Эксплуатационные требования к программным продуктам.

Разработка технического задания

Эксплуатационные требования определяют некоторые характеристики разрабатываемого ПО, проявляемые в процессе его функционирования.

Характеристики разрабатываемого ПО:

- правильность – функционирование в соответствии с ТЗ;
- универсальность – обеспечение правильной работы при любых допустимых данных и защиты от неправильных данных;
- надежность (помехозащищенность) – обеспечение полной повторяемости результатов, т.е. обеспечение их правильности при наличии различного рода сбоев;
- проверяемость – возможность проверки получаемых результатов;
- точность результатов – обеспечение заданной погрешности результатов;
- защищенность – обеспечение конфиденциальности информации;

- программная совместимость – возможность совместного функционирования с другим ПО;
- аппаратная совместимость – возможность совместного функционирования с некоторым оборудованием;
- эффективность – использование минимально возможного количества ресурсов технических средств;
- адаптируемость – возможность быстрой модификации с целью приспособления к изменяющимся условиям функционирования;
- повторная входимость – возможность повторного выполнения без перезагрузки;
- реентерабельность – возможность «параллельного» использования несколькими процессами.

Правильность - обязательное требование для любого ПО: все, что указано в ТЗ, должно быть реализовано. Следует понимать, что ни тестирование, ни верификация не доказывают правильности созданного программного продукта.

Поэтому говорят об определенной *вероятности наличия ошибок.*

Чем большая ответственность перекладывается на компьютерную систему, тем меньше должна быть вероятность программного и аппаратного сбоя.

Например, вероятность неправильной работы для системы управления атомной электростанцией должна быть близка к нулю.

Универсальность - также входит в группу обязательных требований. Ничего хорошего нет, если разработанная система выдает результат для некорректных данных или аварийно завершает свою работу на некоторых наборах данных. Но доказать универсальность программы, как и ее правильность, невозможно, поэтому говорят о степени универсальности программы.

Чем выше требования к правильности и универсальности ПО, тем выше и требования к его *надежности*.

Источниками помех могут являться все участники вычислительного процесса: технические и программные средства, люди.

Технические средства подвержены сбоям (из-за резких скачков напряжения питания или помех при передаче информации по сетям).

ПО может содержать ошибки. Люди могут ошибаться при вводе исходных данных.

Современные вычислительные устройства достаточно надежны. Сбои технических средств регистрируются аппаратно, результаты вычислений восстанавливаются. При длительных вычислениях промежуточные результаты сохраняют (создание контрольных точек), что позволяет при сбое продолжить вычисления с данными, записанными в последней контрольной точке.

Передача информации по сетям аппаратно контролируется, применяется помехозащитное кодирование, которое позволяет находить и исправлять ошибки передачи данных.

Однако полностью исключить ошибки технических средств невозможно, поэтому в случаях, когда требования к надежности высоки, используют дублирование систем, при котором две системы решают одну и ту же задачу параллельно, периодически сверяя полученные результаты.

Часто самым «ненадежным элементом» современных систем являются люди. В условиях монотонной работы операторы допускают большое количество ошибок.

Средства, позволяющие снизить количество ошибок: там, где это возможно, используют ввод избыточной информации, позволяющий выполнять проверки правильности вводимых данных (ввод контрольных сумм и т.п.). Используют подсказки, когда информацию необходимо не вводить, а выбирать из некоторого списка и т.п.

Повышенные требования к надежности предъявляют при разработке систем управления, функционирующих в режиме реального времени, когда вычисления выполняются параллельно с технологическими процессами.

Это требование важно и для научно-технических систем и БД.

Для обеспечения *проверяемости* следует документально фиксировать исходные данные, установленные режимы и прочую информацию, которая влияет на получаемые результаты. Это существенно в случаях, когда данные поступают непосредственно от датчиков.

Точность или величина погрешности результатов зависит от точности исходных данных, степени адекватности используемой модели, точности выбранного метода и погрешности выполнения операций в компьютере.

Требования к точности результатов наиболее жесткие для систем управления технологическими процессами (например, химическими) и систем навигации (например, система управления стыковкой космических аппаратов).

Обеспечение *защищенности* (конфиденциальности) информации - отдельная и в условиях наличия сетей достаточно сложная задача. Помимо чисто программных средств защиты (кодирование информации, идентификация пользователя) используют специальные организационные приемы. Наиболее жесткие требования предъявляются к системам, в которых хранится информация, связанная с государственной и коммерческой тайной.

