Министерство образования и науки Украины
Севастопольский национальный технический университет

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ
по дисциплине «**Технология программирования**»
для студентов дневной и заочной форм обучения
по направлению подготовки 6.0915 –
«Компьютерная инженерия»

Севастополь
2006

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

УДК 004.413.5

Методические указания к выполнению лабораторных работ по дисциплине “Технология программирования” /Сост. С.А. Бражников, М.А. Лебедева, С.Н. Фисун. - Севастополь: Изд-во СевНТУ, 2006.- 38 с.

Целью методических указаний является оказание помощи студентам в подготовке и выполнении лабораторных работ по индивидуальным заданиям.
Методические указания предназначены для студентов всех форм обучения по специальности 7.091501 “Компьютерные системы и сети”.

Методические указания рассмотрены и утверждены на заседании кафедры кибернетики и вычислительной техники (протокол № 11 от 26.12. 2005 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент: Бондарев В.Н., кандидат технических наук, доцент кафедры информационных систем.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

СОДЕРЖАНИЕ

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

1. ПОРЯДОК ВЫБОРА ВАРИАНТА ЗАДАНИЯ И СОДЕРЖАНИЕ ОТЧЕТА

В качестве индивидуального задания студент выбирает программу, которую он разработал и отладил при изучении предыдущих дисциплин цикла “Программное обеспечение”(“Программирование”, “Системное программирование”, “Вычислительный практикум” и др.).

Индивидуальное задание может быть написано на языке СИ или ПАСКАЛЬ и согласовано с преподавателем. Это задание является сквозным для выполнения практически всех лабораторных работ. По согласованию с преподавателем для выполнения конкретных лабораторных работ индивидуальное задание может быть откорректировано (выбраны отдельные модули и т.д.).

Отчет должен содержать:

- титульный лист;
- схему алгоритма или текст программы;
- порядок выполнения действий;
- необходимые расчеты;
- результаты выполнения;
- выводы.

2. ЛАБОРАТОРНАЯ РАБОТА № 1 РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Цель работы: научиться составлять техническое задание (ТЗ) на разработку программного продукта.

2.1 .Теоретические сведения

Содержание ТЗ должно включать следующие разделы:

Введение
1 .Основание для разработки
2 .Назначение разработки
3 .Требования к программе
4 .Требования к программной документации
5 .Технико-экономические показатели
6 .Стадии и этапы разработки
7 .Порядок контроля и приемки

В разделе “Введение” указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором будут использовать программу.

1. В разделе “Основания для разработки” указывается документ, на основании которого ведется разработка (приказ по университету, задание на лаб. работу и т.п.); организация, утвердившая документ и дата его утверждения; наименование темы разработки.

2. В разделе “Назначение разработки” указывают функциональное

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

и эксплуатационное назначение программы.

3. Раздел “Требования к программе” должен содержать следующие подразделы:

3.1. “Требования к функциональным характеристикам” – состав функций, которые будет выполнять программа, организация входных и выходных данных (синтаксис и семантика входных данных, форматы выходных сообщений и соответствующие им ситуации), временные характеристики. В этом подразделе должно быть описано поведение системы с точки зрения соотношения входа и выхода без конкретизации внутренней структуры и реакция программы на непредусмотренные данные на входе.

3.2. “Требования к надежности” – контроль входных и выходных данных, последствия возможных отказов, время восстановления, защита от несанкционированного доступа и др.

3.3. “Условия эксплуатации” – характеристики операционной среды, вид обслуживания, количество и квалификация персонала, затрачиваемое время процессора и каналов связи, число пользователей и др., а также допустимые параметры окружающей среды.

3.4. “Требования к составу и параметрам технических средств” – конфигурация системы, основные характеристики требуемых устройств.

3.5. “Требования к информационной и программной совместимости” - требования к методам решения, языкам программирования, программным средствам, используемых системой, протоколам обмена, к СУБД и операционным системам.

3.6. “Требования к маркировке и упаковке” – варианты и способы упаковки (обычно специальных требований не предъявляется).

3.7. “Требования к транспортированию и хранению” – места хранения, условия и сроки, способы создания и хранения резервных копий (обычно специальных требований не предъявляется).

Отдельные разделы и подразделы по согласованию с заказчиком могут быть опущены.

4. В разделе “Требования к программной документации” – состав документации и специальные требования к ней. Виды программных документов: спецификация, текст программы, описание программы, пояснительная записка, ТЗ, программа и методика испытаний, руководство программиста, руководство системного программиста, руководство оператора, руководство по техническому обслуживанию и т.д.

5. В разделе “Технико-экономические показатели” – экономические преимущества, предполагаемая экономическая эффективность и годовая потребность, экономические преимущества разработки, предельный объем программы, время реакции программы.

6. В разделе “Стадии и этапы разработки” - перечень стадий, разбивка на этапы, содержание, сроки разработки (дни, недели и т.д.) и исполнители.

7. В разделе “Порядок контроля и приемки – виды испытаний, тре-

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

бования к приемке работ, способы проверки важнейших характеристик. Цель испытаний – установление степени соответствия готового продукта и характеристикам технического задания.

Формы представления результатов: программная документация, конструкторская документация на изделие, программное изделие.

В приложении приводят перечень проведенных научных и исследовательских работ, схемы алгоритмов, таблицы, описания и т.д.

2.2. Задание на лабораторную работу

Для выбранного по индивидуальному заданию программного продукта разработать техническое задание в соответствии с ГОСТ 19.201-78, предполагая, что сначала разрабатывается ТЗ, а затем будет написана программа для ТЗ. Отчет по лабораторной работе должен содержать разделы технического задания.

2.3. Список контрольных вопросов

1. Назначение технического задания?
2. Кто составляет и утверждает ТЗ?
3. На каком этапе разработки программного изделия составляется ТЗ?
4. Какими документами регламентируется написание ТЗ?

3. ЛАБОРАТОРНАЯ РАБОТА № 2
РАСЧЕТ МЕТРИК ХОЛСТЕДА

Цель работы: Изучение основ метрической теории программ Холстеда, расчет количественных характеристик для индивидуального модуля.

3.1. Описание метрик Холстеда

Метрики Холстеда предлагают разумный подход к решению следующих задач:

- предсказание условий, необходимых для программирования по предложенным проектам;
- определение норм первоначальных ошибок;
- количественная оценка языков программирования и эффекта модульности;
- обоснование метода измерения различий между программами, написанными специалистами разного уровня.

В основе вычисления метрик Холстеда лежит концепция, согласно которой алгоритм состоит только из операторов и операндов (проверяется рассмотрением простых вычислительных машин с форматом команд, содержащим две части: код операции и адрес операнда). Операнды - переменные или константы, используемые в данной реализации алгоритма. Операторы - комбинации символов, влияющие на значение или порядок

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

операндов.

В основе вычисляемых свойств алгоритма лежат следующие характеристики:

- n1 - число различных операторов данной реализации;
- n2 - число различных операндов данной реализации;
- N1 - общее число всех операторов;
- N2 - общее число всех операндов;
- n2*- число различных входных и выходных операндов.

На основании приведенных выше характеристик вычисляются:

- словарь n = n1 + n2;
- длина реализации N = N1 + N2.

Метрики Холстеда включают следующие характеристики:

1) Длина программы: N'= n1 * log2(n1) + n2 * log2(n2).

Если программа состоит из нескольких модулей (m - число модулей), то для каждого модуля определяется n1 среднее ($n1_{ср}$) и n2 среднее ($n2_{ср}$). В этом случае длина программы

$N' = m*(n1_{ср} * \log2(n1_{ср}) + n2_{ср} * \log2(n2_{ср}))$.

2) Объем программы: V = N * log2(n).

Такая интерпретация дает объем программы в битах. Объем зависит от языка программирования, на котором реализован алгоритм.

3) Потенциальный (минимальный) объем: V* = (2+ n2*) * log2(2+ n2*).

Минимально возможный объем предполагает существование языка, в котором действия, выполняемые индивидуальным модулем, уже определены или реализованы, возможно, в виде процедуры или функции.

4) Граничный объем: V** = (2+ (n2*)* log2(n2*)) * log2(2+ n2*).

5) Соотношения между операциями и операндами (зависимость числа операндов n2 от числа операций n1: A = n2* /(n2* +2) * log2(n2*/2)

$$B = n2^* - 2 * A$$

$$n2 = A * n1 + B$$

- это частота использования операндов.

6) Уровень программы: L = V*/V,

где V* - потенциальный объем, V - объем программы.

L=1 для потенциального языка, в котором присутствует любая процедура, которая могла бы понадобиться (число таких процедур близко к бесконечности).

Альтернативное определение уровня L: L' = (n1*) * n2/(n1*N2), где n1*=2.

7) Интеллектуальное содержание: I = L' * V или: I = 2 * n2/(n1 * N2)*N * log2(n).

Интеллектуальное содержание - мера того, "сколько было сказано в программе", зависит от сложности задачи. (I≈11-13).

8) Работа по программированию (общее число элементарных мысленных различий, требуемых для порождения программы): E=V/L или $E = V^2/V^*$,

где E – общее число элементарных умственных различений, требуемых для порождения программы.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Это умственная работа, затрачиваемая на превращение заранее разработанного алгоритма в фактическую реализацию на языке программирования.

