. Часть требований могут обеспечить входная информация и условия выполнения модуля, остальное формулируется как требования к предыдущим модулям.
    d. Описанный анализ проводится последовательно по всем модулям от конца к началу. Если модуль участвует в нескольких цепочках, требования к нему анализируются на совместимость и объединяются. Если требования к модулю не совместимы, необходимо пересмотреть модульную структуру программы.

В результате такого анализа формируется совокупность требований к каждому модулю.

3. Разрабатывается межмодульный интерфейс – правила вызова модулей и передачи им параметров. Информация, необходимая для работы большинства модулей организуется в виде общих областей, глобальных переменных, массивов или баз данных.

После завершения третьего этапа формируются задания на программирование. Только после этого и программисты могут приступать к реализации модулей.

На самом деле, процесс выделения модулей и формирования требований к ним проходит в несколько итераций. От качества выделения модулей зависят трудоемкость и реализуемость программного комплекса, сроки его реализации и качество.

4. После написания отдельных модулей проводится их комплексная отладка. В ходе отладки проверяется возможность совместной работы модулей, правильность решения всех задач программного комплекса.

Вопрос: Как применить метод унификации при выделении и разработке программных модулей?

Что такое стандартизация?

В соответствии с законом РФ "О стандартизации" [?] (см. Приложение 2)

Стандартизация - это деятельность по установлению норм, правил и характеристик (далее - требования) в целях обеспечения:

- безопасности продукции, работ и услуг для окружающей среды, жизни, здоровья и имущества;
- технической и информационной совместимости, а также взаимозаменяемости продукции;
- качества продукции, работ и услуг в соответствии с уровнем развития науки, техники и технологии;
- единства измерений;
- экономии всех видов ресурсов;
- безопасности хозяйственных объектов с учетом риска возникновения природных и техногенных катастроф и других чрезвычайных ситуаций;
- обороноспособности и мобилизационной готовности страны.

Во всех перечисленных задачах нам необходимо определить перечень требований и правил, которым должны подчиняться все, от кого зависит решение данных задач.

Образно говоря, если мы хотим быстро и качественно выполнить какую-то большую работу, мы должны договориться о том, кто и как будет делать свою часть этой работы, чтобы в результате получилось то, чего мы хотели. Чтобы самим не забыть о своих договоренностях, познакомить с ними новых участников нашей работы, а также, чтобы потом разобраться, почему часть работы не получилась, все наши договоренности должны быть оформлены в виде документов. Технические требования к изделиям, правила их изготовления и проверки, маркировки и упаковки, хранения и транспортировки называются нормами. Документы, содержащие нормы, называются нормативными документами или стандартами.

Нормативные методы управления

Методы управления, заключающиеся в создании и применении нормативных документов, называются нормативными методами управления. Когда нужно применять нормативными методами управления?

Разработка, производство и использование достаточно сложных объектов требует, чтобы все участники этого процесса договорились по техническим вопросам и придерживались этих договоренностей в своей деятельности. Договоренности, достигнутые сотрудниками одного предприятия, оформляются в виде стандарта предприятия (СТП). После утверждения руководителем предприятия, требования СТП становятся обязательными для всех сотрудников данного предприятия.

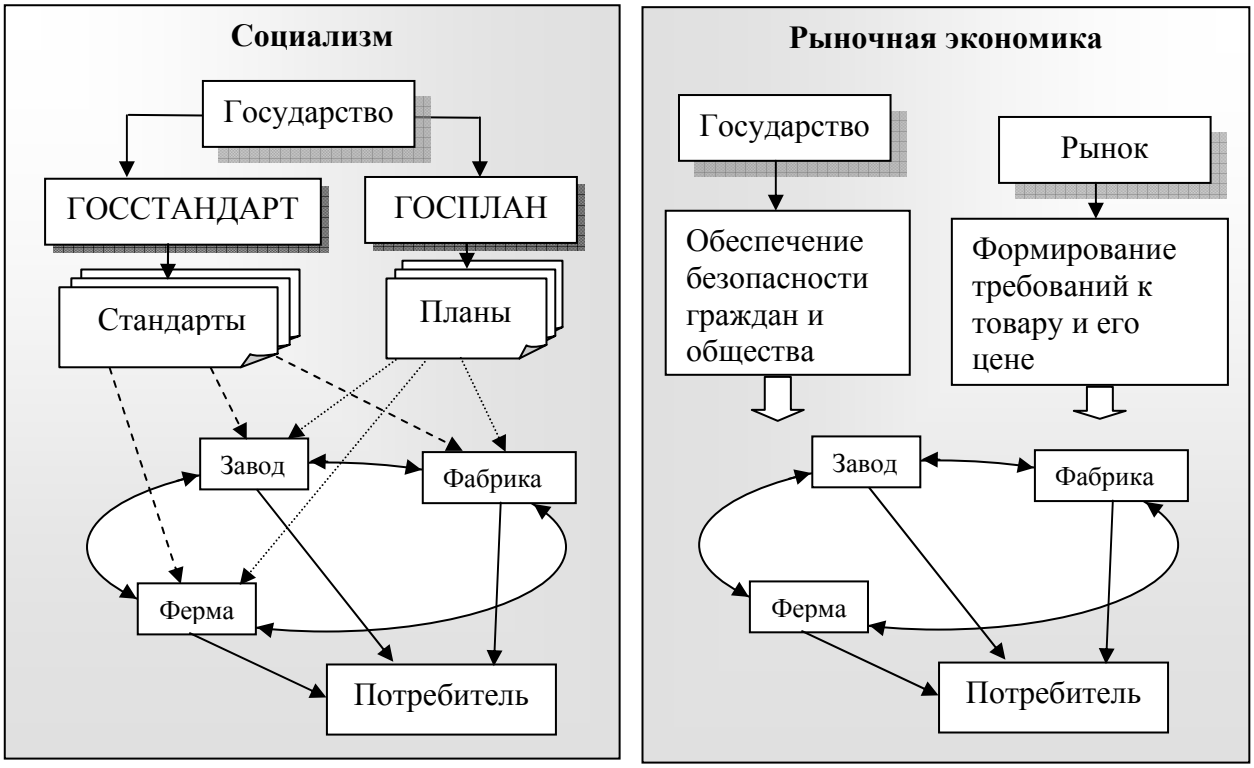


Рис ? Различия применения нормативных методов управления в административной и рыночной системах управления экономикой страны.

Для обеспечения взаимодействия предприятий также требуются стандарты. Совокупность требований, отражающих специфику одной отрасли, оформляется в виде отраслевых стандартов (ОСТов). Требования, специфичные для данного региона, оформляются в виде региональных стандартов. Наконец, требования, касающиеся всех граждан и предприятий страны, оформляются в виде государственных стандартов (ГОСТов).

Для вхождения в мировую экономическую систему, Россия должна присоединиться к международным соглашениям и принять на себя ряд обязательств. Одним из механизмов присоединения страны к мировому сообществу является признание Россией международных стандартов. В этом случае в стране выпускается национальный стандарт, полностью идентичный международному стандарту.

Изменение целей и методов стандартизации при развитии рыночных отношений

В настоящее в России происходят непростые и весьма противоречивые процессы перехода от административных методов управления экономикой к рыночным.

При административной системе только государство могло владеть средствами производства. Фактически, все предприятия страны являлись филиалами одной фирмы – государства. Для обеспечения управления в такой огромной фирме требовались специальные методы (см. левую часть рис. ?).

Государство создало два ведомства: ГОСПЛАН и ГОССТАНДАРТ. ГОСПЛАН осуществлял всеобщее планирование, определял производственные задания всем предприятиям, подбирал для них поставщиков и определял потребителей их продукции.

ГОССТАНДАРТ отвечал за обеспечение единства требований. Он разрабатывал и утверждал стандарты, являющиеся обязательными для всех участников экономического процесса, обеспечивал единство измерений и терминологии.

Таким образом, в социалистической экономике стандарт отражал требования единого хозяина-государства. Устанавливая стандарты и осуществляя контроль за выполнениям их требований, государство управляло своей огромной фирмой.

Как показал опыт, такой способ управления национальной экономикой оказался весьма не эффективным. Сложность задач централизованного управления десятками тысяч предприятий оказалось столь высокой, что приводила к постоянным сбоям и ошибкам планирования и управления народным хозяйством.

В рыночной экономике государство берет на себя значительно меньше обязательств (см. правую часть рис. ?). Важнейшими из них являются обеспечение:

- безопасности граждан и общества;
- законности и порядка взаимодействия всех членов общества.

Формирование требований к товару и его цене обеспечивает рынок. Никому не нужные товары, предлагаемые по завышенным ценам или не обладающие требуемым качеством, просто ни кто не купит. Следовательно, производители этих товаров не будут иметь доход и вынуждены будут или уйти с рынка, или изменить свое отношение к производству.

Таким образом, в рыночной экономике, государство контролирует соблюдение интересов своих граждан, связанных с безопасностью их жизни и здоровья, а также формирует «правовое поле» - правила и законы взаимоотношений граждан, организаций и государства.

Различие в подходах к стандартизации: Официальные и фактические стандарты

В настоящее время существуют два подхода к стандартизации: официальные и фактические стандарты.

В период всеобщей государственной собственности на средства производства проблемами унификации и обеспечения совместимости могло заниматься только государство. Было выпущено множество государственных стандартов (ГОСТов), устанавливающих требования практически ко всей номенклатуре выпускаемой продукции. Это позволило обеспечить достаточно высокий уровень унификации отечественной продукции.

В странах с рыночной экономикой требования государственных стандартов распространяются только на продукцию, закупаемую или производимую государством. Кроме того, требования государственных стандартов обеспечивают безопасность потребляемой в стране (отечественной и импортной) продукции.

Для обеспечения совместимости основные предприятия, выпускающие данную продукцию, разрабатывают собственные стандарты. Предприятия поменьше, заинтересованные в выпуске той же продукции (клонов), вынуждены придерживаться стандартов предприятий-лидеров. Таким образом, стандарты абсолютных лидеров становятся фактическими национальными, а иногда и международными стандартами на данный вид продукции. Яркими примерами фактических стандартов являются стандарты на процессоры фирмы Intel.

Государственная система стандартизации

Цели и задачи

Государственное управление стандартизацией в Российской Федерации осуществляет Комитет Российской Федерации по стандартизации, метрологии и сертификации (Госстандарт России).

ГОССТАНДАРТ:

- формирует и реализует государственную политику в области стандартизации,

- осуществляет государственный контроль и надзор за соблюдением обязательных требований государственных стандартов
- проводит координацию деятельности госорганов;
- взаимодействие с органами власти субъектов Федерации, городов, а также с общественными объединениями и субъектами хозяйственной деятельности,
- организует профессиональную подготовку и переподготовку кадров в области стандартизации
- участвует в работах по международной стандартизации, устанавливает правила применения международных стандартов на территории РФ.

Объектами государственной стандартизации являются: продукция, работы и услуги, имеющие межотраслевое значение. Государственные стандарты должны содержать (обязательные требования подчеркнуты):

- требования к продукции, работам и услугам по их безопасности:
  - для окружающей среды,
  - жизни, здоровья и имущества,
  - пожарной безопасности
  - техники безопасности и производственной санитарии;
- требования по технической и информационной совместимости, а также взаимозаменяемости продукции;
- правила и нормы, обеспечивающие техническое и информационное единство при разработке, производстве и использовании продукции:
  - правила оформления технической документации,
  - допуски и посадки,
  - общие правила обеспечения качества продукции, работ и услуг,
  - сохранения и рационального использования всех видов ресурсов,
  - термины и их определения,
  - условные обозначения,
  - метрологические и другие общетехнические и организационно-технические правила и нормы
- основные потребительские (эксплуатационные) характеристики продукции:
  - методы контроля,
  - требования к упаковке, маркировке, транспортированию, хранению, применению и утилизации продукции.