Требование *программной совместимости* - от возможности совместной установки с указанным ПО до обеспечения взаимодействия с ним (обмена данными и т.п.). Часто приходится обеспечивать функционирование ПО под управлением заданной ОС. Может потребоваться предусмотреть получение данных из программы или передачу некоторых данных ей. Тогда необходимо точно оговорить форматы передаваемых данных.

Требование *аппаратной совместимости* формулируют в виде минимально возможной конфигурации оборудования, на котором будет работать ПО. При использовании нестандартного оборудования должны быть указаны интерфейсы или протоколы обмена информацией.

Эффективность системы оценивается отдельно по каждому ресурсу вычислительной установки. Используют критерии:

- время ответа системы – для систем, взаимодействующих с пользователем в интерактивном режиме, и систем реального времени;
- объем оперативной памяти – для продуктов, работающих в системах с ограниченным объемом ОП;
- объем внешней памяти – для продуктов, использующих внешнюю память (БД);
- количество обслуживаемых внешних устройств – для продуктов, осуществляющих интенсивное взаимодействие с внешними устройствами, например датчиками.

Требования эффективности могут противоречить друг другу: чтобы уменьшить время выполнения фрагмента программы, может потребоваться дополнительный объем ОП.

Адаптируемость - оценивает технологическое качество ПО, поэтому оценить эту характеристику количественно практически невозможно. Можно только констатировать, что при создании продукта использованы технологии и приемы, облегчающие его модернизацию.

Требование *повторной входимости* предъявляется к ПО, резидентно загруженному в ОП (драйверам). Для обеспечения требования необходимо так организовать программу, чтобы никакие её исходные данные не затирались в процессе выполнения или восстанавливались в начале или при завершении каждого вызова.

Требование *реентерабельности* - более жесткое, чем повторная входимость, в этом случае все данные, изменяемые программой в процессе выполнения, должны быть выделены в специальный блок, копия которого создается для каждого процесса при вызове программы.

Сложность программных систем не позволяет сразу сформулировать четкие требования к ним. Обычно для перехода от идеи создания некоторого ПО к четкой формулировке требований, которые могут быть занесены в ТЗ, необходимо выполнить предпроектные исследования в области разработки.

Предпроектные исследования предметной области

Цель предпроектных исследований - преобразование общих нечетких знаний о предназначении будущего ПО в сравнительно точные требования к нему.

Два варианта неопределенности:

- неизвестны методы решения формулируемой задачи – при решении научно-технических задач;
- неизвестна структура автоматизируемых информационных процессов – при построении АСУ предприятиями.

В первом случае во время предпроектных исследований определяют возможность решения поставленной задачи и методы, позволяющие получить результат, что может потребовать научных исследований, разработки и исследования новых моделей объектов реального мира.

Во втором случае определяют:

- структуру и взаимосвязи автоматизируемых информационных процессов;
- распределение функций между человеком и системой, между аппаратурой и ПО;
- функции ПО; внешние условия его функционирования, особенности интерфейсов;
- требования к программным и информационным компонентам, необходимые аппаратные ресурсы, требования к БД и физические характеристики программных компонент.

Результаты предпроектных исследований предметной области используют в процессе разработки ТЗ.

ТЗ - документ, в котором сформулированы основные цели разработки, требования к программному продукту, сроки и этапы разработки и регламентирован процесс приемно-сдаточных испытаний.

Факторы, определяющие характеристики разрабатываемого ПО:

- исходные данные и требуемые результаты, которые определяют *функции* программы или системы;
- среда функционирования (программная и аппаратная) – может быть задана, а может выбираться для обеспечения параметров, указанных в ТЗ;
- возможное взаимодействие с другим ПО и/или специальными техническими средствами.

Последовательность разработки ТЗ:

- Устанавливают набор выполняемых функций, перечень и характеристики исходных данных.
- Определяют перечень результатов, их характеристики и способы представления.
- Уточняют среду функционирования ПО: комплектацию и параметры технических средств, версию используемой ОС и

На ТЗ существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению».