Правильное разбиение на модули уменьшает работу по программированию:

E = E1 + E2 + E3 + ...

9) Приближенное время программирования: T' = E/S, где S = 18 моментов (различий)/секунд - постоянная Страуда.

Момент – время, требуемое человеческому мозгу для выполнения элементарных различений.

Альтернативное определение времени T: T' = n1*N2*N*log2(n)/(2*S*n2).

10) Уровень языка: A = L * L* V.

Уровень языка определяет его производительность.

11) Уравнение ошибок:

Число переданных ошибок в программе: B = V/E0,

где E0 = V* x V* x V*/(A x A) - среднее число элементарных различений между возможными ошибками в программировании.

«Переданные ошибки» - ошибки, остающиеся после отладки модуля.

3.2. Пример определения характеристик программы

Программа, реализующая алгоритм Евклида на Паскале:

```
program Evclid;
var a,b,q,r,gcd:integer;
begin
  readln(a,b);
  if a=0 then gcd:=b
        else
          begin
           r:=0;
           while r<>0 do
             begin
               q:=a div b;  r:=a-b*q;
               a:=b;  b:=r
             end;
           gcd:=a
         end;
    writeln(gcd)
end.
```

Подсчет характеристик программы:

Оператор	Номер	Число вхождений
readln	1	1
writeln	2	1
;	3	7
:=	4	7

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

begin...end	5	3
while...do	6	1
if	7	2
=	8	1
<>	9	1
div	10	1
-	11	1
*	12	1
	n1=12	N1=27

Операнд	Номер	Число вхождений
a	1	6
b	2	6
r	3	4
q	4	2
gcd	5	3
0	6	3
	n2=6	N2=24

Словарь n=n1+n2=12+6=18.
Длина реализации N=N1+N2=27+24=51.
Число входных и выходных операндов n2*=3.

3.3. Порядок выполнения работы

1. Для индивидуального модуля определить характеристики программы (n1, n2, n, N1, N2, N, n2*).
2. Рассчитать метрики Холстеда по формулам 1-11.
3. Оценить качество реализации алгоритма на основании метрик Холстеда.
4. Оформить отчет.

3.4 Список контрольных вопросов

1. Где можно использовать метрики Холстеда?
2. Чем определяются характеристики программы?
3. Как оценить качество реализации алгоритма по метрикам?
4. В чем недостаток программометрии?

4. ЛАБОРАТОРНАЯ РАБОТА № 3 ОПТИМИЗАЦИЯ ПРОГРАММ

Цель работы: Ознакомление с видами оптимизации программы, оптимизация индивидуального модуля по выбранному параметру (время выполнения, объем памяти).

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

4.1. Теоретические сведения

Оптимизация - преобразование программы, сохраняющее ее семантику (конструкции языка программирования), но уменьшающие ее размер и время выполнения.

Виды оптимизация программы:

- глобальная (всей программы);
- локальная (нескольких соседних операторов, образующих линейный участок);
- квазилокальная (фрагментов программы фиксированной структуры, например, циклов).

Способы оптимизации:

1. Разгрузка участков повторяемости: вынесение вычислений из многократно проходимых исполняемых участков программы на участки программы, редко проходимые. Таким образом, это преобразование тела цикла или рекурсивных процедур.
2. Упрощение действий: улучшение программы за счет замены групп вычислений на группу вычислений, дающих тот же результат с точки зрения всей программы, но имеющих меньшую сложность.

а) упрощение действий происходит при замене сложных операций в выражениях более простыми: x / 0.4 -> x*0.25;

б) преобразование по объединению или расчленению циклов, по перестановке заголовков циклов, по удалению избыточных выражений (замене их на переменную)

3. Реализация действия: действия над константами заменяются на константы; ликвидация константных распознавателей - замена условного оператора на одну из его ветвей, если его выбирающее условие- выражение имеет постоянное значение;

удаление из программы ненужных пересылок вида:

Y=F(W), X=Y на X=F(W)

4. Чистка программы (удаление ненужных конструкций): недостижимых операторов, существенных операторов, неиспользуемых переменных, видов, операций.
5. Сокращение размера программы: вынесение одинаковых конструкций в начальную или конечную точку программы; поиск в программе похожих объектов и формирование их в виде процедуры.
6. Экономия памяти - уменьшение объема памяти, отводимые под информационные объекты программы (например, параметры процедуры).

4.2. Порядок выполнения работы

1. Для индивидуального модуля выбрать параметр оптимизации и определить его количественные характеристики.
2. Провести оптимизацию программы по выбранному параметру.
3. Сравнить характеристики исходного модуля и модуля, полученного в результате оптимизации.
4. Оформить отчет, содержащий описание, обоснование и результаты оп-

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

тимизации программы.

4.3. Список контрольных вопросов

1. Почему необходимо проводить оптимизацию, а не минимизацию программы?
2. От чего зависит выбор метода оптимизации?
3. Почему большое внимание уделяется циклическим участкам?
4. К каким нежелательным последствиям может привести оптимизация?

5. ЛАБОРАТОРНАЯ РАБОТА № 4 СТРУКТУРНОЕ ТЕСТИРОВАНИЕ

Цель работы: изучение принципов тестирования на основании анализа структуры программы, формирование маршрутов для тестирования, разработка тестов по критерию покрытия всех ветвей графа управления программы (критерий С1).

5.1. Задачи и методы тестирования

Тестирование - процесс испытания программы тестами с целью обнаружения и локализации ошибок, а также проверки соответствия программы требованиям, сформулированным в ТЗ.

Основными задачами в процессе тестирования являются:

- планирование тестирования в соответствии с выбранными критериями;
- разработка конкретных тестовых значений и соответствующих им эталонов;
- тестирование программы;
- обработка и анализ результатов тестирования.

Существует два основных подхода к решению задачи тестирования:

а) функциональное тестирование - тесты строятся на основании функциональных свойств программы без учета ее внутренней структуры;

б) структурное тестирование - учитывается внутренняя структура программы, анализируется прохождение данных от входа к выходу, и эта информация используется для рациональной организации тестирования.

5.2. Структурное тестирование

При рассмотрении методов структурного тестирования удобно рассматривать программу в некоторой графической форме, например в виде графа или блок-схемы. Граф управления программы - совокупность вершин и дуг. Вершинам соответствуют условные утверждения программ (условия), а дугам - последовательности операторов присваивания и безусловного перехода. Последовательность дуг, ведущая от начальной вершины к конечной, называют путем или маршрутом.

Наиболее часто при тестировании структуры программных модулей используют следующие критерии выделения тестируемых маршрутов:

- критерий С1 - выполнение в процессе тестирования всех дуг графа программы;

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

- критерий B1 - покрытие границ условий. Для условных утверждений реализуются три граничных условия (0,e,-e);
- критерий C2 - покрытие всех пар соседних дуг графа программы (всех комбинаций вход-выход для вершин графа);
- критерий B2 - покрытие всех комбинаций "входная дуга - граница условия" для всех вершин (комбинация B1 и C2);
- критерий CP - покрытие всех пар достижимых дуг графа - критерий PH - покрытие всех возможных путей в графе программы при заданном ограничении на число повторений циклов.

Простейший критерий C1 - критерий покрытия всех ветвей графа управления программы. Он состоит в выборе минимального множества маршрутов программы, охватывающих все направления передач управления (ветвления при условных переходах). По этому критерию граф программы должен быть покрыт минимальным набором путей, проходящих через каждый оператор ветвления по каждой дуге. Повторная проверка дуг не оценивается и считается избыточной. При этом в процессе проверки гарантируется выполнение всех передач управления между операторами программы и каждого оператора не менее одного раза. Однако этот критерий не всегда надежен, поскольку не обеспечивает исполнения каждого оператора при различных сочетаниях предшествующих условий.

5. 3. Выполнение работы

1. Составить блок-схему (граф) индивидуального модуля. Считать вершинами - условия, а дугами - операторы присваивания. Обозначить дуги (например, латинскими буквами).
2. По графу выбрать минимальное множество маршрутов, соединяющих начальную и конечную вершины графа. Маршрут записать как последовательность латинских букв. Каждая дуга графа должна входить хотя бы в один из маршрутов. Таким образом, множество маршрутов будет включать все дуги графа.
3. Определить значения входных переменных (тестовый набор), соответствующий каждому выбранному маршруту. Тестовые наборы записать во входной файл.
4. Провести тестирование программы, выполнив ее для каждого тестового набора.
5. Оформить отчет.

Отчет должен включать:

- блок-схему с обозначенными дугами;
- набор маршрутов;
- соответствующие маршрутам тестовые наборы;
- результаты работы программы для каждого тестового набора;
- выводы.

5.4. Список контрольных вопросов

1. Как структурное тестирование учитывает конкретную реализацию программы?

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

2. Перечислить недостатки структурного тестирования?
3. С какой целью используется редуцированный граф?
4. Как учитывается кратность повторения циклов?

6. ЛАБОРАТОРНАЯ РАБОТА № 5
МЕТОД ЭКВИВАЛЕНТНЫХ РАЗБИЕНИЙ

Цель работы: изучение принципов тестирования программы на основании эквивалентного разбиения, выделение классов эквивалентности и построение тестов.