Обязательные требования (подчеркнуты) должны соблюдаться всеми государственными органами управления и субъектами хозяйственной деятельности.

Соответствие продукции и услуг обязательным требованиям государственных стандартов определяется в порядке ***обязательной сертификации*** продукции и услуг.

Иные требования государственных стандартов становятся обязательными только если они включены договора или в техническую документацию изготовителя (поставщика). При этом соответствие продукции и услуг этим требованиям может определяться в порядке ***добровольной сертификации***.

Структура стандарта

Требования к структуре и содержанию стандарта устанавливает ГОСТ 1.5 «Государственная система стандартизации. Построение, содержание и изложение стандартов». Основными частями текста стандарта являются:

- Наименование стандарта – предназначено для однозначной идентификации документа, должно отражать его смысл и принадлежность системе стандартов;

- Вводная часть (преамбула) – указывается область распространения, уточняется сфера действия стандарта;
- Требования стандарта – содержит формулировки требований к объектам стандартизации;
- Ссылки на другие нормативные документы – дается перечень документов, на которые ссылается данный стандарт.
- Кроме того, в стандартах принято давать разделы:
- «Используемая терминология» - содержащие определения или пояснения специальных терминов, используемых в стандарте;
- «Классификация» - приводится классификация объектов стандартизации.

Как пользоваться стандартом

В таблице приведены ситуации, в которых Вам придется изучать и использовать стандарты.

Ситуация	Зачем и как используются стандарты
Вы производите продукцию, например, программу для ЭВМ	Вы должны определить перечень требований к Вашей продукции, стандартные методики определения соответствия Вашей продукции предъявленным требованиям, требования к маркировке, упаковке, хранению и транспортировке продукции.
Вы заключаете договор на производство продукции	Кроме обязательных требований в договоре могут быть указаны дополнительные требования. Вместо того чтобы самостоятельно формулировать все требования, Вы можете подобрать стандарты, в которых эти требования сформулированы.
Вы тестируете свою продукцию	Все требования к Вашей продукции записаны в технической документации. Вам нужно подобрать стандартные методики выполнения измерений Ваших характеристик. Только если стандартные методики отсутствуют, Вам придется разрабатывать свою методику.
Вы передаете продукцию на сертификацию	Вам нужно определить схему сертификации и согласовать ее с сертифицирующим органом. В схеме должны быть указаны требования, соответствие которым подлежит подтверждению, нормативные документы, содержащие эти требования и методики измерения.

Внимательно прочитайте преамбулу стандарта. Определите, распространяются ли его требования на Вашу ситуацию. Разберитесь, что является объектом стандартизации. В каком отношении находятся объект стандартизации, указанный в стандарте и Ваша ситуация. Если Ваш объект является частным случаем, объекта, указанного в стандарте, требования стандарта распространяются на него.

Определите, все ли требования стандарта являются обязательными в Вашей ситуации. Кроме обязательных для всех требований, Вам следует учесть требования, указанные в Договорной документации.

Сертификация

Вы написали хорошую программу. Она реализует оригинальный алгоритм, позволяющий просто и быстро решать очень важные задачи. Вам Ваша программа очень нравится. Но как убедить пользователя в том, что Ваша программа – это именно то, что ему нужно? Вот если бы кто-то, к чьему мнению прислушивается пользователи, подтвердил, что Ваша программа действительна хороша! Тогда бы, рекламируя свою

программу, Вы могли бы сослаться на мнение уважаемого эксперта, что позволило бы Вам быстрее убедить пользователей приобрести именно Вашу программу.

Сущность сертификации

Дословный перевод термина «***сертификация***» - «Я подтверждаю». Любая сертификация – это подтверждение соответствия объекта сертификации предъявленным к нему требованиям. В процессе сертификации можно выделить следующие элементы (см. таблицу)

Элемент	Роль
Объект сертификации	Объект, свойства которого подтверждаются
Заказчик	Хозяин объекта сертификации, государство или третье лицо, потребовавшее проведение сертификации.
Цель сертификации	Зачем проводится сертификация? Как будут использованы ее результаты?
Требования	Перечень свойств объекта сертификации, наличие которых подтверждается в процессе сертификации. Требования могут быть определены законодательно или предъявлены Заказчиком.
Орган сертификации	Государственное учреждение или частная фирма, проводящая сертификацию
Схема сертификации	Правила проведения сертификации, включающие перечень подтверждаемых требований, методику их контроля и подтверждения, а также вид и юридический статус выдаваемого документа.
Сертификат	Документ, выдаваемый органом сертификации, подтверждающий соответствие объекта сертификации, предъявленным к нему требованиям.

Требования к безопасности и качеству

Как уже говорилось, в рыночной экономике государству вовсе не обязательно контролировать все требования к продукции, ее производству и потреблению. Закон разделяет требования на:

Обязательные (для государства) – подтверждать выполнение которых должен государственный орган;

Необязательные – выполнять которые исполнитель должен только, если они указаны в договоре.

Обязательные требования включают требования к безопасности производства и потребления продукции, а также требования к совместимости. Необязательные требования, в основном, требования к качеству продукции. Правила проведения сертификации обязательных и необязательных требований определяются в законе РФ «О техническом регулировании***»***[4].

Документальное удостоверение соответствия продукции или иных объектов требованиям технических регламентов, положениям стандартов или условиям договора, получило название ***подтверждение соответствия***. Обязательное подтверждение соответствия осуществляется в формах:

принятия декларации о соответствии (далее - ***декларирование соответствия***);

обязательной сертификации.

[4] Федеральный закон N 184-ФЗ принят 27 декабря 2002 года. Вступил в силу 27 июня 2003 г. (через 6 месяцев после опубликования).

Обязательная сертификация

Организация и проведение работ по обязательной сертификации возлагаются на ГОСТАНДАРТ, а в случаях, предусмотренных законодательными актами Российской Федерации в отношении отдельных видов продукции, могут быть возложены на другие федеральные органы исполнительной власти. На рис. ? приведена схема формирования требований к конкретному объекту при его обязательной сертификации.

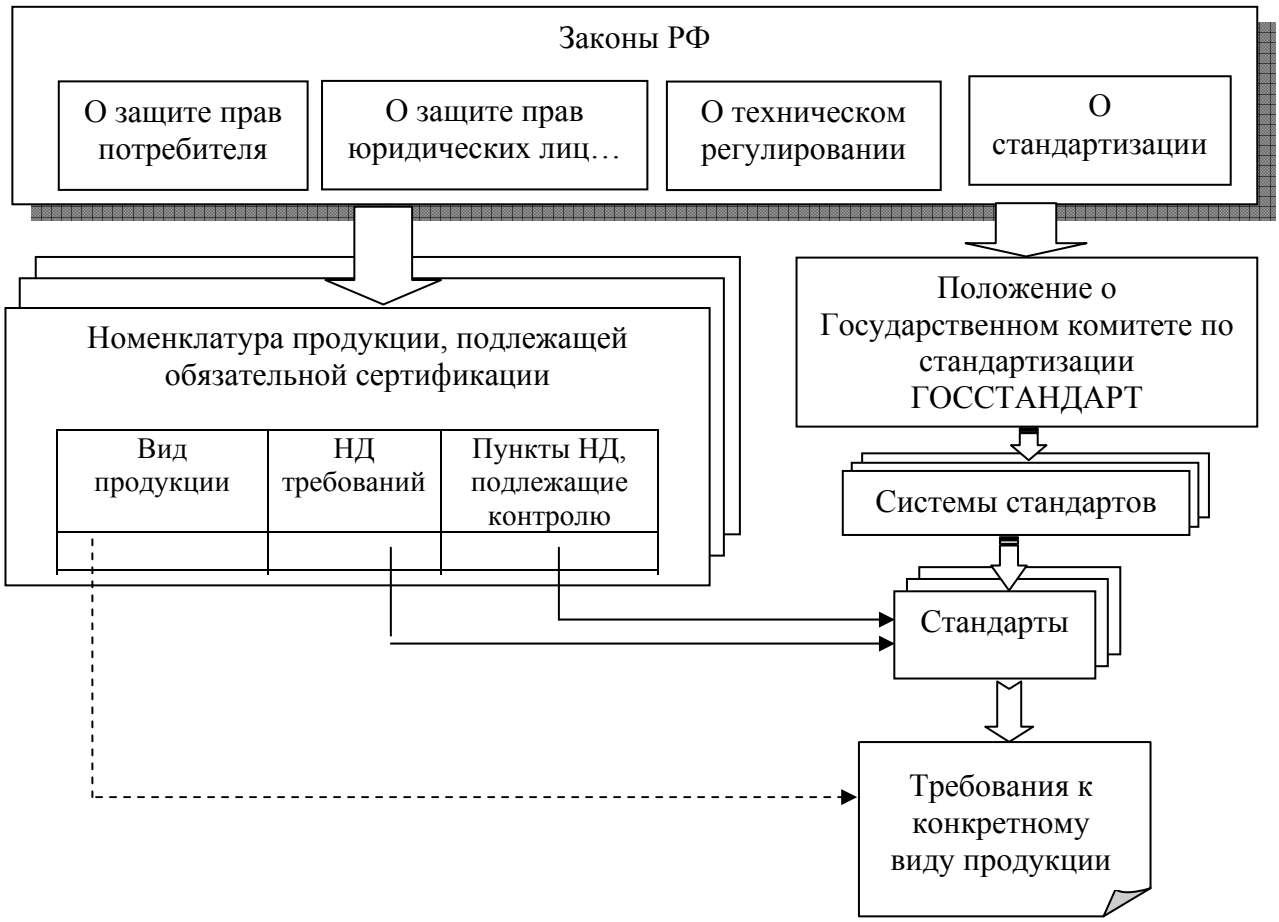


Рис ?. Нормативно-правовая база, устанавливающая требования к номенклатуре и характеристикам продукции, подлежащей обязательной сертификации

Федеральный закон «О техническом регулировании»

Закон определяет основные понятия, устанавливает правила и требования государственного контроля технических характеристик производства и выпускаемой продукции.

Техническое регулирование - правовое регулирование отношений в области установления, применения и исполнения обязательных требований к продукции, процессам производства, эксплуатации, хранения, перевозки, реализации и утилизации, а также в области установления и применения на добровольной основе требований к продукции, процессам производства, эксплуатации, хранения, перевозки, реализации и утилизации, выполнению работ или оказанию услуг и правовое регулирование отношений в области оценки соответствия.

Технический регламент - документ, который принят международным договором Российской Федерации, ратифицированным в порядке, установленном законодательством Российской Федерации, или федеральным законом, или указом Президента Российской

Федерации, или постановлением Правительства Российской Федерации и устанавливает обязательные для применения и исполнения требования к объектам технического регулирования (продукции, в том числе зданиям, строениям и сооружениям, процессам производства, эксплуатации, хранения, перевозки, реализации и утилизации).

Безопасность продукции, ***процессов производства, эксплуатации, хранения, перевозки, реализации и утилизации (далее - безопасность)*** - состояние, при котором отсутствует недопустимый риск, связанный с причинением вреда жизни или здоровью граждан, имуществу физических или юридических лиц, государственному или муниципальному имуществу, окружающей среде, жизни или здоровью животных и растений.

Риск - вероятность причинения вреда жизни или здоровью граждан, имуществу физических или юридических лиц, государственному или муниципальному имуществу, окружающей среде, жизни или здоровью животных и растений с учетом тяжести этого вреда.

Декларирование соответствия - форма подтверждения соответствия продукции требованиям технических регламентов.

Сертификат соответствия - документ, удостоверяющий соответствие объекта требованиям технических регламентов, положениям стандартов или условиям договоров.