ТЗ содержит разделы:

- *введение;*
- *основания для разработки;*
- *назначение разработки;*
- *требования к программе или программному изделию;*
- *требования к программной документации;*
- *технико-экономические показатели;*
- *стадии и этапы разработки;*
- *порядок контроля и приемки.*

При необходимости допускается в техническое задание включать приложения

Введение должно включать наименование и краткую характеристику области применения программы или программного продукта, а также объекта (например, системы) в котором предполагается их использовать.

Основное назначение введения – продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

Раздел ***Основания для разработки*** должен содержать наименование документа, на основании которого ведется разработка, организации, утвердившей данный документ, и наименование или условное обозначение темы разработки. Таким документом может служить план, приказ, договор и т.п.

Раздел ***Назначение разработки*** должен содержать описание функционального и эксплуатационного назначения программного продукта с указанием категорий пользователей.

Раздел ***Требования к программе*** включает подразделы:

- *требования к функциональным характеристикам;*
- *требования к надежности;*
- *условия эксплуатации;*
- *требования к составу и параметрам технических средств;*
- *требования к информационной и программной совместимости;*
- *требования к маркировке и упаковке;*
- *требования к транспортированию и хранению;*
- *специальные требования.*

Наиболее важный - подраздел ***Требования функциональным характеристикам***. В нем должны быть перечислены выполняемые функции и описаны состав, характеристики и формы представления исходных данных и результатов. В этом же разделе при необходимости указывают критерии эффективности: максимально допустимое время ответа системы, максимальный объем используемой оперативной и/или внешней памяти и др.

Если разработанное ПО не будет выполнять указанных в ТЗ функций, то оно считается не соответствующимТЗ, т. е. неправильным с точки зрения критериев качества.

В подразделе *Требования к надежности* указывают уровень надежности, который должен быть обеспечен разрабатываемой системой и время восстановления системы после сбоя. Для систем с обычными требованиями к надежности в этом разделе иногда регламентируют действия разрабатываемого продукта по увеличению надежности результатов (контроль входной и выходной информации, создание резервных копий промежуточных результатов и т.п.).

В подразделе ***Требования к информационной и программной совместимости*** можно задать методы решения, определить язык или среду программирования для разработки, а также используемую ОС и другие системные и пользовательские программные средства, с которым должно взаимодействовать разрабатываемое ПО. Также указывают, какую степень защиты информации необходимо предусмотреть.

В разделе ***Требования к программной документации*** указывают необходимость наличия руководства программиста, пользователя, системного программиста, пояснительной записки и т.п. На все эти типы документов существуют ГОСТы.

В разделе ***Технико-экономические показатели*** указывают ориентировочную экономическую эффективность и экономические преимущества по сравнению с аналогами.

В ***приложениях*** приводят: перечень научно-исследовательских работ, обосновывающих разработку; схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы

Пример 1. Разработать ТЗ на программный продукт, предназначенный для наглядной демонстрации школьникам графиков функций одного аргумента y = f(x). Разрабатываемая программа должна рассчитывать таблицу значений и строить график функций на заданном отрезке по заданной формуле и менять шаг аргумента и границы отрезка. Кроме этого, программа должна запоминать введенные формулы.

1.ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программы построения графиков и таблиц значений функций одной переменной, предназначенной для использования школьниками старших классов.

В школьном курсе элементарной алгебры тема анализа функций является одной из самых сложных. При изучении данной темы школьники должны научиться исследовать и строить графики функций одной переменной, используя все известные характеристические точки функции, включая корни, точки разрыва первого и второго рода и т.д.

Существующее программное обеспечение, которое может решать подобные задачи, является универсальным, например Eurica или MathCad. Оно имеет сравнительно сложный пользовательский интерфейс, ориентированный на пользователя, прослушавшего, как минимум, институтский курс высшей математики, что делает использование подобных средств школьниками невозможным.

Разрабатываемая программа позволит школьникам проверить

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Программа разрабатывается на основе учебного плана кафедры ****.

3. НАЗНАЧЕНИЕ

Основным назначением программы является помощь школьникам при изучении раздела «Исследование функций одного аргумента» школьного курса элементарной алгебры.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1.Требования к функциональным характеристикам

4.1.1. Программа должна обеспечивать возможность выполнения следующих функций:

- ввод аналитического представления функции одной переменной и длительное хранение его в системе;
- ввод и изменение интервала определения функции;
- ввод и корректировку шага аргумента;
- построение таблицы значений функции на заданном интервале или изображение графика функции на заданном интервале при

4.1.2. Исходные данные:

- аналитическое задание функции;
- интервал определения функции;
- шаг изменения аргумента, определяющий количество точек на интервале.