6.1. Теоретические сведения

Тесты, построенные по принципу эквивалентного разбиения, должны обладать двумя свойствами:

1) уменьшать, причем более чем на единицу, число других тестов, которые должны быть разработаны для достижения заранее определенной цели "приемлемого" тестирования, т.е. каждый тест должен включать столько различных входных условий, сколько это возможно, с тем, чтобы минимизировать число различных тестов;

2) покрывать значительную часть других возможных тестов, т.е. необходимо пытаться разбить входную область программы на конечное число классов эквивалентности так, чтобы можно было предположить, что каждый тест, являющийся представителем некоторого класса, эквивалентен любому другому тесту этого класса. Иными словами, если один тест класса эквивалентности обнаруживает ошибку, то и все другие тесты этого класса эквивалентности будут обнаруживать ту же самую ошибку. Наоборот, если тест не обнаруживает ошибки, то следует ожидать, что ни один тест этого класса эквивалентности не будет обнаруживать ошибки.

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа:

- выделение классов эквивалентности;
- построение тестов.

6.2. Выделение классов эквивалентности

Классы эквивалентности выделяются путем анализа каждого входного условия (обычно это предложение или фраза в спецификации, либо множество значений входной переменной) и разбиении его на две или более группы. Для проведения этой операции используют таблицу 6.1:

Таблица 6.1

Входные условия классы	Правильные классы Эквивалентности	Неправильные Эквивалентности

Различают два типа классов эквивалентности: правильные (представляющие правильные входные данные программы) и неправильные,

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

представляющие все другие возможные состояния (т.е. ошибочные входные значения). Выделение классов эквивалентности представляет собой эвристический процесс. При этом существует ряд правил:

1. Если входное значение описывает область значений (например, целое число может принимать значения от 1 до 99), то определяются один правильный класс эквивалентности (1 <= число <= 99) и два неправильных (число < 1 и число >99).
2. Если входное условие описывает число значений (например, "в автомобиле могут ехать от одного до шести человек"), то определяются один правильный класс эквивалентности и два неправильных (ни одного и более шести человек).
3. Если входное условие описывает множество входных значений и каждое значение программа трактует особо, то определяется один правильный класс эквивалентности для каждого значения («да») и один неправильный класс эквивалентности («нет»).
4. Если входное условие описывает ситуацию "должно быть" (например, первым символом идентификатора должна быть буква), то определяется один правильный класс эквивалентности (первый символ - буква) и один неправильный (первый символ - не буква).
5. Если есть основание считать, что различные элементы класса эквивалентности трактуются программой неодинаково, то данный класс эквивалентности разбивается на меньшие классы эквивалентности.

6.3. Построение тестов

Процесс построения тестов включает в себя:

1. Назначение каждому классу эквивалентности уникального номера.
2. Проектирование новых тестов, каждый из которых покрывает как можно большее число непокрытых правильных классов эквивалентности до тех пор, пока все правильные классы эквивалентности не будут покрыты. Если нет правильных классов, содержащих перечисления, то для всех правильных классов строится один тестовый набор.
3. Запись тестов, каждый из которых покрывает один и только один из непокрытых правильных классов эквивалентности, до тех пор, пока все неправильные классы эквивалентности не будут покрыты тестами.

Причина покрытия неправильных классов эквивалентности индивидуальными тестами состоит в том, что определенные проверки с ошибочными входами скрывают или заменяют другие проверки с ошибочными входами. Если тест отражает два ошибочных условия, то программа ограничится проверкой только первого условия, второе условие проверяться не будет.

6.4. Порядок выполнения работы

1. Выделить классы эквивалентности для индивидуального модуля. Для записи классов эквивалентности использовать таблицу 1.
2. Построить тесты по принципу эквивалентного разбиения.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

3. Выполнить программу для каждого тестового набора. Результаты работы программы запомнить.
4. Сравнить результаты, соответствующие правильным и неправильным классам эквивалентности.
5. Оформить отчет.

6.5. Список контрольных вопросов

1. Каковы принципы построения классов эквивалентности?
2. Как генерируются тесты для правильного класса эквивалентности и почему?
3. Как генерируются тесты для неправильных классов эквивалентности и почему?
4. Что дает разбиение на классы эквивалентности?

7 . ЛАБОРАТОРНАЯ РАБОТА N 6
МЕТОД ФУНКЦИОНАЛЬНЫХ ДИАГРАММ

Цель работы: Изучение метода функциональных диаграмм, построение функциональной диаграммы и таблицы решений для программного модуля, генерация тестов.

7.1. Особенности метода функциональных диаграмм

Метод функциональных диаграмм или диаграмм причинно-следственных связей предназначен для исследования комбинаций входных условий. Поскольку число комбинаций обычно велико, выбор произвольного подмножества может привести к неэффективному тесту. Метод функциональных диаграмм помогает систематически выбирать высокорезультативные тесты. Он дает полезный побочный эффект, так как позволяет обнаруживать неполноту и неоднозначность исходных спецификаций.

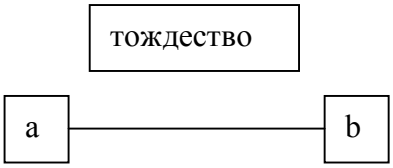
Функциональная диаграмма представляет собой формальный язык, на который транслируется спецификация, написанная на естественном языке. Диаграмме можно сопоставить цифровую логическую сеть. Построение тестов этим методом осуществляется в несколько этапов:
1. Спецификация разбивается на “рабочие” участки. Например, при тестировании компилятора в качестве рабочего участка можно рассматривать каждый отдельный оператор языка программирования.
2. В спецификации определяются причины и следствия. *Причина* – отдельное входное условие или класс эквивалентности входных условий. *Следствие* – выходное условие или преобразование системы (действие, которое входное условие оказывает на состояние программы или системы). Причины и следствия определяются путем последовательного чтения спецификации. Каждой причине и следствию приписывается отдельный номер.
3. Анализируется семантическое содержание спецификации, которая преобразуется в булевый граф, связывающий причины и следствия. Это и

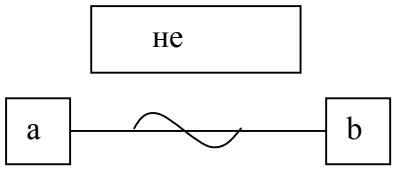
Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

есть функциональная диаграмма.
4. Диаграмма снабжается примечаниями, задающими ограничения и описывающими комбинации причин и (или) следствий, которые являются невозможными из-за синтаксических или внешних ограничений.
5. Путем отслеживания состояний условий диаграммы, она преобразуется в таблицу решений с ограниченными входами. Каждый столбец таблицы решений соответствует тесту.
6. Столбцы таблицы решений преобразуются в тесты.

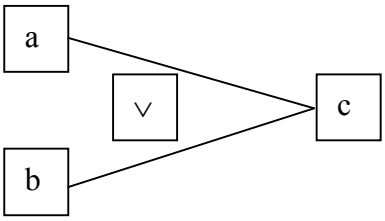
7.2. Базовые символы для записи функциональных диаграмм

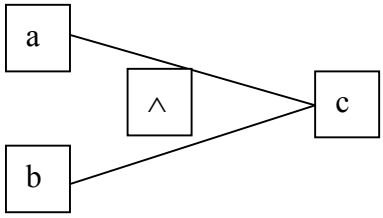
Каждый узел диаграммы может находиться в двух состояниях: 0 или 1; 0 обозначает состояние “отсутствует”, а 1 – “присутствует”.
1. Функция *тождество* устанавливает, что если a=1, то b=1; в противном случае b=0.
2. Функция *не* устанавливает, что если a=1, то b=0; в противном случае b=1.



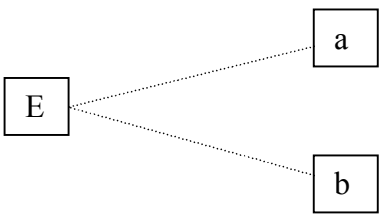


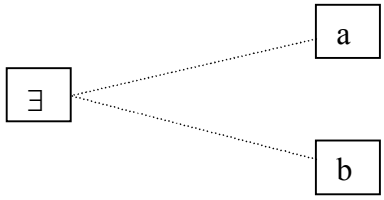
3. Функция *или* устанавливает, что если a=1 или b=1 то c=1; в противном случае c=0.
4. Функция *и* устанавливает, что если и a=1 и b=1 то c=1; в противном случае c=0.
Функции *и*, *или* допускают любое число входов.