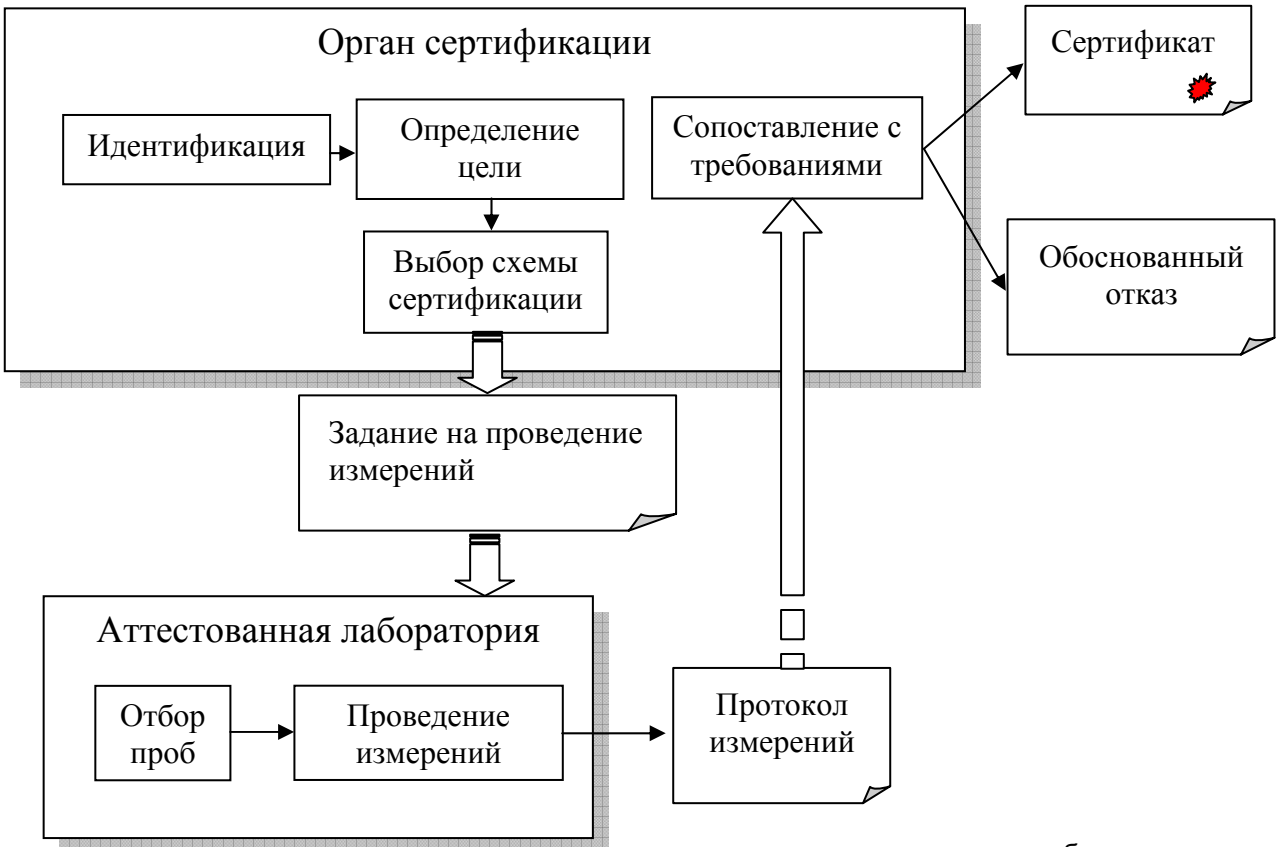
Добровольная сертификация

Роль рынка в повышении качества продукции и услуг

Смысл добровольной сертификации. Торговля рисками

Добровольное подтверждение соответствия осуществляется по инициативе заявителя на условиях договора между заявителем и органом по сертификации. Добровольное подтверждение соответствия может осуществляться для установления соответствия национальным стандартам, стандартам организаций, системам добровольной сертификации, условиям договоров.

Объектами добровольного подтверждения соответствия являются продукция, процессы



производства, эксплуатации, хранения, перевозки, реализации и утилизации, работы и услуги, а также иные объекты, в отношении которых стандартами, системами добровольной сертификации и договорами устанавливаются требования.

Орган по сертификации:

осуществляет подтверждение соответствия объектов добровольного подтверждения соответствия;

выдает сертификаты соответствия на объекты, прошедшие добровольную сертификацию;

предоставляет заявителям право на применение знака соответствия, если применение знака соответствия предусмотрено соответствующей системой добровольной сертификации;

приостанавливает или прекращает действие выданных им сертификатов соответствия.

Организационная структура

Система добровольной сертификации может быть создана юридическим лицом и (или) индивидуальным предпринимателем или несколькими юридическими лицами и (или) индивидуальными предпринимателями

Схема проведения сертификации

Последовательность сертификации (картинка)

Рис. ?. Последовательность проведения сертификации

Постановка задачи на программирование

Что такое постановка задачи

Современный компьютер – мощный инструмент, облегчающий работу с информацией, позволяющий одному человеку быстро и качественно выполнить работу, ранее непосильную и для десятков сотрудников. Однако очень часто компьютер используется в офисе только как печатающая машинка. Это объясняется тем, что сотрудники не умеют использовать возможности компьютера для решения своих задач. Собственно говоря, работа программиста заключается в придумывании и создании новых инструментов и способов применения компьютера для решения задач пользователя.

Чтобы желание использовать компьютер превратилось в готовую программу, должны поработать люди различных специальностей.

- В чьей-то голове должна родится идея программы.
- Идея должна быть обсуждена и оценена с точки зрения реализуемости, практической полезности и коммерческого успеха.
- Чтобы программа продавалась (внедрялась) должен быть определен пользователь программы, исследованы его особенности и предпочтения, определены его проблемы.
- Странно было бы разрабатывать новую программу, если аналогичные программы, решающие нашу задачу, уже существуют. Поэтому необходимо поискать программы–аналоги задуманной.
- Если все-таки придется писать свою программу, стоит продумать ее общую структуру, определить набор задач, которые она способна решать и подобрать методы решения этих задач.
- Современные программы весьма громоздки, а время на их разработку ограничено. Поэтому, разработку серьезных программ, обычно поручают нескольким программистам. Для этого весь проект необходимо представить как систему взаимодействующих программных модулей. Такое представление полезно и при разработке сложной программы одним исполнителем, так как систематизирует его работу и исключает множество ошибок, связанных со сложностью программного кода.
- Разбив программу на модули и определив четкие требования к каждому из них, мы можем приступить к написанию программного кода.

Как видим, от идеи программы до написания кода надо сделать множество дел. Вся эта совокупность дел, предшествующая программированию получила название ***постановка задачи***.

Этапы постановки задачи

Рассмотрим более подробно все этапы (дела), из которых состоит постановка задачи на программирование.

Идея программы

Жизнь новой программы начинается с идеи использования компьютера для новой задачи. Идея может родиться у программиста наблюдающего, как мучаются его коллеги, у самих информационных тружеников, просто у любознательного и умного человека. Одна и та же идея может быть реализована несколькими программами, в одной программе может быть заложено несколько идей. Но, без хорошей идеи не будет и программы.

Например. Замечательная идея Питера Нортона, автора "Norton Commander", в последствии была повторена в десятке различных программ, разбивающих экран на два окна со списками файлов и позволяющих в наглядном виде осуществить процедуры копирования и переноса файлов.

Кто Ваш пользователь и зачем ему нужна эта программа?

Даже из замечательной идеи программы не получится, если такая программа никому не нужна. Отбросим экзотический случай «программирования для души» (программист, прежде всего профессионал!). Во всех остальных случаях программа пишется, если она кому–то нужна.

Для кого предназначена программа, которую Вы собираетесь написать? Какие проблемы стоят перед ее будущим пользователем? Как Ваша программа поможет ему решить эти проблемы? От ответов на эти вопросы зависит дальнейшая постановка.

Если Ваш пользователь еще не имеет достаточного опыта применения компьютера в своей работе, интерфейс Вашей программы должен быть максимально ему понятный, система подсказок должна содержать подробные справки о выполнении всех операций. Модули ввода данных должны содержать специальные алгоритмы проверки вводимых данных. Система управления программой должна обеспечивать мощную «защиту от дурака[5]» и т.д.

Например. Ваша программа – база данных, рассчитанная на не очень квалифицированного пользователя. Название формы: «Введите поля данных» приведет его в замешательство. В этом случае Вам лучше сформулировать содержательный вопрос. Например: «Введите свою фамилию, имя и отчество».

Неквалифицированный пользователь может заполнять числовые поля буквами или другими символами. Поэтому в программе необходимо предусмотреть проверку типа данных. Если тип вводимых данных не соответствует требуемому, программа должна вежливо объяснить пользователю: в чем его ошибка и попросить ввести правильные данные.

Наконец, если в Вашей программе предусмотрены какие-то необратимые действия, например, стирание информации, программа должна предупредить о последствиях выполнения такого действия и попросить подтверждения на его исполнение. Еще лучше организовать «откат» - запомнить состояние перед действием и, при необходимости, вернуться к нему.

Если Ваш пользователь – узкий специалист и Ваша программа предназначена для решения специальных задач, терминология, используемая в интерфейсе программы,

[5] В термине «защита от дурака» нет ничего обидного. Он появился в период Второй мировой войны. Снабжая армию США современным вооружением, специалисты столкнулись с проблемой. Плохо обученные солдаты ломали новую технику гораздо эффективнее, чем артиллерия противника. Для устранения этой беды были разработаны специальные методы, обеспечивающие сохранность техники при неверных действиях персонала.

должна быть взята из предметной области пользователя. Представление входных и выходных данных, а также хода решения задачи должны соответствовать нормам, принятым в данной предметной области.

Например. Вы разрабатываете программу по бухгалтерскому учету. Вам придется овладеть бухгалтерской терминологией. То, что программист воспринимает как последовательность операций со счетами, бухгалтер называет «проводкой» и именно так должна называться эта последовательность в Вашей программе.

В бухучете разработана специальная система документации, строго регламентирующая форму документов, последовательность полей и используемую терминологию. Бухгалтера привыкли к этим формам, а Закон требует их строгого соблюдения. Поэтому: входные и выходные формы Вашей программы должны строго соответствовать требованиям к документам. Терминологические справочники, используемые в программе должны содержать все разрешенные термины и только их.

Поиск программ-аналогов

В настоящее время в мире работают миллионы программистов. Задачи, которые они решают, очень часто схожи с Вашими задачами. Поэтому Вам стоит познакомится с программами, разработанными другими программистами для решения аналогичных задач.

Вполне может оказаться, что и для решения Вашей задачи уже есть готовая программа. Вы можете ей воспользоваться. Но для этого Вам нужно выполнить необходимые условия использования объектов чужой интеллектуальной собственности. Например, приобрести лицензию. Несмотря, на то, что лицензионное программное обеспечение стоит дорого, Вы можете сэкономить много сил и времени, отказавшись от разработки собственной программы.

Еще одним аргументом в пользу поиска готовых программ является уже имеющийся опыт их эксплуатации. Как бы не была продумана постановка Вашей программы, как бы тщательно Вы не обрабатывали в ней все сбойные варианты, жизнь оказывается богаче на сюрпризы, чем подсказывают Вам Ваше воображение и опыт. В программе, которая уже применялась на практике, как правило, уже учтены наиболее часто встречающиеся реальные ситуации, в том числе, приводящие к сбою или неверным вычислениям.

Часто бывает, что чужую программу можно использовать только частично, или после ее модификации. Если у Вас есть исходный текст программы, а вносимые изменения не существенны, Вам, наверное, стоит рискнуть и модифицировать чужую программу для своих нужд. Однако, если изменения существенны (например, затрагивают логику данных или вычислений) трудоемкость отладки таких изменений будет сопоставима, а часто и превышать трудоемкость написания и отладки собственной программы. В этом случае можно воспользоваться только отдельными программными модулями, для которых Вы хорошо представляете алгоритмы их работы и способ обмена данными.