4.2. Требования к надежности

Предусмотреть контроль вводимой информации.

Предусмотреть блокировку некорректных действий пользователя при работе с системой.

4.3. Требования к составу и параметрам технических средств

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

- тип процессора… ……Pentium и выше;

объем оперативного запоминающего устройства……32 Мб и более.

4.4. Требования к информационной и программной совместимости

- Система должна работать под управлением семейства операционных систем Win32 (Windows 2000, Windows NT и т.п.).

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

Разрабатываемая программа должна включать справочную информацию об основных терминах соответствующего раздела математики и подсказки учащимся.

В состав сопровождающей документации должны входить:

- Пояснительная записка на 25-30 листах, содержащая описание разработки.
- Руководство пользователя.

Пример 2. Разработать ТЗ на создание системы решения комбинаторно-оптимизационных задач. Система должна включать алгоритмы решения задач: поиска цикла минимальной длины (задача коммивояжера), поиска кратчайшего пути и поиска минимального связывающего дерева.

Комбинаторными называют задачи, решение которых сводится к выбору варианта из конечного множества решений. В комбинаторно-оптимизационных задачах в конечном множестве допустимых решений отыскивается такое, для которого целевая функция достигает оптимального (минимального или максимального) значения.

Задача коммивояжера или поиска цикла минимальной длины в простейшем варианте формулируется следующим образом. Задан список городов и дорог, соединяющих данные города. Известны расстояния между городами. Необходимо объехать все города, не заезжая ни в какой город дважды, и вернуться в исходный город так, чтобы суммарная длина пути была минимальной.

- Задача поиска кратчайшего пути при тех же исходных данных предполагает другую цель: необходимо проехать из одного города в другой так, чтобы суммарная длина пути была минимальной.
- Задача поиска минимального связывающего дерева ставится на тех же исходных данных, но теперь мы прокладываем телефонные линии вдоль дороги и хотим, чтобы длина кабеля была минимальной.

Принципиальные решения начальных этапов проектирования

На начальных этапах процесса проектирования должны быть приняты принципиальные решения, определяющие этот процесс, а также качество и трудоемкость разработки.

К таким решениям относят:

- выбор структуры ПО;
- выбор типа пользовательского интерфейса и технологии работы с документами;
- выбор подхода к разработке (структурного или объектного);
- выбор языка и среды программирования.

Эти решения определяют, что проектируется, с какими потребительскими характеристиками, как и какими средствами.

Часть решений может быть определена в техническом задании, образовав группу технологических требований, остальные должны быть приняты как можно раньше, так как представляют собой исходные данные для процесса проектирования.

Выбор архитектуры ПО. *Архитектура ПО - совокупность базовых концепций (принципов) его построения. Архитектура ПО определяется сложностью решаемых задач, степенью универсальности разрабатываемого ПО и числом одновременно работающих пользователей.*

Различают архитектуры:

- *однопользовательскую – ПО рассчитано на одного пользователя, работающего за ПК;*
- *многопользовательскую – ПО рассчитано на работу в локальной или глобальной сети. Реализуют системы, построенные по принципу «клиент-сервер»*

В рамках однопользовательской архитектуры различают: программы; пакеты программ; программные комплексы; программные системы.

Программа - адресованный компьютеру набор инструкций, точно описывающий последовательность действий, которые необходимо выполнить для решения конкретной задачи. Программа представляет собой отдельно компилируемую программную единицу, которая может использовать стандартные библиотеки подпрограмм, но, как правило, не организует свои. Это самый простой вид архитектуры, для решения небольших задач.

Пакеты программ - совокупность программ, решающих задачи некоторой прикладной области (пакет графических или математических программ). Программы пакета связаны между собой только принадлежностью к определенной прикладной области. Пакет программ реализуют как набор отдельных программ, каждая из которых сама вводит необходимые данные и выводит результаты. По сути, пакет программ - это некоторая библиотека программ.