В большинстве программ определенные комбинации входных условий невозможны из-за синтаксических или внешних ограничений. В этом случае используются дополнительные входные схемы:
5. Ограничение *исключает* устанавливает, что при E=1, хотя бы одно значение a или b принимает значение 1(a и b не могут быть 1 одновременно):





6. Ограничение *включает* устанавливает, что по крайней мере одна из ве-

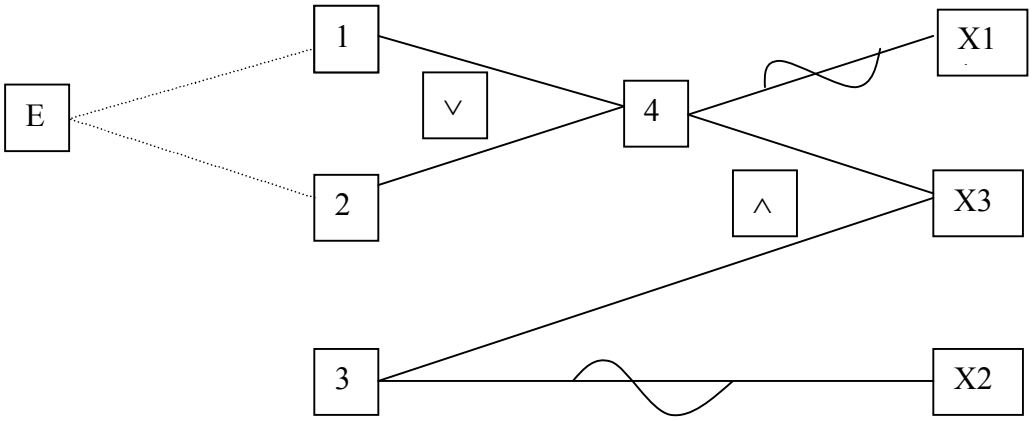
Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

личин a или b всегда должна быть равна 1 (не могут принимать значение 0 одновременно)
7. Ограничение *Одно и только одно* устанавливает, что одна и только одна из величин a или b должна быть равна 1
8. Ограничение R устанавливает, что если a=1, то и b должна быть равна 1
9. Ограничение M устанавливает, что если a=1, то b должна быть равна 0

7.3. Пример построения функциональной диаграммы

Рассмотрим диаграмму, изображающую спецификацию: Первый вводимый символ должен быть либо буквой A либо B, а второй – цифрой. Если первый символ неправильный, выдается сообщение X1, если второй символ неправильный – сообщение X2. В случае, если оба символа правильные, происходит обновление некоторого файла F (сообщение X3).

Функциональная диаграмма имеет вид:



7.4. Преобразование функциональной диаграммы в таблицу решений и построение тестов

Генерация таблицы решений заключается в следующем:
1. Выбрать некоторое следствие, которое должно быть в состоянии 1.
2. Найти все комбинации причин, которые устанавливают это следствие в 1, прокладывая из этого следствия обратную трассу через диаграмму.
3. Построить столбец в таблице решений для каждой комбинации причин.
4. Определить состояние других следствий и поместить их в соответствующий столбец решений. На этом шаге необходимо учитывать следующие положения:

- Если обратная трасса прокладывается через узел *или,* выход которого равен 1, не следует устанавливать в 1 более одного входа в этот узел.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Такое ограничение называется чувствительностью пути. Цель данного правила – избежать пропуска определенных ошибок из-за того, что одна причина маскируется другой.

- Если обратная трасса прокладывается через узел *и,* выход которого равен 0, то все комбинации входов, приводящие выход в 0 должны быть перечислены. Когда исследуется ситуация, где один вход – 0, а один или более других входов 1, не обязательно перечислять все условия, при которых остальные входы могут быть равны 1.
- Если обратная трасса прокладывается через узел *и,* выход которого должен принимать значение 0, необходимо указать лишь одно условие, согласно которому все входы являются 0.

Эти положения позволяют исключить малорезультативные тесты.

Таблица решений, построенная на основании функциональной диаграммы, имеет вид:

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	1	1	1	0
X1	1	0	0	0
X2	0	1	1	0
X3	0	0	0	1

Если состояние причины несущественно, либо состояние причины очевидно вследствии состояний других зависимых причин, то вместо 0 или 1 ставится пробел.

Набор из четырех тестов представлен ниже.

1. 12
2. A1
3. B1
4. AC

В тесты, полученные на основании метода функциональных диаграмм, могут входить и граничные условия.

7.5. Порядок выполнения работы

1. Построить функциональную диаграмму для индивидуального модуля.
2. Провести преобразование функциональной диаграммы в таблицу решений.
3. На основании полученной таблицы решений построить тестовые наборы.
4. Выполнить программу для полученных тестовых наборов. Результаты работы программы проанализировать.
5. Оформить отчет.

7.6. Список контрольных вопросов

1. Как в методе функциональных диаграмм используются особенности программы?

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

2. В чем заключаются недостатки рассматриваемого метода?
3. Что дает функциональное тестирование?
4. В чем отличие метода функциональных диаграмм от метода эквивалентных разбиений?

8. ЛАБОРАТОРНАЯ РАБОТА №7 РАССТАНОВКА КОНТРОЛЬНЫХ ТОЧЕК

Цель работы: изучение методики расстановки контрольных точек с целью восстановления маршрута выполнения программы.

8.1. Теоретические сведения

Любая программа может быть представлена в виде ориентированного графа, вершины которого соответствуют операторам программы, реализующим ветвления, а дуги - передаче управления между операторами или линейным участкам программы.

Остов графа - совокупность вершин и дуг, таких, что каждая вершина должна быть покрыта один раз. Все вершины и только те дуги, которые не вошли в остов, называют коостовом. Дуги коостова называют хордами.

Минимальное число контрольных точек определяется по формуле

$$k = n - m + 1,$$

где n- число дуг, m- число вершин.

Минимальное по мощности множество контрольных точек получается, если контрольные точки размещаются в каждой хорде. При этом каждая хорда разрывается на две хорды и добавляется новая вершина, которая соответствует контрольной точке. При расстановке контрольных точек добавляется новая (фиктивная) дуга, которая замыкает конечную вершину с начальной (она никогда не выполнится). Затем определяется множество базисных циклов. Базисный цикл - это замкнутый контур, содержащий только одну хорду и соответственно одну контрольную точку.

Выбираются тестовые наборы таким образом, чтобы в результате выполнения программы все контрольные точки были пройдены. После выполнения программы строятся векторы (планы) контрольных точек, в которых 1 соответствует тому, что контрольная точка выполнилась, 0 - что не выполнилась. С помощью линейной комбинации базисных циклов восстанавливается маршрут выполнения программ. Базисные циклы выбираются на основании единичных значений вектора контрольных точек.

8.2. Порядок выполнения работы

1. Для индивидуального модуля построить схему алгоритма в виде управляющего ориентированного графа.
2. Выделить остов Т графа и отметить множество хорд коостова Т*.
3. Определить множество базисных циклов, каждый из которых включает

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

ровно одну хорду.
4. Расставить контрольные точки в дугах коостова. Определить, где эти точки должны быть в тексте модуля и вставить в текст операторы вывода координат точки.
5. Подготовить такой набор тестов, чтобы можно было проверить выполнение всех полных маршрутов графа модуля.
6. Выполнить модуль для различных тестов. Построить векторы контрольных точек.
7. По результатам выполнения контрольных точек построить линейную комбинацию базисных циклов, однозначно определяющую выполненный маршрут.
8. Провести анализ полученных результатов.
9. Оформить отчет.

Содержание отчета:

- схема модуля;
- перечень контрольных точек;
- набор входных тестов;
- множество базисных циклов;
- маршруты выполнения модуля;
- векторы контрольных точек;
- выводы.

8.3. Список контрольных вопросов

1. Для чего предназначен метод контрольных точек?
2. Какую информацию содержат результаты выполнения программ?
3. Какой остов целесообразно выбрать из множества остовов программного модуля?
4. Каковы недостатки метода контрольных точек?

9. ЛАБОРАТОРНАЯ РАБОТА № 8 МУТАЦИОННЫЙ АНАЛИЗ

Цель работы - определение способности тестового набора обнаруживать ошибки на основе мутационного анализа, оценка качества тестового набора, определение множества потенциальных ошибок в тестируемой программе, не обнаруживаемых заданным набором тестов.

9.1. Теоретические сведения

Мутационный анализ заключается в определении множества потенциальных ошибок в тестируемой программе, не обнаруживаемых заданным набором тестов, путем внесения в программу поочередно большого числа искусственных ошибок - небольших, синтаксически правильных искажений текста программы, которые называются мутациями. Тест, хорошо обнаруживающий искусственные потенциальные ошибки, будет надежен и для реальных ошибок. Ошибка считается обнаруженной, если

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

результат выполнения некоторым тестом программы с ошибкой отличается от результата выполнения эталонной программы.
Список необнаруженных потенциальных ошибок, выдаваемый в результате мутационного анализа, является непосредственной мерой надежности набора тестов.

Рассматриваемая программа P представляет собой множество N операторов, каждый из которых может содержать потенциальную ошибку. Пусть Pm - программа, содержащая только одну ошибку m из набора ошибок M. Тогда выполнение набора тестов t каждой из M ошибочных программ Pm определит подмножества обнаруженных Dt и необнаруженных Lt ошибок. Если оператор программы содержит хотя бы одну необнаруженную потенциальную ошибку, его называют потенциально ошибочным, в противном случае - надежно проверяемым. Надежность набора тестов t для программы будет определяться как множеством Lt необнаруженных потенциальных ошибок, так и множеством Nt потенциально ошибочных операторов.