Даже если Вы не нашли никаких программ, подходящих для решения Вашей задачи, время было потрачено не зря. Вы познакомились с новыми методами и программистскими приемами решения задач, схожих с Вашей. Вы убедились, что Ваша задача уникальная, однако в и ней можно использовать некоторые приемы, позаимствованные из других задач. Наконец, Вы повысили свой кругозор и программистскую культуру!

Разработка общей структуры программы

Сценарий работы с программой

Как пользователь будет использовать вашу программу? Хорошая программа должна помогать пользователю решать его задачи и, по возможности, не создавать дополнительной головной боли.

Чтобы лучше понять, как должна быть устроена программа, попробуем представить ее взаимодействие с пользователем. Вот пользователь включает компьютер и вызывает Вашу программу. Что он должен увидеть на экране? Что он может делать дальше? Как должно быть организованно основное меню программы, чтобы пользователю было удобно с ним работать? Описывая последовательность действий пользователя и программы, мы постепенно строим ***сценарий работы с программой***. По ходу дела, мы определяем: какие функции должна выполнять программа, когда и какие сообщения она должна выдавать.

Продумывание сценария поможет Вам написать действительно удобную и полезную программу. Сформулируем свойства, которыми должна обладать хорошая программа:

Программа должна быть удобной. Инструменты, реализуемые программой должны быть всегда «под рукой». Форма входных данных должна соответствовать имеющейся документации. Форма и содержание выходных данных – требованиям к ним.

В своей работе программа должна обращаться к пользователю только тогда, когда ситуация слишком сложная для формального анализа.

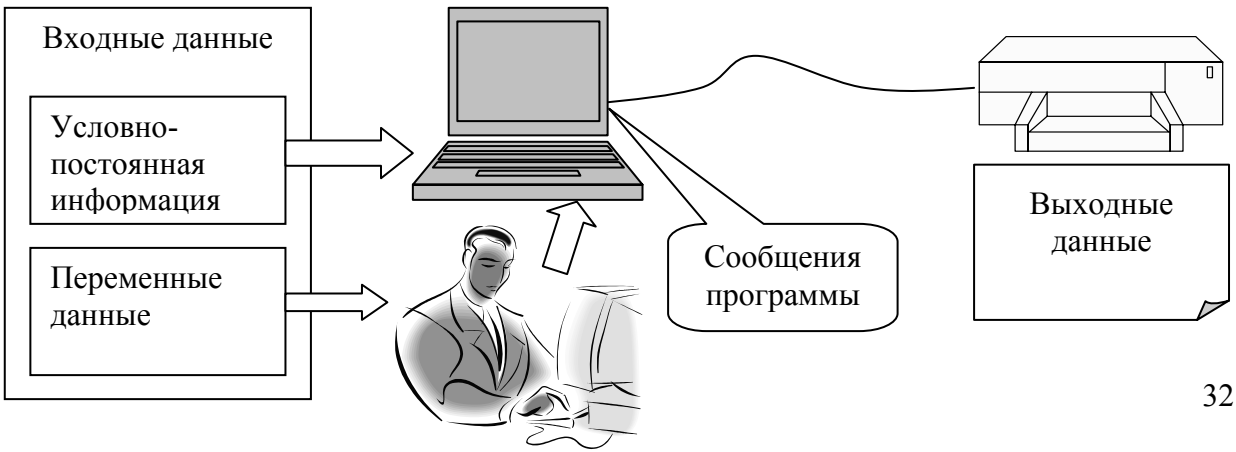
Например. Если типовая задача решается последовательностью вызова программных модулей (всегда одной и той же), лучше нарисовать еще одну кнопку (или пункт меню) и «навесить» на нее вызов необходимой последовательности. Чем больше основное меню программы будет напоминать список решаемых пользовательских задач, тем лучше и удобнее Ваша программа.

Пользователь должен доверять программе. Ход выполнения программы и полученные результаты должны комментироваться, чтобы у пользователя не возникло подозрения, что эта «дьявольская железяка опять сделала чего ни будь не то». Сама программа должна проверять, вводимые в нее данные. В случае обнаружения ошибки или даже подозрения на ошибку, программа должна попросить пользователя перепроверить введенные данные. Ибо человек, имеет право ошибиться, а Ваша программа – нет.

Данные и функции

Разработка сценария позволит Вам определить перечень данных (информации), необходимых для работы программы. Составьте список необходимых данных.

Следующий шаг – мы должны определить источники, откуда можно взять необходимую информацию (см. рис.?). Часть информации необходимо ввести в компьютер заранее. Это ***входные данные программы*.** Некоторая информация может быть сгенерированна самим компьютером в ходе вычислений и анализа программы. Вновь сгенерированная информация может служить для организации диалога с



пользователем (это ***сообщения программы***) или содержать конечные результаты работы с программой. Это ***выходные данные программы.***

Рис ? Классификация данных, используемых программой

Входные данные расклассифицируем на ***условно постоянную информацию*** и ***переменные данные***.

К условно постоянной информации относятся данные, которые не надо менять при каждом обращении к программе. Это могут быть характеристики объекта расчетов (например, твердость стали в программе расчета стальных конструкций), нормативные данные (например, формы бухгалтерских документов) или другая не часто меняющаяся информация. Ввод условно-постоянной информации обычно происходит при создании программы. При постановке задачи, необходимо определить: из каких источников будет взята условно-постоянная информация, насколько достоверны эти источники и как часто нужно обновлять эти данные. Некоторые программы снабжаются специальными подсистемами, позволяющими редактировать условно-постоянную информацию.

Переменные данные нужно вводить в компьютер при каждом обращении к программе. Обычно этим занимается пользователь. Он же отвечает за достоверность вводимых данных. В программе должна быть специальная подсистема ввода данных. Для исключения ошибок, желательно организовать входной контроль данных.

Сценарий работы программы позволит Вам сформулировать набор функций, которые необходимо реализовать в программе. Для этого просмотрим каждый шаг сценария. Выясним, что должен сделать компьютер на этом шаге.

Например. Ваша программа позволяет решать систему линейных уравнений методом Гаусса. Вы разработали сценарий программы, состоящий из четырех шагов:

1. На первом шаге Вы указываете программе: сколько переменных Вам надо найти.
2. Затем Вы вводите значения переменных.
3. Программа определяет, существует ли решение у данной системы и если да, то решает ее, а если нет, предлагает пользователю изменить введенные данные.
4. Вы или изменяете данные, или заканчиваете работу с программой.

Рассмотрим действия программы на каждом шаге.

1. Программа должна поприветствовать пользователя, объяснить ему свое назначение и дать возможность ввести число решаемых уравнений.
2. Программа должна позволить ввести коэффициенты уравнений и свободные члены. При этом количество коэффициентов должно быть равно числу переменных, а свободный член, пусть даже равный нулю, должен быть у каждого уравнения.
3. Программа вычисляет главный определитель системы уравнений и если он не равен нулю, вычисляет корни. Иначе, программа формирует сообщение пользователю и предоставляет ему возможность изменить исходные данные.

Выделенный набор функций называется ***пользовательскими функциями***. Если набор сформирован верно, пользовательский функций должно хватить для решения всех задач программы. Но завершать постановку и приступать к программированию выделенных функций еще рано. Очень часто для реализации нескольких пользовательских функций компьютеру приходится выполнять одни и тоже действия.

Попробуйте выделить в пользовательских функциях повторяющиеся действия. Программную реализацию таких действий организуйте в виде подпрограмм. Укажите параметры, которые нужно передавать каждой подпрограмме (входные параметры) и параметры, которые подпрограмма передает в вызвавшую ее программу (возвращаемые параметры). После программирования этих подпрограмм в языке, которым Вы

пользуетесь, у Вас как бы появятся новые операторы и функции. Теперь программную реализацию пользовательских функций можно представить как последовательность применения стандартных операторов языка и вновь созданных подпрограмм.

Логическое проектирование

Настало время соединить вместе наши требования к информации и набору функций. Каждый программный модуль должен быть настроен на восприятие (например, прочтение) информации, необходимой для его работы. Если наша информация не систематизирована, данные не имеют четкой структуры, программировать модули крайне сложно. Поэтому, перед программированием, следует систематизировать информацию и, по возможности, уменьшить (унифицировать) разнообразие форм ее хранения.

Самый эффективный способ систематизации информации – представление ее в виде описаний объектов и признаков. Если наша программа имеет дело с большим количеством одинаково описанных объектов (например, база данных студентов, для каждого из которых указана фамилия, имя и отчество, пол и возраст), мы можем организовать эту информацию в таблицу, каждая строчка которой – описание одного объекта, а столбец- перечень значений одного признака.

Таблица ?

Описание студентов

Код_Сту	**Фамилия**	**Имя**	**Отчество**	**Пол**	**Год рождения**
1	Иванов	Иван	Иванович	Муж.	1986
2	Петрова	Елена	Владимировна	Жен.	1986

Если мы имеем дело с несколькими видами объектов, построим такие таблицы для каждого вида. В нашем примере студенческой базы данных придется построить таблицы для специальностей, дисциплин, преподавателей и т.д.

Таблица ?

Описание специальностей

Код_Спец	**Специальность**	**Сокращенное название**	**Квалификационные требования**
1	Программирование вычислительных систем	ПВС	
2	Технология машиностроения	ТМС	
3	Механика и аппараты пищевых производств	МПП	

Такие таблицы называются справочниками объектов. В справочниках обязательно должны присутствовать столбцы кодов и названий объектов. Кроме того, в них можно добавить столбцы (признаки), свойственные только этому объекту (в нашем примере: пол и год рождения).

Аналогично опишем другие объекты, используемые в нашей программе:

Таблица ?

Описание предметов (дисциплин)

Код_Дис	**Дисциплина**	**Содержание**
1	Математика, часть I	Функции, дифференцирование, интегрирование, аналитическая геометрия…
2	Физика, часть I	Механика: кинематика, динамика, вращательное движение…
…	**…**	**…**

Таблица ?

Список групп

Код_К	Курс	Группа
1	1	ПВС 11
2	1	ПВС 12
…	…	…

Таблица ?

Список преподавателей

Код_Преп	Фамилия	Имя	Отчество	Должность	Кафедра
1	Кац	Альберт	Маркович	Зав. кафедрой	КИТФ
…	…	…			

Тот факт, что студент Иванов поступил в институт в 2001 году на специальность ПВС, можно записать в таблицу:

Таблица ?

Таблица соответствия
«Студент-специальность-год поступления»

Код_Ст	*Код_Спец*	Год поступления
…	…	…
1	1	2002
…	…	…

В этой таблице можно указать дату поступления и специальность для каждого студента. Такие таблицы называются таблицами соответствия. Характерной чертой таблицы соответствия является связи ее столбцов со справочниками объектов с помощью кодов. Полный перечень справочников и таблиц соответствия (рис ?) с указанием связей между ними называется ***логическая модель данных***.