Программные комплексы - совокупность программ, совместно обеспечивающих решение небольшого класса сложных задач одной прикладной области. Для решения задачи может потребоваться решить несколько подзадач, последовательно вызывая программы комплекса. Вызов программ в программном комплексе осуществляется специальной программой – диспетчером, который обеспечивает несложный интерфейс с пользователем и, возможно, выдачу некоторой справочной информации.

От пакета программ программный комплекс отличается еще и тем, что несколько программ могут последовательно или циклически вызываться для решения одной задачи.

Программные системы - организованная совокупность программ (подсистем), позволяющую решать широкий класс задач из некоторой прикладной области. В отличие от программных комплексов программы, входящие в программную систему, взаимодействуют через обилие данные. Программные системы имеют развитые пользовательский и внутренние интерфейсы, что требует их тщательного проектирования.

Многопользовательские программные системы в отличие от

программных систем должны организовывать сетевое взаимодействие отдельных компонентов ПО, что еще усложняет процесс его разработки.

Для разработки подобного ПО используют специальные технологии или платформы (CORBA, COM, Java и т.п.).

Выбор типа пользовательского интерфейса. *Типы ПИ:*

- *примитивные – реализуют единственный сценарий работы, например, ввод данных – обработка – вывод результатов;*
- *меню – реализуют множество сценариев работы, операции которых организованы в иерархические структуры, например, «вставка»: «вставка файла», «вставка символа» и т.д.;*
- *со свободной навигацией – реализуют множество сценариев, операции которых не привязаны к уровням иерархии и предполагают определение множества возможных операций на конкретном шаге работы; такие интерфейсы используют Windows-приложения;*
- *прямого манипулирования – реализуют множество сценариев, представленных в операциях над объектами, основные операции инициируются перемещением пиктограмм объектов мышью.*

Тип ПИ во многом определяет сложность и трудоемкость разработки..

Появление ОО визуальных сред разработки ПО, использующих событийный подход к программированию и рассчитанных на создание интерфейсов со свободной навигацией, снизило трудоемкость разработки ПИ.

Выбор подхода к разработке. Если выбран интерфейс со свободной навигацией или прямого манипулирования, то это предполагает использование событийного программирования и объектного подхода, современные среды визуального программирования предоставляют интерфейсные компоненты в виде объектов библиотечных классов. В зависимости от сложности предметной области ПО может реализовываться с использованием объектов и классов и чисто процедурно. Исключение - случаи использования специализированных языков разработки Интернет-приложений (Perl), построенных по совершенно другому принципу.

Примитивный интерфейс и интерфейс типа меню совместимы со структурным и с объектным подходами к разработке. Поэтому выбор подхода осуществляют с использованием дополнительной информации.

Практика показывает, что объектный подход эффективен для разработки больших программных систем (более 100.000 операторов) и в случаях, когда объектная структура предметной области ярко выражена. Объектный подход необходимо осторожно использовать при жестких ограничениях на эффективность разрабатываемого ПО, например, при разработке систем резального времени.

Во всех прочих случаях выбор подхода остается за разработчиком.

Выбор языка программирования. В большинстве случаев проблемы выбора ЯП реально не существует. Язык может быть определен:

- - организацией, ведущей разработку;
- - программистом, который по возможности будет использовать знакомый язык;
- - устоявшимся мнением и т.п.
- ЯП можно разделить на группы:
- универсальные языки высокого уровня;
- специализированные языки разработчика программного обеспечения;
- специализированные языки пользователя;
- языки низкого уровня.

Выбор среды программирования. *Среда программирования* - программный комплекс, который специализированный текстовый редактор, встроенные компилятор, компоновщик, отладчик, справочную систему и другие программы, использование которых упрощает процесс написания и отладки программ.

- Распространены среды визуального программирования (Delphi, C++ Builder, Visual C++, Visual Studio), в которых программист получает возможность визуального подключения к программе некоторых кодов из специальных библиотек компонентов, что стало возможным с развитием ООП.

Выбор между этими средами должен определяться характером проекта.

Выбор или формирование стандартов разработки.
Современная технология проектирования должна обеспечивать соответствие стандарту ISO/IEC 12207: 1995 (поддержка всех процессов ЖЦ ПО).

Реальное применение любой технологии проектирования ПО требует формирования или выбора ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта.

Стандарты: стандарт проектирования;

- стандарт оформления проектной документации;
- стандарт интерфейса пользователя.