Относительное число обнаруженных потенциальных ошибок

$$St = 1 - Lt/M. \quad (1)$$

Относительное число надежно проверенных операторов

$$Qt = 1 - Nt/N. \quad (2)$$

9.2 .Описание мутаций

Исходная программа, подвергнутая мутации, называется мутантом. Мутации - минимальные синтаксически правильные искажения текста программы. В качестве множества M возможных потенциальных ошибок рассматриваются следующие типы мутаций:

- замена имени переменной именем другой переменной;
- изменение константы на 1;
- изменение знака константы;
- добавление (вычитание) 1 к(из) арифметического выражения;
- замена логического условия в условном операторе другим условием или логической константой;
- замена переменной в правой части оператора присваивания;
- замена переменной в логическом или индексном выражении на константу;
- замена переменной в левой части оператора присваивания на неиспользуемую переменную и т.п.

9.3. Порядок выполнение работы

1. Разработать для индивидуального модуля набор тестовых данных, по возможности предусматривающий все возможные ситуации.
2. Для каждого теста (входные данные) определить соответствующие выходные данные, получаемые в результате работы программы. Результаты работы программы запомнить.
3. Вносить в программу по одной искусственной ошибке - мутации (см.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

п.2), и определять, обнаруживает ли ее набор тестов (ошибка считается обнаруженной, если результат работы программы после внесения ошибки отличается от работы исходной программы).
4. Выполнить действия, описанные в п.3, 8-10 раз и определить величины M, N, Lt, Nt (M - количество мутаций; N - количество операторов программы, содержащих мутации; Lt - количество невыявленных мутаций, Nt - количество потенциально ошибочных операторов).
5.Рассчитать St и Qt по формулам (1) и (2).
6. Оформить отчет. В отчете отразить вид ошибки и результат: выявлена ли ошибка тестовым набором или нет.

9.4. Список контрольных вопросов

1. По какому принципу строится множество тестов?
2. С какой целью вводят мутации?
3. Как искусственное введение ошибок влияет на оценку надежности программы?
4. Как определить необходимое число мутаций?

10. ЛАБОРАТОРНАЯ РАБОТА № 9
ОЦЕНКА НАДЕЖНОСТИ ПРОГРАММ

Цель работы: определение количественных показателей надежности индивидуального модуля.

10.1. Теоретические сведения

Надежность программного обеспечения - это вероятность того, что программа какой-то период времени будет работать без сбоев с учетом степени их влияния на выходные результаты. Надежность не является свойством, присущим программе, а в большей мере относится к тому, как используется программа. Надежность является статистической оценкой, связывающей систему с набором требований, в соответствии с которыми она должна разрабатываться. Система считается надежной, если велика вероятность того, что при обращении к ней получим требуемую услугу. Система рассматривается как ненадежная, если она допускает сбои в особых ситуациях, даже если эти ситуации составляют небольшой процент от общего времени использования системы.

Степень приспособленности систем к выполнению поставленной задачи характеризуется величинами, которые называются показателями надежности. Показатели надежности делятся на качественные, порядковые и количественные.

Качественные показатели надежности указывают на то, что система обладает каким-либо свойством для выполнения поставленной задачи. Качественные показатели позволяют отличать системы друг от друга, но не позволяют сравнивать их по надежности.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Порядковые показатели надежности содержат информацию, позволяющую обосновывать предпочтение одного из вариантов системы при их сравнении без количественной оценки степени предпочтения.

Количественные показатели выражают надежность в виде чисел, выраженных в абсолютных или относительных единицах в принятой шкале оценок. Количественные показатели определяются путем непосредственных статистических наблюдений на основе обработки результатов применения или испытания систем.

Для получения оценки надежности программного обеспечения в качестве исходных предпосылок используются следующие утверждения:

1) в результате выполнения программы для каждого множества N_i входных данных получаем однозначный выходной результат;

2) множество N_i исходных данных определяет все вычисления, выполняемые программой;

3) каждая ошибка в программе вызывает сбои для некоторой части входных данных;

4) выполнение программы для некоторого подмножества входных данных представляет собой единичное наблюдение ее действия.

Тогда вероятность P появления сбоя есть вероятность того, что множество входных данных N_i, выработанное для прогона программы таково, что порождает сбои. P может быть выражено через вероятность P_i выбора для работы множества N_i (вероятность появления тестового набора N_i) и переменную y_i:

yi_i= 0, если выходной результат верен для N_i ,

y_i = 1, в противном случае.

Вероятность безотказной работы программного обеспечения будет представлена как:

$$P = 1 - (P_1*y_1 + P_2*y_2+...+ P_n*y_n), \quad (1)$$

где n - число всевозможных входов программного обеспечения (ПО).

Надежность R для некоторого рабочего участка P_i может быть определена как:

$$R = 1 - P = P_1*(1-y_1) + P_2*(1-y_2) + ... + P_n*(1-y_n) \quad (2)$$

Для получения вероятности отказа ПО экспериментальным путем можно применить следующий подход. Если испытывается определенное ПО относительно n различных входов и относительно k из них имеют место отказы, то вероятность отказа ПО оценивается как:

$$P_i = \frac{k}{n} . \quad (3)$$

При равномерном распределении испытуемых входов во входном множестве и при достаточно большом n $P \cong P_i$

В обычном ПО входы во входном множестве вряд ли распределяются однородно, можно говорить только о некотором используемом

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

множестве входов. Рассмотрим входное множество $S_в$ и используемое множество $S_и$. Вероятность отказа в $S_в$ определяется во время программирования, а вероятность отказа в $S_и$ определяется в соответствии с устранением ошибок. Предположим, что $P_в$ и $P_и$ как вероятности отказа $S_в$ и $S_и$ соответственно постоянны. Тогда можно записать:

$$P_и = \frac{k}{n}. \qquad (4)$$

Предполагая, что m причин отказа устранено, можно записать значение вероятности отказа после устранения ошибок для некоторой промежуточной точки:

$$P_и^{j+m} = P_и^j - \frac{m}{n_и} \quad . \qquad (5)$$

При полном устранении всех ошибок ПО становится надежным, т.е.

$$P_и^{j+m} = P_и^j - \frac{m}{n_и} = 0; \quad m = \frac{P_и^j}{n_и} \quad . \qquad (6)$$

В качестве технических требований на разработку надежного ПО рекомендуются следующие критерии:
- корректность программного обеспечения - число серьезных ошибок в программе и время, необходимое для их устранения;
- обслуживаемость системы - степень влияния ошибок ПО на обслуживаемость системы;
- безотказность системы - частота системных отказов, вызываемых ошибками ПО.

10.2. Порядок выполнения работы

1. Выбрать n наборов входных данных Ni, i=1,...,n.
2. Внести в индивидуальный модуль 1-2 мутации.
3. Выполнить программу n раз для каждого входного набора Ni.
4. Подсчитать количество отказов k, вероятность отказа P (считать появление входных наборов равновероятным), yi.
5. Рассчитать вероятность безотказной работы и надежность R.
6. Устранить ошибки и рассчитать количественные характеристики программы без ошибок.
7. Оформить отчет.

10.3. Список контрольных вопросов

1. В чем отличие надежности программного обеспечения от надежности технических устройств?
2. Какие меры могут быть предложены по повышению надежности?
3. Какими условиями определяются размеры тестового набора?
4.Что понимают под отказом программного обеспечения?

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

11. ЛАБОРАТОРНАЯ РАБОТА № 10
РАСЧЕТ МЕТРИК ЧИДАМБЕРА-КЕМЕРЕРА

Цель работы: Оценка объектно-ориентированных программных систем путем расчета проектных метрик, ориентированных на классы.

11.1 Теоретические сведения

При конструировании объектно-ориентированных систем значительная часть затрат приходится на создание визуальных моделей. Решение задачи оценки качества этих моделей требует введения специального метрического аппарата. Метрики Чидамбера и Кемерера ориентированы на классы, которые являются фундаментальными элементами объектно-ориентированных (ОО) систем.

Метрика 1: Взвешенные методы на класс WMC (Weighted Methods Per Class).

В классе *C* определены *n* методов со сложностью c_1, c_2,..., c_n. В этом случае

$$WMC=C_1+C_2+C_3+...+C_n .$$

Количество методов и их сложность являются индикатором затрат на реализацию и тестирование классов. С ростом количества методов в классе его применение становится все более специфическим, тем самым ограничивается возможность многократного использования. По этим причинам метрика WMC должна иметь разумно низкое значение.

Очень часто применяют упрощенную версию метрики. При этом полагают $C_i = 1$. и тогда WMC - количество методов в классе.

При подсчете количества методов будем учитывать только методы текущего класса. Унаследованные методы игнорируются. Обоснование - унаследованные методы уже подсчитаны в тех классах, где они определялись. Наиболее важным источником информации для понимания того, что делает класс, являются его собственные операции. Если класс не может отреагировать на сообщение (например, в нем отсутствует собственный метод), тогда он пошлет сообщение родителю. Вообще, класс, имеющий максимальное количество методов среди классов одного с ним уровня, является наиболее сложным; скорее всего, он специфичен для данного приложения и содержит наибольшее количество ошибок.

Метрика 2: Высота дерева наследования DIT (Depth of Inheritance Tree)

DIT определяется как максимальна длина пути от листа до корн дерева наследования классов. Соответственно, для отдельного класса DIT, это длина максимального пути от данного класса до корневого класса в иерархии классов. По мере роста DIT вероятно, что классы нижнего уровня будут наследовать много методов. Высокая иерархия классов (большое значение DIT) приводит к большей сложности проекта, так как означает привлечение большего количества методов и классов. Вместе с тем, боль-

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

шое значение DIT подразумевает, что многие методы могут использоваться многократно.