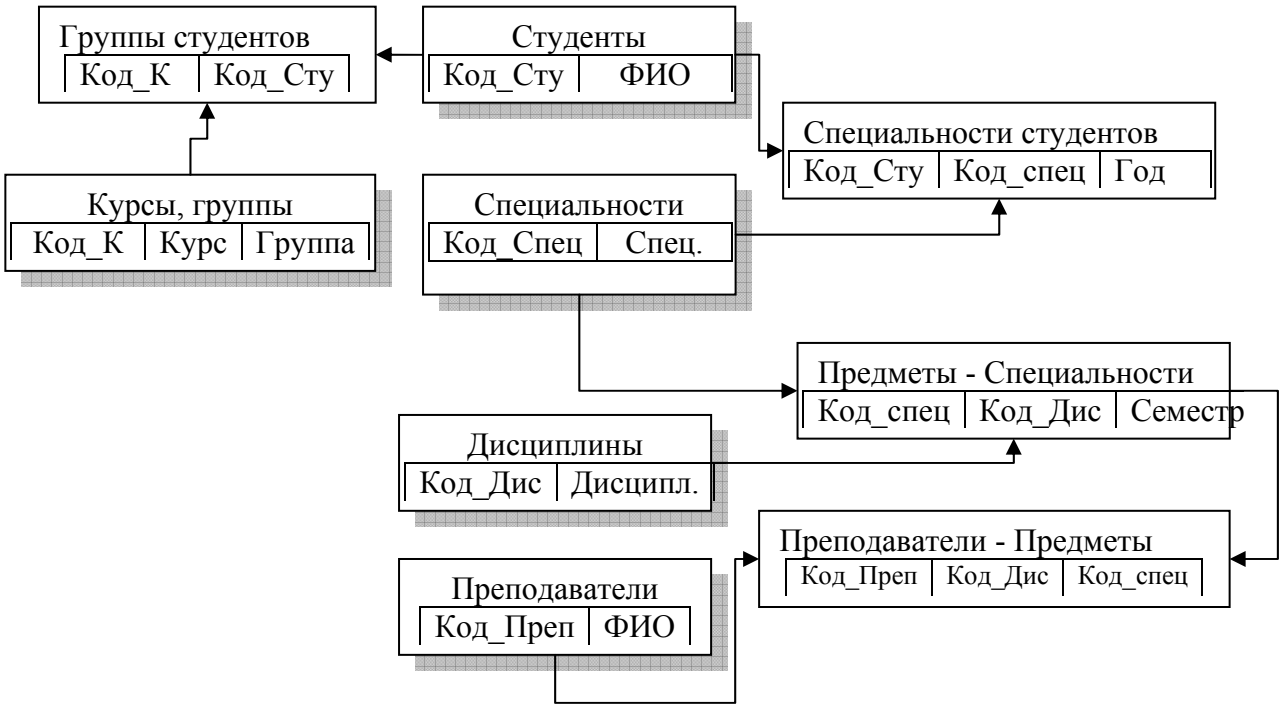


Рис ?. Логическая модель данных

И так, нам удалось представить наши данные в виде справочников объектов и таблиц отношений между ними. Такая организация данных носит название ***реляционная модель*** (от английского слова relation – отношение). Метод систематизации исходной информации в подобные таблицы называется ***нормализацией информации***, а форма такой организации информации получила название ***четвертая нормальная форма***.

В настоящее время большинство ***систем управления базами данных*** (СУБД) и языков программирования ориентированны на реляционную модель данных. Опыт программирования многих систем показывает, что отказ от реляционной модели данных, отклонение от четвертой нормальной формы резко повышает сложность разработки программных модулей, снижает их эффективность, надежность и устойчивость работы.

Оценка полноты логической модели данных и структуры программных модулей

Последним шагом разработки общей структуры программы является проверка ее правильности. Рассмотрим перечень выделенных нами программных модулей. (Напоминаю, мы их еще не программировали, а просто сказали, что такие модули должны быть в нашей программе и выполнять указанные действия). Проверим, хватит ли нам этого набора модулей для реализации всех задач, поставленных перед программой. Для этого попробуем построить решение каждой из задач программы в виде последовательности работы программных модулей. Это напоминает детский конструктор. Хватит ли у нас деталек нужной конфигурации, чтобы собрать задуманное? Отличие только в том, что один и тот же модуль можно использовать многократно, в нескольких задачах[6]. Если все задачи могут быть представлены таким образом, наш набор достаточно полный и можем двигаться дальше.

В ходе проверки полноты набора модулей мы получаем еще одну полезную информацию. Мы видим, какие модули взаимодействуют друг с другом. Работу никогда не взаимодействующих модулей не стоит согласовывать, а значит, требования к ним можно ослабить.

Следующая проверка полноты и удобства представления информации. Рассмотрим работу всех программных модулей. Достаточно ли информации, учтенной в нашей логической модели для работы каждого модуля? В удобной форме нужная информация представлена? Если информации не хватает или для ее использования отдельными модулями требуются серьезные преобразования, попробуйте пересмотреть логическую модель данных. Хорошо сконструированная модель данных существенно сократит число проблем и время, которое придется потратить на программирование модулей.

Проектирование интерфейса программы

Система меню

Хорошее понимание сценария использования Вашей программы позволит Вам сконструировать удобную систему управления программой в виде дерева меню. Главные веточки этого дерева – основное меню, должны соответствовать основным задачам или режимам работы программы. Более мелкие веточки, вырастающие из главных – подменю предназначены для уточнения параметров работы. «Листиками» на этом дереве будут конкретные действия (функции подпрограммы).

[6] Очевидно, что чем чаще используются одни и те же модули, тем меньше будет общее число модулей, а значит, легче запрограммировать наш проект.

Хорошо организованное меню как бы вступает с пользователем в диалог, последовательно уточняя, что ему нужно и предлагая варианты возможных действий. Структура дерева, принадлежность подменю пунктам основного меню должна быть интуитивно понятной пользователь и соответствовать его представлениям о решаемых задачах. Названия пунктов меню должны соответствовать вызываемым действиям, быть привычными и понятными пользователю.

Инструменты, которые часто используются должны быть на виду: в основном меню или хотя бы в высших уровнях подменю.

Контекстная подсказка

При работе с программой у пользователя может возникать множество вопросов. По характеру, вопросы можно разделить на 3 типа:

- Что это такое (терминологические вопросы)
- Для чего это нужно (методические вопросы)
- Как это сделать (технические вопросы)

Терминологические вопросы возникают при встрече пользователя с новым незнакомым термином или в случае непонимания текста, вызванного непривычным использованием терминологии. Исследования психологов показали, что в до 80% случаях непонимания текста, причина незнание или неверная интерпретация терминов, используемых в тексте. Поэтому, контекстную подсказку стоит снабдить ***глоссарием*** – списком используемой терминологии с пояснениями или определениями каждого термина.

Методические вопросы возникают тогда, когда пользователь не понимает: для чего предназначена данная программа и как с ее помощью можно решить его конкретную проблему. Чтобы быть понятной пользователю, методическая подсказка должна содержать теоретическую часть, доступно описывающую идею программы и методы, используемые в программе. Кроме этого, следует описать методики решения типовых задач с помощью данной программы. Пользователь далеко не всегда самостоятельно может понять причину своего непонимания. Поэтому хорошая подсказка должна иметь систему гиперссылок, постепенно отсылающих пользователя от конкретных вопросов, которые он не понимает, к все более общим вопросам методики.

Технические вопросы связаны с назначением конкретных органов управления (клавиш, кнопок, меню и т.д.), а также с рекомендациями (инструкциями) по выполнению конкретных действий. Подсказки по техническим вопросам ориентированы на оператора – человека, который непосредственно управляет программой.

К сожалению. Довольно часто программисты «забывают» написать методическую и терминологическую части подсказок. Если программа достаточно сложная и использует нетривиальные методы, отсутствие этих подсказок приводит к непониманию и множеству недоразумений при ее использовании. Примером хороших контекстных подсказок являются Help’ы к MS Office.

Ниже даны рекомендации по формированию контекстной подсказки. Принята следующая терминология:

Топик – раздел подсказки, посвященный одному вопросу. Топик имеет заголовок и идентификатор. Большинство современных средств программирования позволяют организовать контекстный вызов топика из определенного места программы (например, из окна).

Рабочий топик – посвящен техническим вопросам работы с конкретным окном программы.

Методический топик – посвящен вопросам теории и методики решения задач пользователя с помощью программы.

1. Общие рекомендации:

1.1. Текст топика не должен быть особенно длинным. Желательно, чтобы он помещался на одной странице справки (или незначительно превышал ее). Особенно это касается рабочих топиков. Для методических топиков размеры могут быть несколько больше.

1.2. Терминологические топики небольшого размера, в тексте которых нет гиперссылок, удобно вызывать как всплывающее окошко (Popup).

1.3. Использование гиперссылок должно обеспечивать логичные переходы между разделами справок. Если в тексте топика упоминается какой - либо из экранов, элементов системы, либо по данному вопросу имеется отдельный рабочий топик, на него желательно сделать гиперссылку.

1.4. Содержание справки обычно формируется средствами генератора Help'ов. В структуре содержания желательно выделить теоретическую часть (основные понятия и используемые методы), глоссарий и рабочие топики.

1.5. При создании справки должен, по возможности, выдерживаться единый стиль. Цвета и размеры фона и шрифтов, оформление заголовков и гиперссылок, должны быть приятными и удобными для чтения. При выборе стиля желательно придерживаться фактического стандарта Windows. Различия в оформлении должны помогать ориентироваться в справочном файле. Например, рабочие, и методические топики могут отличаться каким-либо элементом стиля (цветом фона, формой заголовка или видом шрифта).

1.6. До каждого топика мы должны как-то добраться: он может быть вызван непосредственно из программы, указан в оглавлении или на него должна быть гиперссылка из других вызываемых топиков.

2. **Требования к рабочим топикам.** Эти топики должны вызываться контекстно из каждого экрана программы. Основные требования к ним:

2.1. **Унификация терминологии.** Все термины, обозначения в разных топиках должны быть одинаковы. Они также должны соответствовать тому, что присутствует на экране рабочей системы – названию экранов, названию элементов экрана, органов, элементов управления, и т.д. (Напомню, что сами названия экранов и элементов, расположенных на экране должны быть унифицированы, и носить содержательный характер, выражать именно тот смысл, который они несут).

2.2. **Полнота описания.** В топике должны быть описаны все элементы, присутствующие на экране и все состояния, в котором может находиться тот или иной элемент. Для каждого элемента надо сказать, что он означает, какую роль выполняет, что значит то или другое его состояние. В тех случаях, когда используются какие либо унифицированные элементы, (например, кнопки, которые могут присутствовать на разных экранах и имеющих одинаковое значение), для описания такого унифицированного элемента создается отдельный топик, и если такой элемент присутствует на экране, в топике для этого экрана дается гиперссылка на описание такого унифицированного элемента.

2.3. **Описание порядка работы.** В рабочем топике должны присутствовать ответы на следующие вопросы:

2.3.1. Для каких целей предназначен этот экран, какие задачи решаются с его помощью. Они могут быть обозначены очень кратко, более подробное описание будет содержаться в методическом топике, вызываемом по гиперссылке.

2.3.2. Какие действия и в какой последовательности необходимо или возможно производить с помощью этого рабочего экрана. Каким образом активизируются элементы управления – мышью, клавишами, и т.д. Те или иные условия таких действий могут быть только обозначены, более подробное описание будет содержаться в методическом топике, вызываемом по гиперссылке.

2.3.3. Если необходимо вводить какие-то данные на экране, следует указать, откуда следует взять эти данные, и способ их введения. При необходимости указывается формат данных, или другие их характеристики.

2.3.4. Что в результате происходит при работе с этим экраном. На какие экраны происходит переход с этого экрана, в каких случаях. Для интерпретации получаемых результатов делаются гиперссылки на соответствующие методические топики.

3. **Методические топики.** Должны включать:

3.1. Описание задач, которые решатся с помощью данной рабочей системы, последовательность, порядок обращения к различным экранам. Для иллюстрации методики можно привести описания различных приемов работы, условия, при которых целесообразно использовать те или другие приемы работы с программой. Если в рабочих топиках основное внимание уделяется техническим приемам работы с конкретным экраном, то в методических – решению общих задач, для которых и предназначена программа, а также последовательности взаимодействия человека и программы.

