**Курсовая работа**

**"Этапы разработки программы на языке программирования"**

# Введение

Программы небольшого и среднего размера (несколько тысяч строк) создаются, как правило, в два этапа. Сначала необходимо точно установить, что надо сделать, продумать соответствующий алгоритм, определить структуры данных, объекты и взаимодействие между ними (это этап системною анализа)*,* а затем выразить этот алгоритм в виде, понятном машине (этап кодирования)*.* Если же разрабатывается крупный проект объемом от десятков тысяч до миллионов строк кода, тогда приходится применять специальные методологии проектирования, охватывающие период разработки ПО*.*

# Постановка задачи

Решение любой задачи начинается с построения модели. Иными словами, процесс построения модели называется постановкойзадачи. На содержательном уровне постановка задачи состоит в выявлении всех существенных факторов и связей этих факторов с требуемым результатом. Не менее важна и другая сторона этого процесса, как говорят, формальная – запись всех выявленных факторов и связей на языке, допускающем лишь однозначное толкование информации. Более того, имея в виду применение компьютера, надо записать формулировку задачи и процесс ее решения на языке, понятном не только человеку, но и компьютеру, т.е. на языке программирования.

Преобразование задачи, полученной на этапе ее постановки, в такую задачу, которая вписывается в рамки языка программирования, будем называть формализацией задачи.

Формализация нередко понимается шире, чем это указано выше, – как обособление имени объекта от самого объекта и возможность манипулирования с ним как с самостоятельным объектом. В компьютере мы как раз имеем дело с манипулированием кодами объектов, а не с самими объектами.

Как видно из этих рассмотрений, формализация тесно связана с. феноменом языка. Напомним, что формализованнымназывается язык, в котором за каждым словом закреплены ровно одно значение и ровно один смысл. Язык называется формальным,если жестко фиксирован алфавит и грамматика языка.

Исходная задача, как правило, формулируется на естественном языке. Построение модели преследует простую цель выделения таких предположений, при которых можно будет однозначно определить, что в данной задаче служит исходными данными и что результатом. Иными словами, построение модели сопровождается формализацией языка, посредством которого фиксируется информация, составляющая суть строящейся модели. Это еще раз подтверждает тезис, что язык всякой науки формализован, ибо любая наука занимается тем, что создает модели, дающие человеку знания об окружающем его мире.

Привлечение компьютера для решения задачи, для которой уже проведена формализация, требует перехода к формальному языку, поскольку компьютер, как формальный исполнитель, понимает только формальный язык.

Сказанное означает, что между указанными видами языков – естественным, формализованным и формальным существуют взаимосвязи. А именно: переход от описания объекта (процесса или явления) на естественном языке к описанию на формализованном языке является формализацией;переход от описания на формализованном языке к описанию на формальном называется кодированием.

# 

# Структуры данных

Данные, хранящиеся в памяти компьютера представляют собой совокупность нулей и единиц (битов). Биты объединяются в последовательности: байты, слова и т.д. Каждому участку оперативной памяти, который может вместить один байт или слово, присваивается порядковый номер (адрес).

Какой смысл заключен в данных, какими символами они выражены – буквенными или цифровыми, что означает то или иное число – все это определяется программой обработки. Все данные необходимые для решения практических задач подразделяются на несколько типов, причем понятие тип связывается не только с представлением данных в адресном пространстве, но и со способом их обработки.

Любые данные могут быть отнесены к одному из двух типов: основному (простому), форма представления которого определяется архитектурой ЭВМ, или сложному, конструируемому пользователем для решения конкретных задач.

Данные простого типа это – символы, числа и.т. п. элементы, дальнейшее дробление которых не имеет смысла. Из элементарных данных формируются структуры (сложные типы) данных.

Некоторые структуры:

Массив (функция с конечной областью определения) – простая совокупность элементов данных одного типа, средство оперирования группой данных одного типа. Отдельный элемент массива задается индексом. Массив может быть одномерным, двумерным и т.д. Разновидностями одномерных массивов переменной длины являются структуры типа кольцо, стек, очередь и двухсторонняя очередь.

Запись (декартово произведение) – совокупность элементов данных разного типа. В простейшем случае запись содержит постоянное количество элементов, которые называют полями. Совокупность записей одинаковой структуры называется файлом. (Файлом называют также набор данных во внешней памяти, например, на магнитном диске). Для того, чтобы иметь возможность извлекать из файла отдельные записи, каждой записи присваивают уникальное имя или номер, которое служит ее идентификатором и располагается в отдельном поле. Этот идентификатор называют ключом.