Стандарт проектирования должен определять:

- набор необходимых моделей (схем, диаграмм) на каждой стадии проектирования и степень их детализации;
- правила фиксации проектных решений на диаграммах, правила именования объектов и соглашения по терминологии, набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм и т.д.;
- требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы и используемых CASE-средств;
- механизм обеспечения совместной работы над проектом, в том числе и правила интеграции подсистем проекта и анализа

Стандарт оформления проектной документации регламентирует:

- комплектность, состав и структуру документации на каждой стадии;
- требования к оформлению документации (требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.);
- правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков на каждой стадии;
- требования к настройке CASE – средств для обеспечения подготовки документации в соответствии с правилами.

Стандарт интерфейса пользователя должен определять:

- правила оформления экранов (шрифты и цветовую палитру), состав и расположение окон и элементов управления;
- правила пользования клавиатурой и мышью;
- правила оформления текстов помощи;
- перечень стандартных сообщений;
- правила обработки реакции пользователя.

Все описанные выше проектные решения существенно влияют на трудоемкость и сложность разработки.

Только после их принятия следует переходить к анализу требований и разработке спецификаций проектируемого ПО.

Стиль оформления программы

Хорошим считают стиль оформления программы, облегчающий ее восприятие как самим автором, так и другими программистами.

«*Помните, программы читаются людьми*», Д. Ван Тассел.

Стиль оформления программы включает:

- правила именования объектов программы (переменных, функций, типов, данных и т. п.);
- правила оформления модулей;
- стиль оформления текстов модулей.

Правила именования объектов программы

При выборе имен программных объектов следует придерживаться правил:

- имя объекта должно соответствовать его содержанию:

MaxItem - максимальный элемент;

NextItem - следующий элемент;

- использовать символ «_» для визуального разделения имен, состоящих из нескольких слов:

Max_Item, Next_Item;

- необходимо избегать близких по написанию имен:

Index и *InDec.*

Правила оформления модулей

Каждый модуль должен предваряться заголовком, который содержит:

- **название модуля;**
- **краткое описание его назначения;**
- краткое описание входных и выходных параметров *с указанием единиц измерения;*
- список используемых (вызываемых) модулей;
- краткое описание алгоритма (метода) и/или ограничений;
- ФИО автора программы:
- идентифицирующую информацию (номер версии и/или дату последней корректировки).

Стиль оформления текстов модулей

определяет использование отступов, пропусков строк и комментариев, облегчающих понимание программы.

Пропуски строк и комментарии используют для визуального разделения частей модуля.

Для языков Pascal, C++ и Java, использование отступов позволяет прояснить структуру программы: обычно дополнительный отступ обозначает вложение операторов языка

Комментарии

Переводить с английского языка каждый оператор программы не нужно. Комментировать следует цели выполнения действий, а также группы операторов, связанные общим действием, т.е. комментарии должны содержать некоторую дополнительную (неочевидную) информацию, например:

```
{проверка условия и выход, если условие не выполняется}
if  n<0  then
    begin
     WriteLn('Количество отрезков отрицательно');
     exit;
    end;
```

Для языков низкого уровня (Ассемблера) может оказаться целесообразным комментировать и блоки операторов, и каждый оператор

Способы уменьшения времени выполнения

Необходимо анализировать циклические участки программы с большим количеством повторений.

При их написании необходимо:

- выносить вычисление константных, т.е. не зависящих от параметров цикла, выражений из циклов;
- избегать «длинных» операций умножения и деления, заменяя их сложением, вычитанием и сдвигами:
- минимизировать преобразования типов в выражениях;
- оптимизировать запись условных выражений – исключать лишние проверки;
- исключать многократные обращения к элементам массивов по индексам (особенно многомерных, так как при вычислении адреса элемента используются операции умножения на значение индексов);
- избегать использования различных типов в выражении и т. п.

Программирование «с защитой от ошибок»

Любая из ошибок программирования, которая не обнаруживается на этапах компиляции и компоновки программы, в конечном счете может проявиться тремя способами: привести к выдаче системного сообщения об ошибке, «зависанию» компьютера и получению неверных результатов.

Целесообразно проверять:

- правильность выполнения операций ввода-вывода;
- допустимость промежуточных результатов (значений управляющих переменных, значений индексов, типов данных, значений числовых аргументов и т.д.).