3.2. Описание видов результатов, получаемых при работе с системой, правила их использования, их интерпретации.

4. **Терминологические топики.** Кроме определений и пояснений в глоссарий желательно включить схемы и рисунки, иллюстрирующие взаимосвязи основных понятий. Сами определения желательно снабдить гиперссылками вида: «См. также…», отсылающими пользователя к определениям понятий, связанных с данным. Ссылками на термины, помещенные в глоссарий желательно снабдить все случаи использования данного термина в справочной системе.

Формирование конкретных требований к программным модулям

«Обратная волна» требований

Нам удалось представить решение всех задач нашей программы в виде последовательности работы программных модулей. Как формулировать требования к этим модулям?

Очевидно, что требования к результатам работы последнего модуля в цепочке такие же, как и требования к решению задачи. Что нужно сделать, чтобы модуль смог выполнить поставленную перед ним задачу? Часть необходимой информации модуль получит из входных данных, остальное должны передать ему другие модули, стоящие раньше в цепочке решения задачи. Так формулируются требования к предыдущим модулям. Рассматривая их как последние, сформулируем требования к модулям, стоящим в цепочке раньше. И так до начала цепочки. Это похоже на волну, прокатывающуюся от конца цепочки до начала, поэтому такой метод получил название ***«Обратная волна требований».***

Один и тот же модуль может участвовать в решении нескольких задач. Поэтому требования к нему должны быть сформулированы в результате анализа всех цепочек, в которых он участвует.

Советы молодому программисту.

Может показаться, что описанный путь постановки задачи слишком долгий и скучный. Гораздо проще и интереснее сразу приступить к программированию, хотя бы того, что понятно уже сейчас. Потом можно дописывать программу по мере осмысления задачи и требований пользователя.

Однако, поступив так, Вы достаточно скоро поймете, что разработка Вашей программы зашла в тупик. Добавление нового модуля требует существенной реорганизации данных и изменения логики работы всей программы. Поэтому каждый новый модуль добавляется все с большими трудностями. В конце концов, трудности становятся такими непреодолимыми, что проще написать всю программу заново.

Еще такой путь программирования напоминает строительство жилья богатеющим купцом. В центре, фасадом на улицу, стоит хибара. (Когда-то у купца денег хватило только на нее). Поодаль, в виде пристройки к хибаре, стоит добротный кирпичный дом (разбогател купец). А на отшибе, там, где у других стоит туалет, возведен роскошный особняк.

Хотелось бы Вам жить в таком доме?

Единая система программной документации

Для обеспечения сопоставимости и единства интерпретации программной документации в Советском Союзе была разработана Единая Система Программной Документации (ЕСПД). Ниже приводятся выдержки из головного стандарта системы: ГОСТ 19.001-77 «Общие положения», определяющего назначение, состав и область применения ЕСПД.

Назначение и цели ЕСПД

Единая система программной документации - комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации. В стандартах ЕСПД устанавливают требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ, что обеспечивает возможность:

- унификации программных изделий для взаимного обмена программами и применения ранее разработанных программ в новых разработках;
- снижения трудоемкости и повышения эффективности разработки, сопровождения, изготовления и эксплуатации программных изделий;
- автоматизации изготовления и хранения программной документации.

В понятие «сопровождение программы» включается:

- анализ функционирования программы,
- развитие и совершенствование программы,
- внесение изменений в нее с целью устранения ошибок.

В состав ЕСПД входят:

- основополагающие и организационно-методические стандарты;
- стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных;
- стандарты, обеспечивающие автоматизацию разработки программных документов.

Классификация и обозначение стандартов ЕСПД

Стандарты ЕСПД подразделяют на группы, приведенные в таблице.

Таблица ?

Группы стандартов ЕСПД

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7	Резервные группы
8	
9	Прочие стандарты

Обозначения стандартов ЕСПД строят по классификационному признаку.

В обозначение стандарта ЕСПД должны входить:

- цифры 19, присвоенные классу стандартов ЕСПД;
- одна цифра (после точки), обозначающая код классификационной группы стандартов, указанной в п. 3.1;
- двузначное число, определяющее порядковый номер стандарта в группе;
- двузначное число (после тире), указывающее год регистрации стандарта.

Пример обозначения стандарта “ Единая система программной документации. Общие положения ”:

```
ГОСТ 19.001-77
|    |  ||  |
|    |  ||  |  Год регистрации стандарта
|    |  ||     Порядковый номер стандарта в группе
|    |  |       Классификационная группа стандартов
|    |          Класс (стандарты ЕСПД)
|               Категория стандарта (государственный стандарт)
```

Изменение целей и назначения системы стандартов ЕСПД при переходе к рыночной экономике

При переходе к рыночным методам управления экономикой страны отпала необходимость жесткой регламентации формы и содержания программной документации для обеспечения ее сопоставимости. В социалистической экономике дублирование разработок (например, разработка одной и той же программы несколькими авторами) жестко пресекалась, так как вело к необоснованному перерасходу государственных средств. В рыночной экономике, такое дублирование даже приветствуется. Фирмы ведут разработки на свои средства, а потом выставляют конкурирующую продукцию на рынок. Покупатель сам определяет, какую продукцию ему купить. Конкуренция заставляет разработчиков выпускать все более совершенную и качественную продукцию.

В этих условиях изменяется роль стандартов ЕСПД. Их требования остаются обязательными только при определенных условиях[7] или в случае, если соблюдение

[7] Например, при регистрации программы как объекта интеллектуальной собственности оформление программной документации должно соответствовать требованиям стандартов 19.104-78 и 19.106-78.

требований стандарта упомянуто в договоре. В остальных случаях требования стандарта носят рекомендательный характер.

Стандарты, составляющие ЕСПД

В таблице ? приведен перечень стандартов, составляющих ЕСПД. Мы будем рассматривать только некоторые из них, определяющие требования к содержанию и оформлению программных документов.

Таблица ?

Перечень стандартов, входящих в
Единую Систему Программной Документации

ГОСТ	Название
19.001-77	Общие положения
19.002-80	Схемы алгоритмов и программ. Правила выполнения
19.003-80	Схемы алгоритмов и программ. Обозначение условные графические
19.004-80	Термины и определения
19.101-77	Виды программ и программных документов
19.102-77	Стадии разработки
19.103-78	Обозначение программ и программных документов
19.104-78	Основные надписи
19.105-78	Общие требования к программным документам
19.106-78	Требования к программным документам, выполненным печатным способом
19.201-78	Техническое задание. Требования к содержанию и оформлению
19.202-78	Спецификация. Требования к содержанию и оформлению
19.301-79	Программа и методика испытаний. Требования к содержанию и оформлению
19.401-78	Текст программы. Требования к содержанию и оформлению
19.402-78	Описание программы
19.403-79	Ведомость держателей подлинников
19.404-79	Пояснительная записка. Требования к содержанию и оформлению
19.501-78	Формуляр Требования к содержанию и оформлению
19.502-78	Описание применения требования к содержанию и оформлению
19.503-79	Руководство системного программиста. Требования к содержанию и оформлению
19.504-79	Руководство программиста. Требования к содержанию и оформлению
19.505-79	Руководство оператора. Требования к содержанию и оформлению
19.506-79	Описание языка. Требования к содержанию и оформлению
19.507-79	Ведомость эксплуатационных документов
19.508-79	Руководство по техническому обслуживанию. Требования к содержанию и оформлению

19.601-78	Общие правила дублирования, учета и хранения
19.602-78	Правила дублирования, учета и хранения программных документов, выполненных печатным способом
19.603-78	Общие правила внесения изменений
19.604-78	Правила внесения изменений в программные документы, выполненные печатным способом

Виды программной документации

В таблице ? перечислены основные виды программных документов, создаваемых и используемых в настоящее время при разработке, регистрации, сертификации и использовании компьютерных программ.

Таблица ?

Основные виды программных документов

Вид документа	Комментарий
Спецификация.	Перечень документов
Формуляр	Титульный лист, в котором указываются основные реквизиты программы
Техническое задание.	Система требований к программному комплексу
Пояснительная записка	Пояснение к техническому заданию
Программа и методика испытаний.	Методика, по которой проверяется соответствие программы предъявленным к ней требованиям
Текст программы.	Исходный код
Сопроводительная документация	Передается Заказчику или покупателю вместе с программой. Перечисленные ниже документы входят в сопроводительную документацию. Допускается объединять их в один документ.
Руководство пользователя	
Описание программы	
Описание применения	
Руководство системного программиста	
Руководство программиста.	
Руководство оператора.	

Разработка программной документации

На рис ? представлена схема разработки и использования программной документации. Техническое задание и пояснительная записка к нему разрабатываются на этапе постановки задачи. Фактически они являются результирующими (выходными) документами данного этапа. В них фиксируются требования к разрабатываемой программе. Если заказчик и разработчик программы работают в разных организациях, техническое задание становится обязательной частью договора. В дальнейшем, при тестировании программы и принятии решения о ее работоспособности, именно техническое задание будет определять требования, подлежащие контролю.

На этапе разработки формируется текст программы (исходный код) и описание программы. На заключительном этапе разработки формируется система контекстной подсказки (Help) и Руководство пользователя. Текст программы используется при регистрации программы как объекта интеллектуальной собственности. При этом он

полностью или частично депонируется[8]. В случае возникновения споров об авторстве программы агентство, в котором зарегистрирована программа, предоставляет в арбитраж[9] копию исходного кода для установления авторства. Именно поэтому свои программы стоит подписывать!

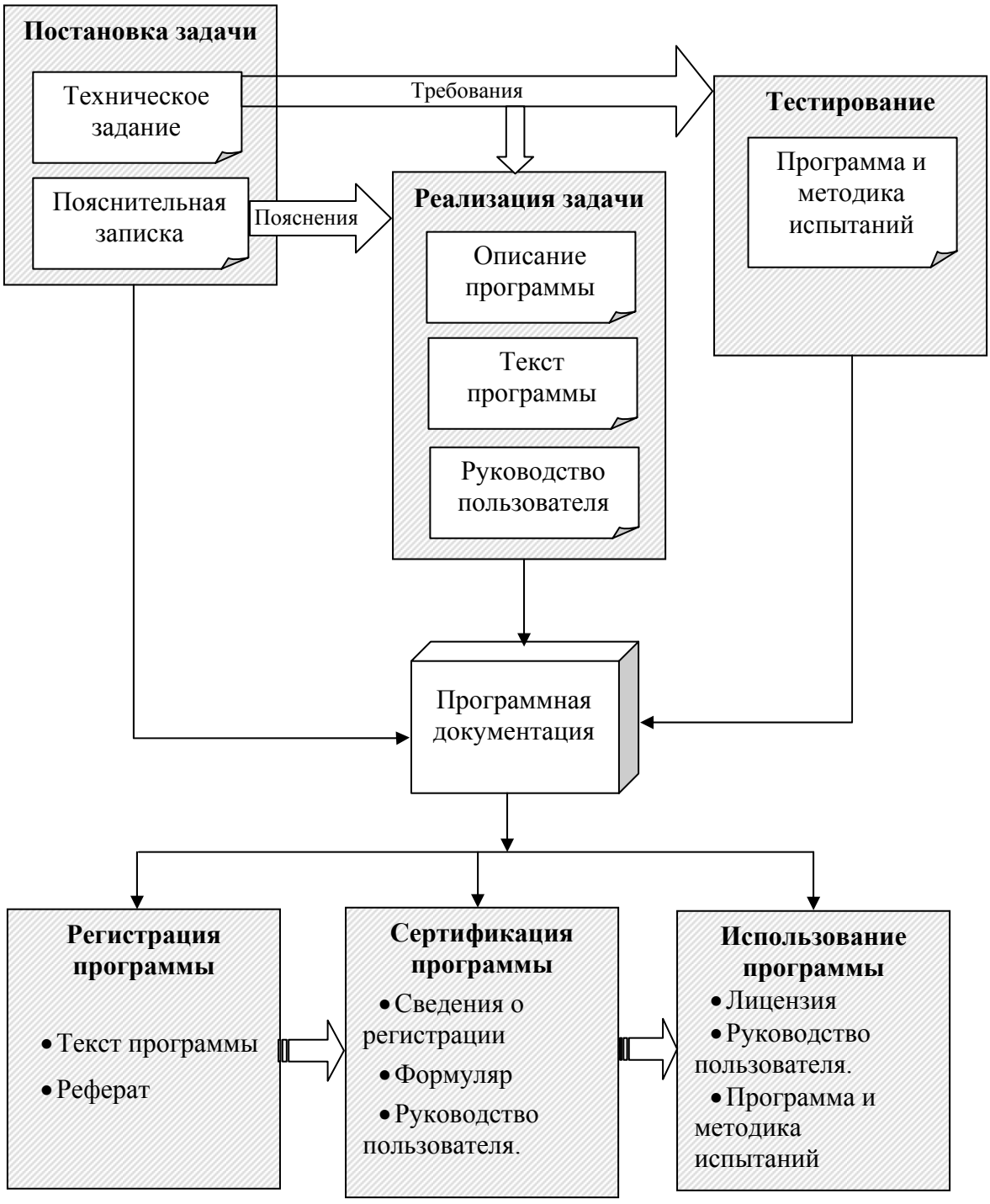


Рис ?. Схема создания и использования программной документации

Для проведения тестирования разрабатывается специальный документ: «Программа и методика испытаний». В документе перечисляются требования к программе, которые необходимо проверить. Перечень требований определяется