Такие структуры данных как массив или запись занимают в памяти ЭВМ постоянный объем, поэтому их называют статическими структурами. К статическим структурам относится также множество.

Имеется ряд структур, которые могут изменять свою длину – так называемые динамические структуры. К ним относятся дерево, список, ссылка.

Важной структурой, для размещения элементов которой требуется нелинейное адресное пространство является дерево. Существует большое количество структур данных, которые могут быть представлены как деревья. Это, например, классификационные, иерархические, рекурсивные и др. структуры.

# 

# Основные этапы компьютерного моделирования

## Постановка

Как уже было сказано, постановка задачи является самым первым и наиболее ответственным этапом, поскольку даже небольшая ошибка может привести к тому, что вся дальнейшая работа окажется ненужной. Именно на этом этапе определяется вид входной информации, подлежащей обработке компьютером, и то, что требуется получить после работы программ.

## Алгоритмизация.

Для обработки информации требуется подобрать уже известные или разработать новые алгоритмы. Фактически это первый этап формализации. Поскольку это очень важный процесс, от которого зависит не только скорость обработки информации, но и соответствие постановке, он выделяется в отдельную часть. Существует весьма сложная и объемная математическая дисциплина – «Теория алгоритмов», предметом изучения которой, в частности, как раз и является этот этап.

## Кодирование.

Перевод алгоритмически описанной модели на язык программирования. Именно на этом этапе применяется нисходящее проектирование.

## Комплексное тестирование и отладка.

Программа, написанная на одном из языков программирования запускается на компьютере, находятся ошибки и несоответствия с начальной постановкой и с алгоритмами, которые пользовались при решении. В случае необходимости выбираются другие алгоритмы или (что самое неприятное) вообще переделывается постановочная часть. Иными словами, компьютерная модель должна быть адекватна реальному процессу.

В случае несоответствия модели реальному процессу необходимо вернуться к одному из предыдущих этапов. Таким этапом может быть любой – от постановки до кодирования. После внесения изменений вся цепочка разработки программы проходится заново. Если результаты соответствуют экспериментальным данным, программу можно запускать в промышленную эксплуатацию.

Сопровождение.

Никакое самое тщательное тестирование и отладка не способны выявить абсолютно все ошибки и несоответствия. Кроме того, в процессе эксплуатации программы выясняется, что в ней не предусмотрены некоторые функциональные возможности, которые были бы крайне полезны пользователю,

Все это приводит к тому, что требуется специальная служба, систематизирующая все такие ситуации и обеспечивающая разработку новых, более совершенных версий программы.

Алгоритм и программа

Управлять компьютером нужно по определенному алгоритму*.* Алгоритм – это точно определенное описание способа решения задачи в виде конечной (по времени) последовательности действий. Такое описание еще называется формальным*.* Для представления алгоритма в виде, понятном компьютеру, служат языки программирования*,* Сначала всегда разрабатывается алгоритм действий, а потом он записывается на одном из таких языков. В итоге получается текст программы – полное, законченное и детальное описание алгоритма на языке программирования. Затем этот текст программы специальными служебными приложениями, которые называются трансляторами*,* либо переводится в машинный код, либо исполняется.

# Написание программы на языке программирования

программа язык моделирование

## Язык программирования

Самому написать программу в машинном коде весьма сложно, причем эта сложность резко возрастает с увеличением размера программы и трудоемкости решения нужной задачи. Условно можно считать, что машинный код приемлем, если размер пр6граммы не превышает нескольких десятков байтов и нет потребности в операциях ручного ввода / вывода данных.

Поэтому сегодня практически все программы создаются с помощью языков программирования. Теоретически программу можно написать и средствами обычного человеческого (естественного) языка – это называется программированием на метаязыке(подобный подход обычно используется на этапе составления алгоритма), но автоматически перевести такую программу в машинный код пока невозможно из-за высокой неоднозначности естественного языка.

Языки программирования – искусственные языки. От естественных они отличаются ограниченным числом «СЛОВ», значение которых понятно транслятору, и очень строгими правилам и записи команд*.* Совокупность подобных требований образует синтаксисязыка программирования, а смыслкаждой команды и других конструкций языка – его семантику*.* Нарушение формы записи про граммы приводит к тому, что транслятор не Может понять назначение оператора и выдает сообщение о синтаксической ошибке, а правильно написанное, но не отвечающее алгоритму использование команд языка приводит к семантическим ошибкам (называемым еще логическими ошибками или ошибками времени выполнения).