Метрика 3: Количество детей NOC (Number of children)

Подклассы, которые непосредственно подчинены суперклассу, называются его детьми. Значение NOC равно количеству детей, то есть количеству непосредственных наследников класса в иерархии классов. С увеличением NOC возрастает многократность использования, гак как наследование - это форма повторного использования. Однако при возрастании NOC ослабляется абстракция родительского класса. Это означает, что в действительности некоторые из детей уже не являются членами родительского класса и могут быть неправильно использованы.

Метрика 4: Сцепление между классами объектов CBO (Coupling between object classes)

CBO - это количество сотрудничеств, предусмотренных для класса, то есть количество классов, с которыми он соединен. Соединение означает, что методы данного класса используют методы или экземплярные переменные другого класса. Другое определение метрики имеет следующий вид: CBO равно количеству сцеплений класса; сцепление образует вызов метода или свойства в другом классе. Данная метрика характеризует статическую составляющую внешних связей классов. С ростом CBO многократность использования класса уменьшается. Очевидно, что чем больше независимость класса, тем легче его повторно использовать в другом приложении. Высокое значение CBO усложняет модификацию и тестирование, которое следует за выполнением модификации. Поэтому *CBO* для каждого класса должно иметь разумно низкое значение.

Метрика 5: Отклик для класса RFC (Response For a Class)

Множество отклика класса RS - это множество методов, которые могут выполняться в ответ на прибытие сообщений в объект этого класса. Формула для определения RS имеет вид:

$RS = \{M\} U_{\text{all } i} \{R_i\}$,

где $\{R_i\text{-}\}$ - множество методов, вызываемых методом i, *{M}* - множество всех методов в классе.

Метрика RFC равна количеству методов во множестве отклика, то есть, равна мощности этого множества:

RFC - card{RS}.

Приведем другое определение метрики: RFC - это количество методов класса плюс количество методов других классов, вызываемых из данного класса.

Метрика RFC является мерой потенциального взаимодействия данного класса с другими классами, позволяет судить о динамике поведения соответствующего объекта в системе. Данная метрика характеризует динамическую составляющую внешних связей классов. С ростом RFC увеличивается сложность класса. Наихудшая величина отклика используется при определении времени тестирования.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Метрика 6: Недостаток связности в методах LOOM (Lack of Cohesion in Methods)

Каждый метод внутри класса обращается к одному или нескольким свойствам (экземплярным переменным). Метрика *LCOM* показывает, насколько методы не связаны друг с другом через свойства (переменные). Если все методы обращаются к одинаковым свойствам, то LCOM - 0.Введем обозначения:

НЕ СВЯЗАНЫ - количество пар методов без общих экземплярных переменных;

СВЯЗАНЫ - количество пар методов с общими экземплярными переменными. Q

Ij - набор экземплярных переменных, используемых методом M_j.

Тогда формула для вычисления недостатка связности в методах примет вид:

LCOM=НЕ СВЯЗАНЫ -СВЯЗАНЫ, если (НЕСВЯЗАНЫ > СВЯЗАНЫ);

LCOM = *0* в противном случае.

Можно определить метрику по-другому: LCOM - это количество пар методов, не связанных по свойствам класса, минус количество пар методов, имеющих такую связь.

Пример 1. Применения метрики LCOM.

В классе имеются методы: M_1, M_2, M_3, M_4. Каждый метод работает со своим набором экземплярных переменных:

I_1-{a, b}; I_2-{a, c}; I_3-{x, y}; I_4-{m, n}. В этом случае

НЕ СВЯЗАНЫ - card (I_{13},I_{14},I_{23},I_{24},I_{34}) - 5;

СВЯЗАНЫ - card (I_{12})-1.

LCOM - 5 - 1 - 4.

Связность методов внутри класса должна быть высокой, так как это содействует инкапсуляции. Если LCOM имеет высокое значение, то методы слабо связаны друг с другом через свойства. Это увеличивает сложность, в связи с чем возрастает вероятность ошибок при проектировании.

Высокие значения LCOM означают, что класс, вероятно, надо спроектировать лучше (разбиением на два или более отдельных класса). Любое вычисление LCOM помогает определить недостатки в проектировании классов, так как эта метрика характеризует качество упаковки данных и методов в оболочку класса.

11.2 .Использование метрик Чидамбера-Кемерера

Пример расчета метрик для структуры представлен в таблице 11.1.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

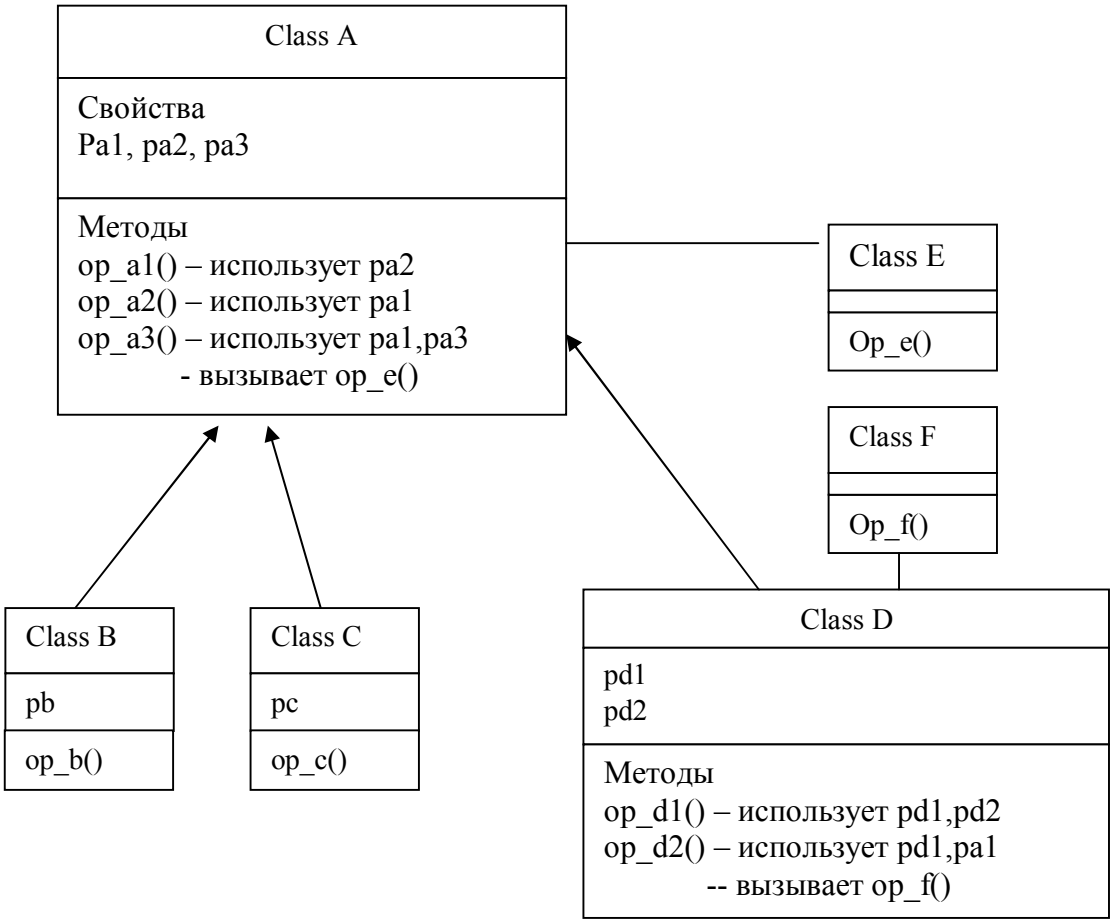


Таблица 11.1 Пример расчета метрик Чидамбера-Кемерера

Имя класса	WMC	DIT	NOC	CBO	RFC	LCOM
Class A	3	0	3	1	4	1
Class B	1	1	0	0	1	0
Class C	1	1	0	0	1	0
Class D	2	1	0	2	3	0

Прокомментируем результаты расчета. Класс Class A имеет три метода (op_al(); op_a2(), op_a3()), трех детей (Class B, Class C, Class D) и является корневым классом. Поэтому метрики WMC, NOC и DIT имеют, соответственно, значения 3, 3 и 0. Метрика CBO для класса Class A равна 1, так как он использует один метод из другого класса (метод op_e() из класса Class E, он вызывается из метода op_a3()). Метрика RFC для класса Class A равна 4, так как в ответ на прибытие в этот класс сообщений возможно выполнение четырех методов (три объявлены в этом классе, а четвертый метод op_e() вызывается из op_a3()). Для вычисления метрики LCOM надо определить количество пар методов класса. Оно рассчитывается по формуле

$$C_m^2 = \frac{m!}{2(m-2)!}, \qquad (1)$$

где m- количество методов класса.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Поскольку в классе три метода, возможны три пары: op_al() и op_a2(), op_al(); op_a3(), op_a2() и op_a3().Общее свойство имеет только третья пара Количество несвязанных пар равно 2, количество связанных равно 1, LCOM=2-1=1.