[8] Депонирование – обеспечение сохранности какого либо объекта или документа.
[9] Арбитраж – специальный суд, разбирающий вопросы промышленного права

Техническим заданием, а также стандартами или особыми требованиями, записанными в договоре. Для каждого требования описывается методика, по которой это требование должно быть проверено. Документ «Программа и методика испытаний» может быть использован и при сертификации программы.

При успешной регистрации программы как объекта интеллектуальной собственности, авторам выдается свидетельство о регистрации. Свидетельство дает право:

- продавать программу целиком – без права использовать ее самому и продавать еще раз. Такое право оформляется ***исключительной лицензией;***
- продавать право использование программы одному или многим пользователям с правом пользоваться самому. Это ***неисключительная лицензия,***
- дарить или безвозмездно уступать программу кому угодно.

Таким образом, регистрация программы закрепляет за автором права собственности и позволяет ему извлечь из использования программы коммерческую выгоду.

Пример разработки комплекта документов. Решение квадратного уравнения

Особенности разработки комплекта программной документации рассмотрим на примере программы, предназначенной для знакомства школьников с методами решения квадратного уравнения[10].

Результаты постановки задачи зафиксируем в документах: «Пояснительная записка» и «Техническое задание». В Пояснительной записке опишем идею программы, а в Техническом задании сформулируем основные требования к программе.

Пояснительная записка

Введение.

Наименование программы: “Решение квадратного уравнения“. Программа предназначена для знакомства школьников с методами использования компьютера на примере нахождения корней квадратного уравнения.

Назначение и область применения.

Программа является обучающей. Её цель:

1) приобщить школьников к использованию компьютера для решения математических задач;
2) познакомить с теорией квадратного уравнения;
3) научить решать квадратные уравнения.

Технические характеристики.

Программа должна выдавать результат с сообщением понятным пользователю. Для решения квадратного уравнения вычисляется его дискриминант (**D**), после чего находятся корни уравнения ($\mathbf{x}_1$,$\mathbf{x}_2$). Если введённые коэффициенты описывают не квадратное, а линейное уравнение (**A**=0), то находится корень линейного уравнения, о чём сообщается пользователю. В случае если введённые коэффициенты не соответствуют ни квадратному, ни линейному уравнениям (**A**=0 и **B**=0), то должно выдаваться сообщение.

При запуске программы пользователю предлагается ввести коэффициенты **A,B,C**, после чего выдаётся результат в виде сообщения.

Программа написана в среде C++ Builder 6.0 Enterprise. Программа разработана для IBM PC со следующими характеристиками:

1) процессор Intel Pentium 266 MHz или выше;

[10] Комплект документации по этой программе подготовили студентки группы ПВС 23 Баркова Наталья, Пугачева Светлана и Улдарова Кадрия

2) объём ОЗУ не менее 64 MB;
3) наличие установленной операционной системы Windows 98 / NT / 2000 / XP
4) наличие стандартной клавиатуры;
5) наличие манипулятора типа ”мышь”

Источники, использованные при разработке.

При написании программы использовалась литература: Учебник математики для седьмого класса средней школы, Методика преподавания математики в средних классах общеобразовательной школы, Руководство по C++ Builder 6.0 Enterprise.

—

Введение.

Наименование программы: “Решение квадратного уравнения“. Программа предназначена для знакомства школьников с методами использования компьютера на примере нахождения корней квадратного уравнения.

Назначение и область применения.

Программа является обучающей. Её цель:

4) приобщить школьников к использованию компьютера для решения математических задач;
5) познакомить с теорией квадратного уравнения;
6) научить решать квадратные уравнения.

Технические характеристики.

Программа должна выдавать результат с сообщением понятным пользователю.
Для решения квадратного уравнения вычисляется его дискриминант (**D**), после чего находятся корни уравнения ($\mathbf{x_1}$,$\mathbf{x_2}$). Если введённые коэффициенты описывают не квадратное, а линейное уравнение (**A**=0), то находится корень линейного уравнения, о чём сообщается пользователю. В случае если введённые коэффициенты не соответствуют ни квадратному, ни линейному уравнениям (**A**=0 и **B**=0), то должно выдаваться сообщение.

При запуске программы пользователю предлагается ввести коэффициенты A**,B,C**, после чего выдаётся результат в виде сообщения.

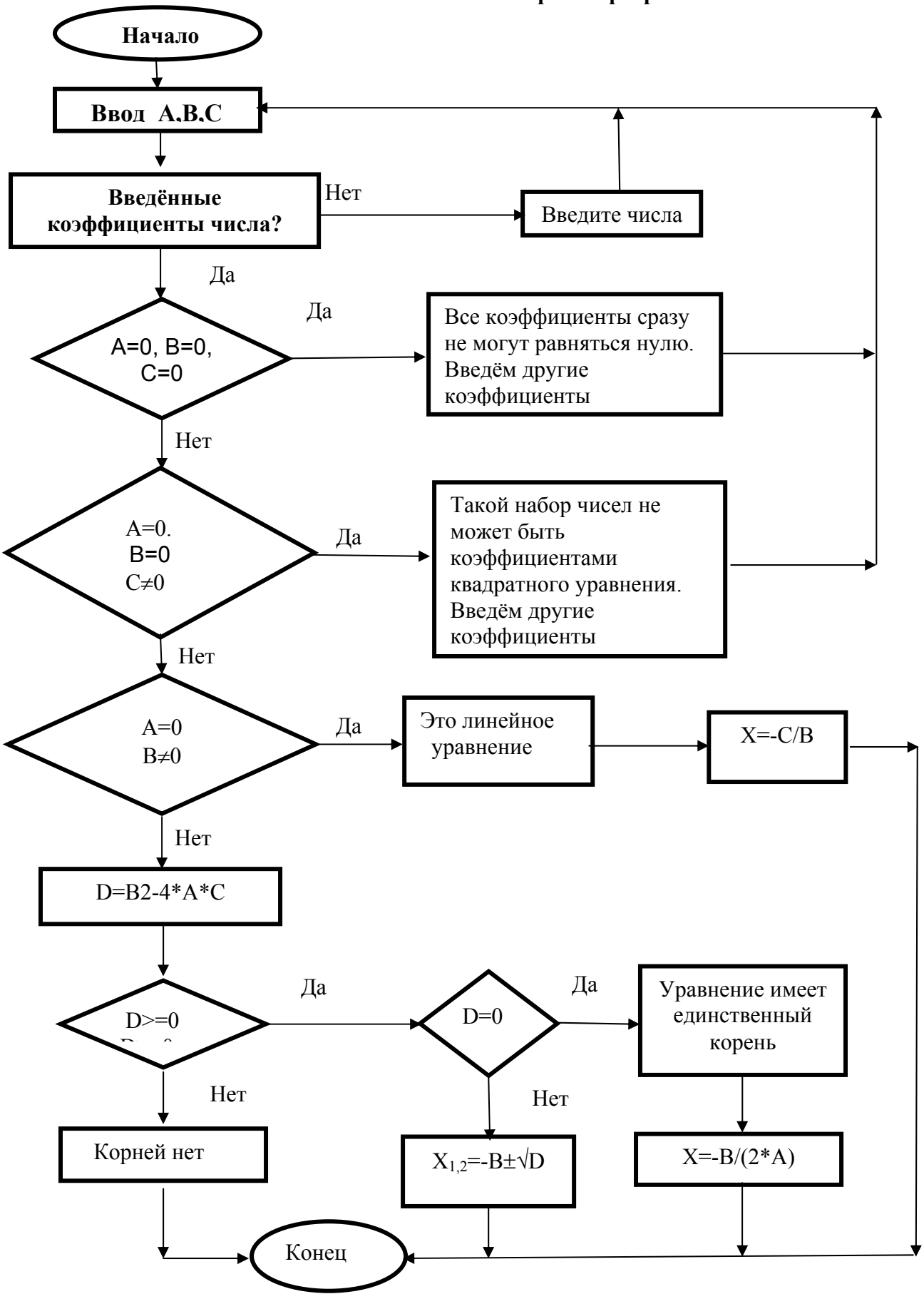
Программа написана в среде C++ Builder 6.0 Enterprise. Программа разработана для IBM PC со следующими характеристиками:

6) процессор Intel Pentium 266 MHz или выше;
7) объём ОЗУ не менее 64 MB;
8) наличие установленной операционной системы Windows 98 / NT / 2000 / XP
9) наличие стандартной клавиатуры;
10) наличие манипулятора типа ”мышь”

Источники, использованные при разработке.

При написании программы использовалась литература: Учебник математики для седьмого класса средней школы, Методика преподавания математики в средних классах общеобразовательной школы, Руководство по C++ Builder 6.0 Enterprise.

Алгоритм программы



ОПИСАНИЕ ПРОГРАММЫ

Общие сведения

Наименование программы: «Решение квадратного уравнения ». Программа предназначена для школьников. Программа может использоваться для обучения школьников приемам работы на компьютере на примере нахождения корней квадратного уравнения. Для написания данной программы использовался язык программирования C++, так как он наиболее удобен и известен авторам.

Функциональное назначение.

Программа является обучающей. Ее цель:

1) приобщить школьников к использованию компьютера для решения математических задач;
2) познакомить с теорией квадратного уравнения;
3) научить решать квадратные уравнения.

Программа написана в среде “C++ Builder 6.0 Enterprise”.

Программа работает под управлением операционных систем Windows 98/NT/2000/XP.

Используемые технические средства.

Программа разработана для IBM PC со следующими характеристиками:

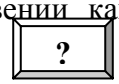
- Процессор Intel Pentium 266 MHz, или выше.
- Объём ОЗУ не менее 64Mb.
- Наличие установленной операционной системы Windows 98/NT/2000/XP.
- Наличие стандартной клавиатуры.
- Наличие манипулятора типа “мышь”.

Вызов и загрузка.

При установке программы на рабочем столе появляется ярлычок, кликнув на который запускается наша программа. Программа имеет одну входную точку, которой соответствует ввод коэффициентов с клавиатуры.

Входные данные.