Процесс поиска ошибок в программе называется тестированием*,* процесс устранения ошибок – отладкой.

## Уровни языков программирования

Разные типы процессоров имеют разные наборы команд, Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкою уровня*.* В данном случае «низкий уровень» не значит «ПЛОХОЙ». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

Языком самого низкого уровня является язык ассемблера*,* который просто представляет каждую команду машинного кода, но не в виде чисел, а с помощью символьных условных обозначений, называемых мнемониками,Однозначное преобразование одной машинной инструкции в одну команду ассемблера называется транслитерацией*.* Так как наборы инструкций для каждого модели процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык ассемблера, и написанная на нем про грамма может быть использована только в этой среде.

С помощью языков низкого уровня создаются очень эффективные и компактные про граммы, так как разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а результирующая программа не может быть перенесена 'На компьютер с другим типом процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие требующие интенсивных вычислений алгоритмы обработки изображений.

Языкипрограммированиявысокого уровнязначительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных И мощных команд значительно проще, а ошибок при создании про грамм допускается гораздо меньше.

# Комплексная отладка программы

Тестирование при комплексной отладке – применение программы к конкретным данным, которые могут возникнуть у пользователя, но, возможно, в моделируемой (а не в реальной) среде.

Тестирование архитектуры программного средства. Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ программного средства. К моменту начала тестирования архитектуры программы должна быть уже закончена автономная отладка каждой подсистемы.

Тестирование внешних функций. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ программного средства. Несмотря на то, что все эти программы автономно уже отлажены, указанные расхождения могут быть,

Тестирование качества программы. Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества программного средства. Завершенность программы проверяется уже при тестировании внешних функций.

Тестирование документации по применению программы. Целью тестирования является поиск несогласованности документации по применению и совокупностью программ программного средства а также выявление неудобств, возникающих при применении программы. Этот этап непосредственно предшествует подключению пользователя к завершению разработки программного средства.

Тестирование определения требований к программе. Целью тестирования является выяснение, в какой мере программного средства не соответствует предъявленному определению требований к нему. Особенность этого вида тестирования заключается в том, что его осуществляет организация-покупатель или организация-пользователь. Обычно производится с помощью контрольных задач – типовых задач, для которых известен результат решения.

Организация тестирования в объектно-ориентированных системах

Тестирование является достаточно независимым процессом, применимым к программному обеспечению, разработанному с помощью любого метода проектирования. Тем не менее объектно-ориентированный подход привносит свои особенности.

Традиционно тестирование делится на тестирование элементов, интеграционное тестирование и системное тестирование.

На уровне элементов тестирование объектно-ориентированных программ отличается по следующим показателям:

* определение единиц тестирования;
* тестирование наследования;
* тестирование полиморфизма.

Естественной единицей тестирования является класс. Разбиение его на более мелкие элементы (методы) нецелесообразно, поскольку они не существуют отдельно от классов. Иногда за единицу тестирования принимается тесно связанная группа классов.

Тестирование наследования состоит в тестировании методов, унаследованных классом от своего базового класса. Если базовый класс уже прошел тестирование, нужно ли повторять тестирование для унаследованных методов – Вопреки достаточно распространенным надеждам программистов перетестирование необходимо. Основная причина та, что методы выполняются в новом контексте.

Применимость тестовых сценариев базового класса для тестирования производного класса зависит от того, является ли наследование классификацией. Если нет, то даже для унаследованных методов необходимо разрабатывать новые тестовые сценарии.

Тестирование полиморфизма сходно с тестированием наследования в том, что в тестовых сценариях необходимо предусмотреть все варианты связывания, т.е. все варианты конкретной реализации полиморфизма.

Интеграционное тестирование представляет собой тестирование того, как отдельные элементы программы работают вместе. Свойства объектно-ориентированных языков исключают целый ряд возможных ошибок, прежде всего за счет строгого определения внешних интерфейсов классов и объектов. Однако это не означает, что интеграционное тестирование становится легче. Планирование интеграционного тестирования немного отличается.