Для класса Class D метрика CBO равна 2, так как здесь используются свойство pal и метод op_f( ) из других классов. Метрика LCOM в этом классе равна 0, поскольку методы op_dl( ) и op_d2( ) связаны по свойству pdl, а отрицательное значение запрещено.

11.3.Порядок выполнения работы

1. Изучить назначение метрик и пример их расчета.
2. Выбрать текст программы с использованием ООП. Текст должен содержать не менее 3-х классов, связанных между собой через методы или свойства.
3. Построить диаграмму связей и рассчитать метрики .
4. Результаты расчета представить в виде таблицы. провести анализ результатов исследуемой программы.

11.4.Содержание отчета о выполнении работы

- Краткие теоретические сведения;
- Структура классов:
- Результаты расчета;
- Выводы о качестве программы;

11.5.Список контрольных вопросов

1. Охарактеризуйте класс программ, для которых предложенные метрики являются наиболее эффективными.
2. Приведите недостатки предложенных метрик.
3. Какие свойства ОО-программ не учитывают метрики?
4. Что означают высокие значения метрики LCOM?

12. ЛАБОРАТОРНАЯ РАБОТА № 11 ОЦЕНКА КАЧЕСТВА ПРОГРАММ

Цель работы: ознакомление с методами оценки качества программных средств, формирование номенклатуры показателей качества, расчет показателей качества программного средства.

12.1.Теоретические сведения

Испытание – завершающий этап разработки программных средств (ПС). На этом этапе экспериментально определяются характеристики свойств ПС, на основании которых можно сделать заключение о пригод-

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

ности ПС для использования его по назначению. Совокупность свойств ПС, обусловливающих его пригодность удовлетворять определенные потребности в соответствии с назначением, называется качеством программного средства. Показатели качества отражаются в технических заданиях на разработку ПС.

Цель испытаний:

- определение комплексных показателей качества (функциональности, надежности функционирования, удобства использования, рациональности, сопровождаемости, перенесенности), характеризующих несколько свойств ПС;
- определение единичных показателей, значения которых определены в техническом задании или в описании ПС, характеризующих одно из свойств ПС .

Для определения значений показателей качества применяются следующие методы:

1. Измерительный – основан на получении информации свойствах и характеристиках ПС с использованием измерительных, технических и программных средств. (Определение объема памяти, числа строк исходного текста, числа выполненных операторов, количества ветвей в программе).

2. Регистрационный – основан на получении информации при регистрации определенных событий (времени начала и окончания работы, число сбоев и отказов, время передачи управления от модуля к модулю).

3. Расчетный – основан на использовании теоретических и эмпирических зависимостей статистических данных, накапливаемых при испытаниях ПС. Используется для определения показателей точности, надежности, ресурсоемкости.

4. Экспертный – основан на опыте и интуиции группы экспертов-специалистов, компетентных в решении данной задачи. Применяется для определения таких показателей, как удобство применения, анализируемость, адаптируемость, документируемость, структурированность.

Для оценки значений показателей используются следующие типы шкал (множеств возможных значений показателей):

1) метрическая (1.1 – абсолютная, 1.2 – отношения, 1.3 – интервальная);
2) порядковая (ранговая), позволяющая ранжировать характеристики путем сравнения их с опорными значениями (ранжирование – разбивка значений ПК на диапазоны, соответствующие различным степеням удовлетворения требований);
3) номинальная (классификационная), характеризующая наличие или отсутствие рассматриваемого свойства у рассматриваемого ПС.

12.2.Номенклатура показателей качества

В зависимости от количества характеризуемых свойств различают единичные и групповые (комплексные) показатели качества (ПК). Значения единичных показателей подлежат непосредственной оценке, значения

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

групповых ПК определяются расчетным путем.

Номенклатура свойств и показателей качества ПС (НПК) – многоуровневая иерархическая структура. 1-й (верхний) уровень составляют группы ПК, нижний уровень - единичные ПК. Промежуточные уровни - подгруппы ПК. Группы и подгруппы свойств ПК программных средств устанавливаются ДСТУ 2850-94 и приведены в таблице 12.1. Единичные ПК устанавливаются для каждого индивидуального ПС.

Таблица 12.1

Наименование характеризуемых свойств	Определение характеризуемых свойств
1. ФУНКЦИОНАЛЬНОСТЬ	Группа свойств ПС, обусловливающая способность выполнять в заданной среде определенный перечень функций в соответствии с назначением ПС.
1.1.Функциональная полнота	Подгруппа свойств функциональности, характеризующаяся наличием и степенью достаточности основных функций для решения задач в соответствии с значением ПС.
1.2 .Способность к взаимодействию	Подгруппа свойств функциональности, характеризующаяся возможностью его взаимодействия при функционировании с другими ПС или системами.
1.3.Защищенность	Подгруппа свойств функциональности, характеризующаяся его способностью предотвращать несанкционированный доступ к программам и данным
1.4. Массовость	Подгруппа свойств функциональности, характеризующаяся диапазоном и предельными значениями входных данных, которые могут быть преобразованы в правильный результат.
1.5. Согласованность	Подгруппа свойств функциональности, характеризующаяся степенью соблюдения установленных стандартов, соглашений, правил и рекомендаций.
1.6. Производительность	Подгруппа свойств функциональности, характеризующаяся количеством видов работ по обработке данных (в том числе и однотипных), выполняемых в единицу времени функционирования.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Продолжение таблицы 12.1	
2.НАДЕЖНОСТЬ ФУНКЦИОНИРОВАНИЯ	Группа свойств ПС, обусловливающая способность сохранять работоспособность и преобразовывать исходные данные в искомый результат в заданных условиях за установленный период времени
2.1. Безотказность	Подгруппа свойств надежности функционирования, характеризующаяся частотой отказов из-за ошибок или несовершенства ПС. (Отказ – проявление неработоспособности ПС).-
2.2. Устойчивость к аномалиям	Подгруппа свойств надежности функционирования, обусловливающая способность ПС выполнять свои функции в аномальных условиях (аномальные условия – сбои и отказы технических средств, ошибки операторов, ошибки в исходных данных).
2.3. Восстанавливаемость	Подгруппа свойств надежности функционирования, обусловливающая способность к восстановлению работоспособности ПС и данных после отказов.
2.4. Точность	Подгруппа свойств надежности функционирования, характеризующаяся близостью результатов обработки данных к истинным, специфицированным или теоретически верным значениям.
2.6. Реактивность	Подгруппа свойств надежности функционирования, характеризующаяся способностью своевременно преобразовывать входные данные в искомый результат.
3.УДОБСТВО ИСПОЛЬЗОВАНИЯ	Группа свойств ПС, обеспечивающая установленному или предполагаемому кругу пользователей необходимые условия для использования ПС (условия для использования должны использовать психофизические возможности человека)
3.1. Осваиваемость	Подгруппа свойств удобства использования ПС, характеризующаяся условиями, необходимыми для освоения пользователями условий, процедур и правил применения ПС.
3.2. Простота подготовки к работе	Подгруппа свойств удобства использования ПС, обусловливающая легкость подготовки входных данных и запуска в работу ПС.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Продолжение таблицы 12.1	
3.3. Удобство пользовательского интерфейса	Подгруппа свойств удобства использования ПС, обусловливающая необходимые условия пользовательского интерфейса в процессе функционирования
3.4. Анализируемость результатов	Подгруппа свойств удобства использования ПС, обусловливающая легкость понимания результатов функционирования (конечных выходных данных)
3.5. Документированность	Подгруппа свойств удобства использования ПС, характеризующаяся степенью поддержки документацией ПС процессов освоения ПС, подготовки их к работе, взаимодействия пользователя с процессом обработки данных и анализа результатов обработки.
4. РАЦИОНАЛЬНОСТЬ	Группа свойств ПС, характеризующаяся степенью соответствия используемых ресурсов среды функционирования уровню качества функционирования ПС при заданных условиях применения. Ресурсы среды функционирования могут включать другие ПС, технические средства, материалы и услуги персонала. Качество функционирования ПС характеризуется показателями надежности функционирования
4.1. Занятость ресурсов	Подгруппа свойств рациональности ПС, характеризующаяся объемом ресурсов среды функционирования, используемых в процессе применения ПС.
4.2. Использование ресурсов	Подгруппа свойств рациональности ПС, характеризующаяся степенью занятости ресурсов вычислительной системы, используемых при выполнении ПС за определенный период времени
5.СОПРОВОЖДАЕМОСТЬ	Группа свойств ПС, характеризующаяся усилиями, необходимыми для выполнения определенных модификаций. (Модификация – изменение, не являющееся адаптацией к конкретным техническим средствам)

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Продолжение таблицы 12.1	
5.1. Анализируемость	Подгруппа свойств сопровождаемости ПС, обусловливающая их приспособленность к диагностике недостатков и отказов, выявлению частей, которые необходимо изменить и прогнозу последствий этих изменений.
5.2. Исправляемость	Подгруппа свойств сопровождаемости ПС, характеризующаяся усилиями, необходимыми для устранения в ПС выявленных недостатков
5.3. Модернизируемость	Подгруппа свойств сопровождаемости ПС, обусловливающая их приспособленность к модернизации.
5.4. Тестируемость	Подгруппа свойств сопровождаемости ПС, характеризующаяся усилиями, необходимыми для аттестации модифицируемого ПС.
6. ПЕРЕНОСИМОСТЬ	Группа свойств ПС, обусловливающая его приспособленность для переноса из одной среды функционирования в другие.
6.1. Адаптируемость	Подгруппа свойств переносимости ПС, обусловливающая его способность к адаптации к функционированию в средах, отличных от тех, для которых оно разрабатывалось.
6.2. Настраиваемость	Подгруппа свойств переносимости ПС, характеризующаяся усилиями, необходимыми для переноса ПС в одну из заранее предусмотренных сред функционирования.