При запуске программы появляется персонаж, который предлагает познакомиться: сообщает своё имя, спрашивает имя пользователя и его возраст. Если возраст школьника меньше 12 лет, то выводиться сообщение: «А не рано ли решать квадратное уравнение?», иначе: «Давай попробуем!» После этого предлагается ввести коэффициенты уравнения в «ячейки ввода» как показано в ниже приведённом примере.

При возникновении каких-либо вопросов пользователь может воспользоваться кнопкой помощи (

), которая содержит следующую полезную информацию:

квадратное уравнение – уравнение вида: $A*x^2 + B*x + C = 0$,
где x - переменная, A, B, C – некоторые числа ($A \neq 0$). A, B и C называются коэффициентами квадратного уравнения. A – первый коэффициент, B – второй коэффициент, C – свободный член.

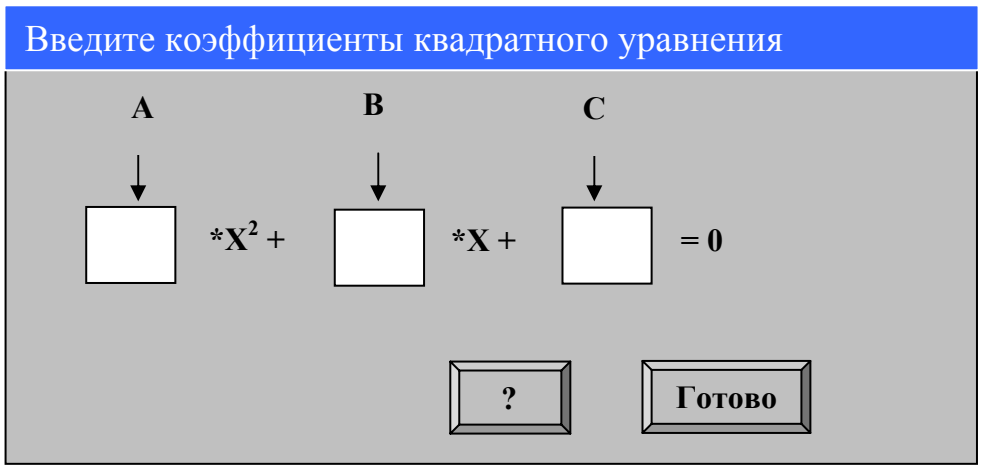
Если в квадратном уравнении хотя бы один коэффициент равен нулю, то такое уравнение называется неполным квадратным уравнением, которое бывает трёх видов:

1) $A*x^2 + C = 0$ ($C < 0$);
   $A*x^2 = - C$;
   $x^2 = - C/A$;
   Если $x^2 = - C/A > 0$, то уравнение имеет два решения:
   $x_1=\sqrt{(-C/A)}$, $x_2=-\sqrt{(-C/A)}$,
   а иначе ни одного.
2) $A* x^2 + B*x = 0$ ($A \neq 0$ и $B \neq 0$);
   $x (A*x + B) = 0$;
   $x = 0$ или $A*x + B = 0$;
   $x_1=0$
   $x_2 = - B/A$;
3) $A* x^2 = 0$;
   $x = 0$;

Если $A = 1$, то полученное квадратное уравнение называется приведённым.
Данное уравнение вида:
$– x^2 + p*x + q = 0$
можно решить двумя способами: по теореме Виета и через дискриминант.
По теореме Виета: $x_1*x_2 = q$
$x_1 + x_2 = - p$.
Если дискриминант больше нуля ($D > 0$), то уравнение имеет два корня, если $D = 0$, то один корень, а если $D < 0$, то уравнение не имеет решения.
Когда B является чётным числом, квадратное уравнение принимает следующий вид:
$A* x^2 + 2*k*x + C = 0$;
$D1= k^2 - A*C$
$x_{1,2}= (-k \pm \sqrt{(D1)}/A)$
После ввода коэффициентов нужно нажать на кнопку «готово» (Готово)

Выходные данные.

При вводе коэффициентов, при которых квадратное уравнение не имеет смысла, выводиться сообщение с разъяснением неправильного выбора коэффициентов и предложением ввести новые коэффициенты. В другом случае выводиться результат с сообщением: «Ты молодец! Вот твой результат»
Затем продолжается диалог со школьником и задаётся вопрос: «Тебе понравилось? Может, решим ещё одно уравнение?» Если пользователь нажимает кнопку «Да», то возвращается к вводу коэффициентов; если «Нет», то выводиться сообщение: «Ну ладно. Тогда в другой раз. Пока!»



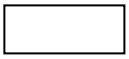
 - «ячейка ввода» для коэффициентов.

Рис ? Окно ввода коэффициентов

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ

Назначение программы.

Программа является обучающей. Ее цель:

1) приобщить школьников к использованию компьютера для решения математических задач;
4) познакомить с теорией квадратного уравнения;
5) научить решать квадратные уравнения;

Условия выполнения программы.

Программа разработана для IBM PC со следующими характеристиками:

- Процессор Intel Pentium 266 MHz, или выше.
- Объём ОЗУ не менее 64Mb.
- Наличие установленной операционной системы Windows 98/NT/2000/XP.
- Наличие стандартной клавиатуры.
- Наличие манипулятора типа “мышь”.

Выполнение программы.

Для загрузки, запуска, выполнения и завершения программы необходимо осуществить следующие действия:

1) после открытия гибкого диска находимо навести курсор на ярлычок SETUP и дважды нажать левую кнопку «мышки»;
2) происходит запуск программы, и на рабочем столе появляется ярлычок для запуска программы;
3) после запуска программы на экране появляется персонаж. Между школьником и персонажем ведется диалог, в ходе которого предлагается ввести коэффициенты уравнения;
4) нажав кнопку “Готово” выводится либо сообщение о неправильно введенных коэффициентах , либо результат;
5)затем продолжается диалог: пользователю предлагается решить ещё одно уравнение. В случае согласия программа возвращается к вводу коэффициентов ,иначе происходит выход из программы .

Сообщения оператору.

В ходе выполнения программы появляются сообщения:

“Введите числа!”

Это сообщение выдаётся в том случае, если пользователь ввел бессмысленные коэффициенты и ему нужно ввести именно числа.

Например : «, . / # % ^) (! ~ \ $ * № и т.д.»

“Введём другие коэффициенты”

Это сообщение выдаётся в том случае, если пользователь ввел коэффициенты, при которых уравнение не имеет смысла. Это возможно в том случае, если все коэффициенты нулевые.

"<C>не может равняться нулю. Введём другие коэффициенты"
Это сообщение выдаётся в том случае, если коэффициенты "A" и "B" были заданы как нулевые, а значение " C" отлично от нуля. Т.е. константа "C" не равная нулю, по полученному уравнению должна равняться нулю.
"Это линейное уравнение"
Это сообщение выдаётся в том случае, если при введённых коэффициентах получается не квадратное, а линейное уравнение. Поэтому решением уравнения является единственный корень.
Пример: если A = 0, то получаем линейное уравнение B*X+C=0

"Единственный корень"
Это сообщение выдаётся в том случае, если дискриминант равен нулю, и квадратное уравнение имеет два одинаковых корня.
Например: если D = 0, то $X_{1,2} = -B/2*A$

"Корней нет"
Это сообщение выдаётся в том случае, если дискриминант меньше нуля, и уравнение не имеет действительных корней.

Сертификация программных продуктов

Цели и задачи сертификации

Что дает сертификат

Тестирование программ

Пример: расчет корней квадратного уравнения

Советы молодому программисту
Правила написания надежных программ

3. Всегда проверяйте возможность выполнения планируемых действий
   a. Если в Вашей программе есть деление, проверьте, не равняется ли нулю знаменатель
   b. Если требуется извлечь квадратный корень, проверьте: не отрицательное ли у Вас подкоренное выражение
   c. Если Вы хотите распечатать Ваши результаты, проверьте, доступен ли принтер
   d. …
4. Всегда проверяйте результат выполнения операций
   a. Если Вам нужно извлечь данные из БД, проверьте, а действительно ли нужные Вам данные имеются в базе
   b. Проверяйте, справились ли со своей задачей команды: «открыть файл», «вызвать программный модуль» и т.д.
   c. В большинстве современных языков выполнение операций сопровождается формированием кода ошибки. Проверяйте значение этого кода и программируйте реакцию компьютера в зависимости от этого значения.
5. Обеспечение защиты от ошибок оператора

a. Просите подтверждения выполнения необратимых действий. Перед тем как выполнить необратимое действие (например, стереть файл) Ваша программа должна запросить его подтверждение. При этом по умолчанию должно считаться, что необратимое действие следует отменить! Пользователь не должен машинально (не думая) подтвердить выполнение необратимого действия.

b. Предусмотрите «откат» к предыдущему состоянию. Выполнив действие, пользователь может обнаружить, что оно ошибочно. Поэтому не спешите что ни будь удалять или изменять безвозвратно. Создайте временную версию Вашего объекта и сделайте изменения на ней. А предыдущую версию пока сохраните. Если пользователь поймет, что выполнил неверное действие, Ваша программа всегда сможет обратится к предыдущей версии.

c. Помните! Только глубоко эшелонированная оборона от случайных ошибок может сделать Вашу программу надежной.

Типовые требования к программам

Советы молодому программисту

Как написать программу, понятную пользователю

1. Если Вы хотите использовать программу в России, ее интерфейс должен быть написан по-русски.
2. Используйте терминологию, понятную пользователю. Избегайте «программисткой фени». Пользователь не обязан знать, как устроена Ваша программа изнутри. Для него, скорее всего, формы ввода и вывода представляются просто документами, высвечиваемыми на экране. Он привык работать с такими документами: заполнять, регистрировать, утверждать и т.д. Если пользователь видит на экране кнопки с названиями знакомых ему операций, он быстрее освоит Вашу программу, она ему понравится.
3. В названии экрана указывайте задачу, которую нужно решать с его помощью.
4. На каждом экране Вашей программы должна быть представлена вся информация, необходимая при выполнении операции для которой предназначен этот экран.

Например. Вы ведете личные карточки студентов. Выбрав из списка нужную Вам фамилию, Вы приступаете к заполнению формы успеваемости студента. Вас отвлекли, после вынужденного перерыва, Вы продолжаете работу. Если на экране не указана фамилия студента и номер группы, какую информацию вводить?

5. Разработайте файл контекстной помощи для Вашей программы. Современные средства программирования позволяют организовать вызов нужного места (топика) Help-файла с каждого экрана. (Обычно это достигается указанием параметра Help_ID). Расскажите в подсказке, для какой задаче нужен данный экран, что нужно сделать для ее решения. Опишите назначения органов управления.
6. Организуя помощь, помните, что пользователь может не знать смысла применяемых Вами терминов. В этом случае он не поймет Вашу подсказку и может натворить бед. Чтобы этого не случилось, включите в ваш Help-файл глоссарий (список терминов), в котором разъясняется смысл каждого термина. Во всех текстах Help-файла, где встречается данный термин, организуйте гиперссылки на его определение в глоссарии. Это поможет пользователю быстрее и лучше освоить Вашу программу, избавит его и Вас от многих проблем.

Методики тестирования программ

Тестирующие задачи
Тестирующие данные
Полигоны

Тестирование данных

Типовые требования к данным

Методики тестирования данных

Оценка полноты охвата источников информации
Оценка целостности данных
Оценка достоверности данных
Оценка актуальности данных

Сертификация баз данных

Классификация баз данных

Выбор цели сертификации

Последовательность проведения сертификации БД

Использование сертификата

Литература

6. Закон РФ от 27 апреля 1993 г. N 4871-I "Об обеспечении единства измерений"
7. Кумэ Х. "Статистические методы повышения качества". Пер с англ., М., "Финансы и статистика", 1990, 301 стр