При традиционном тестировании имеется такая характеристика, как степень покрытия кода. При тестировании по принципу открытого, или белого ящика (т.е. в случае, когда тестирующему известна структура кода) необходимо обеспечить прохождение управления по всем ответвлениям программы. Иными словами, требуется выполнить как можно больший процент инструкций, имеющихся в программе. Наряду с этой характеристикой планирование тестов для объектно-ориентированной программы должно включать «покрытие состояний».

То, что объект соединяет в себе состояние и поведение, обусловливает необходимость проверки всех возможных переходов из состояния в состояние. В планировании таких тестов должна помочь модель состояний класса, построенная на этапе анализа и проектирования.

Системное тестирование проверяет всю программную систему целиком и строится в большинстве случаев по принципу «черного ящика». Тестирующий знает только внешние характеристики системы, но не знает, как она работает.

Построение требований к системе в форме вариантов использования обеспечивает естественный и простой способ планирования системного тестирования. Фактически система вариантов использования становится основой плана тестов на этапе системного тестирования.

# 

# Программная документация

Пожалуй, самым неприятным и тяжелым этапом программистской работы является создание программной документации. К сожалению, обычно этому либо не учат совсем, либо, в лучшем случае, не обращают на качество получаемых документов должного внимания. Тем не менее, владение этим искусством является зачастую одним из важнейших фактором, определяющим качество программиста.

Во-первых, умение создавать программную документацию определяет профессиональный уровень программиста. Заказчик не будет вникать в тонкости и особенности даже самой замечательной программы. Заказчик будет сначала читать документацию. Большую роль играет в этом и психологический фактор. В частности, во всем мире ценилась (и ценится сейчас) былая советская школа программирования. Современные же отечественные программисты котироваться перестали. Класс не тот. Нынче программы уже не пишутся, а составляются (а это – «две большие разницы»). Так вот, созданный в «классическом» стиле пакет программной документации (далее – ПД) создаст у вашего заказчика или работодателя самое что ни на есть благоприятное впечатление. Тем более, если автор ПД будет избегать фраз вида «кликните на скроллбар…», «винт» и т.п. К сожалению, за подобной жаргонной трескотней обычно скрывается либо скудость мыслей, либо полная пустота (неизгладимое впечатление произвел на автора рассказ одного его знакомого о неком «геймере», который с кем-то там то ли «чатился», то ли «модераторством» занимался или что-то в этом роде.). Язык ПД – это своего рода бюрократический, весьма консервативный язык. Есть в нем своя особая прелесть. Согласитесь, что термины НЖМД, НГМД, ручной манипулятор типа «мышь» (или «колобок», как значилось в одном из старинных пакетов ПД) звучат совсем иначе, нежели соответствующие «винт», «флоп» и просто «мышь». Между прочим, дело уже дошло до того, что, говорят, появилась даже особая специальность – технический писатель, т.е. человек, умеющий создавать программную документацию.

Во-вторых, грамотно составленный (точнее, созданный) пакет ПД избавит вас от многих неприятностей. В частности, избавиться от назойливых вопросов и необоснованных претензий можно просто отослав пользователя к документации. Это касается прежде всего важнейшего документа – Технического задания. Об этом мы будем говорить ниже, а сейчас можно напомнить о многомиллионном иске к компании IBM. Этот иск предъявило одно крупное издательство, неудовлетворенное качеством ВТ и программного обеспечения. IBM суд выиграла. И выиграла только благодаря тому, что предъявила подписанное обеими сторонами Техническое задание. Было это давно, еще в 70-х гг., однако сути дела это не меняет.

И еще одно. Важно создать первый пакет ПД. Этого будет достаточно, чтобы на его основе строить все последующие, используя его как образец или шаблон. Но сделать это надо очень качественно. Не спеша. Очень основательно.

Для начала необходимо вооружиться ГОСТами. ГОСТ определяет все. В частности, в него входит и интересующая нас Единая система программной документации (ЕСПД).

Техническое задание оформляют на листах формата А4 и / или А3, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

Для внесения изменений и дополнений в техническое задние на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

1. Наименование и область применения;

2. Основание для разработки;

3. Назначение разработки;

4. Технические требования к программе или программному изделию;

5. Технико-экономические показатели;

6. Стадии и этапы разработки;

7. Порядок контроля и приемки;

8. Приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

# 

# Список литературы

1. Информатика базовый курс. 2-е издание / Под ред. С.В. Симоновича. – Спб.: Питер, 2008. – 640 с.: ил
2. Гейн А.Г., Сенокосов А.И. справочник по информатике для школьников. – Екатеринбург: «У-Фактория», 2003. – 346 с.