Формирование НПК для индивидуального ПС заключается в установлении перечня показателей качества и ведется сверху вниз: от групповых до единичных показателей. Свойства, входящие в группу должны взаимно дополнять друг друга. Совокупность свойств должна быть упорядочена в виде дерева свойств. Каждый из ПК должен быть коррелирован с определенным свойством ПС. Для каждого из выбранных показателей следует установить коэффициент весомости.

Выбранная НПК должна удовлетворять следующим критериям: полнота, корреляция с качеством, однозначность определений, верифицируемость, неизбыточность, независимость показателей, прогрессивность.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

12.3.Методы оценки уровня качества ПС

12.3.1.Дифференциальный метод

Основан на определении единичных показателей качества и определении того, достигнут ли уровень базовых значений показателей качества.

Относительное значение отдельных показателей

$$Q_i = \frac{P_i}{P_{ib}} \ , \qquad (1)$$

$$Q_i = \frac{P_{ib}}{P_i}, \ \text{i=1,2,…N} \ , \qquad (2)$$

где P_i - значение i-го показателя качества ,

P_{ib} - базовое значение i-го показателя ,

N - количество показателей качества.

Формула (1) используется в случае соблюдения условия «лучше, если значение показателя больше». В противном случае используется формула (2).

При использовании дифференциального метода уровень качества (УК) считается выше или равным уровню базовых значений, если все Q_i>1. В противном случае УК ниже уровня базовых значений. Если часть Q_i>1, а часть значений Q_i<1, используют комплексный или смешанный методы оценки.

Базовые (эталонные) значения ПК должны соответствовать:

- значениям ПК лучших ПС из числа аналогов;
- нормативным значениям ПК;
- значениям ПК, устанавливаемым в техническом задании на разработку ПС.

12.3.2.Комплексный метод

Основан на применении обобщенного показателя, являющегося функцией от нескольких единичных (групповых) показателей и коэффициентов их весомости.

Установление коэффициентов весомости производят экспертным путем, используя при этом 5-бальную шкалу (5 – крайне важно, чтобы данный показатель имел высокое зн–чение, 4 - важно, чтобы данный показатель имел высокое значение, 3 – хорошо бы иметь высокое значение данного показателя, 2 – при низких значениях показателя ощутимых потерь нет).

Приведенное значение коэффициента весомости

$$M_i = \frac{M_i}{\sum_{i=1}^{N} M_i} \ , \qquad (3)$$

где M_i - значение i-го коэффициента весомости.

Если $\sum_{i=1}^{N} M_i$=1, то коэффициенты весомости называются параметра-

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

ми весомости.

Значения коэффициентов (параметров) весомости определяются при написании технического задания.

Обобщенный показатель качества выражается средним взвешенным показателем U

$$U = \sum_{i=1}^{N} Q_i \cdot M_i , \qquad (4)$$

где Q_i - относительный i-й ПК, вычисляемый по формулам (1), (2).

M_i - параметр весомости i-го показателя, входящего в обобщенный показатель;

N – число показателей, составляющих средний взвешенный показатель.

При трехуровневой иерархической структуре НПК расчет УК производится снизу вверх от единичных показателей до получения обобщенного ПК

$$U_{yk} = \sum_{k=1}^{K} M_k \sum_{ik=1}^{Ik} M_{ik} \sum_{jik=1}^{Jik} Q_{jik} M_{jik} , \qquad (5)$$

где Q_{jik} - относительное значение j-го единичного показателя i-й подгруппы k-й группы, jik=1,2,…,Jik ;

M_{jik} - коэффициент весомости j-го единичного показателя i-й подгруппы k-й группы;

M_{ik} - коэффициент весомости i-й группы k-й подгруппы; i=1,2,…,Ik;

M_k - коэффициент k-й группы показателей; k=1,2,…,K.

M_k

12.3.3.Смешанный метод

Основан на совместном применении единичных и комплексных показателей, причем часть единичных показателей объединяется в группы. После объединения вычисляются значения групповых показателей и некоторых единичных по формулам (1), (2) и (3). Смешанный метод оценки УК применяется в случаях:

1) Совокупность единичных показателей является слишком обширной:

2) Обобщенный ПК в комплексном методе не позволяет сделать выводы о соответствии значений некоторых групповых показателей требуемому уровню.

12.4.Порядок выполнения работы

Провести оценку качества индивидуального программного средства. Процесс оценки качества включает этапы:

- установление цели оценки;
- формирование номенклатуры показателей качества (НПК);
- определение (уточнение) требований к качеству;
- формирование базовых (эталонных) значений показателей качества (ПК);
- выбор методов определения значений ПК и шкал оценок;

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

- определение значений ПК;
- определение уровня качества.

Сформировать 3-х уровневую НПК. Группы и подгруппы свойств показателей качества (1 и 2 уровни) выбрать из таблицы 1. Включить в НПК 2-3 группы показателей. Единичные ПК (3 уровень) установить, исходя из особенностей индивидуального ПС, с учетом назначения и условий использования ПС. Пример единичных показателей - машинонезависимость, доступность, структурированность, завершенность, точность, число строк исходного текста, число выполненных операторов, количество ветвей в программе и т.д..

Построить дерево НПК.

Полученные данные оформить в виде таблицы «Условия оценки уровня качества ПС»:

Таблица 12.2

Индекс ПК	Наименование показателей	Метод определения значений ПК	Знаки лучше (+), хуже (-)	Шкала оценок	Базовые значения показателей	Параметры (коэффициенты) весомости показателей
1	2	3	4	5	6	7

Примечания:

Для обозначения индекса ПК в графе 1 используют 3 десятичные позиции, разделяющиеся точками. Цифра в 1 позиции – номер группы, во 2 – номер подгруппы, в 3 – номер единичного показателя.

1. Для обозначения методов определения значений ПК в графе 3 используются следующие условные знаки: И – измерительный; Рег – регистрационный; Р – расчетный; Э – экспертный.
2. Знак (+) в графе 4 означает, что качество ПС улучшается с увеличением значения этого ПК, знак (-) наоборот.
3. При установлении шкал оценок (в графе 5) используются следующие значения: 1.1 – абсолютная; 1.2 – шкала отношений; 1.3 – интервальная; 2.0 – порядковая; 3.0 – номинальная.
4. В графе 6 для единичных показателей кроме базовых значений в скобках указать выбранные значения единичных показателей.

12.5.Содержание отчета

- Цель работы;
- Номенклатура показателей качества;
- Описание этапов оценки качества ПС;
- Заполненную таблицу «Условия оценки уровня качества ПС»:
- Рассчитанное значение показателя качества программного изделия;
- Выводы.

12.6.Список контрольных вопросов

1. Перечислить методы определения значений единичных показателей

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

качества.

2. Что такое номенклатура показателей качества?
3. Как производится формирование НПК?
4. Каким критериям должна удовлетворять НПК?

13.БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бутаков Е.А. Методы создания качественного программного обеспечения ЭВМ/ Е.А. Бутаков.– М.: Мир, 1984. – 232 с.

2. Введение в тестирование программного обеспечения : Пер.с англ./ Л.Тамре.- М.: Издательский дом «Вильямс», 2003.-368 с.

3. Волховер В.Г. Производственные методы разработки программ/В.Г. Волховер, Л.А. Иванов. - М.: Финансы и статистика, 1983. – 208 с.

4. Кулаков А.Ф. Управление качеством программных средств ЭВМ/ А.Ф. Кулаков. – Киев: Техника, 1989. – 216 с.

5. Кулаков А.Ф. Оценка качества программ ЭВМ/ А.Ф. Кулаков. – Киев.: Техника, 1984. – 167 с.

6. Липаев В.В. Тестирование программ/ В.В. Липаев.– М.: Радио и связь, 1986.–296 с.

7. Лисков Б. Использование абстракций и спецификаций при разработке программ/ Б. Лисков, Дж. Фатэг. – М.: Мир, 1989. – 424 с.

8. Майерс Г. Надежность программного обеспечения/ Г. Майерс. – М.: Мир, 1980. – 360 с.

9. Майерс Г. Искусство тестирования программ/Г.Майерс. – М.:Финансы и статистика, 1982. – 176 с.

10.Буч Г. Объектно-ориентированное проектирование с примерами применения. / Г. Буч . – М.: Конкорд, 1992.-519 с.

11.Орлов С. Технология разработки программного обеспечения: Учебник / С.Орлов . - СПб.: Питер, 2002.-464 с.

12.Холстед М. Начала науки о программах/ М. Холстед.: Финансы и статистика, 1981. – 128 с.

13.Шнейдерман Б. Технология программирования/Б. Шнейдерман.– М.: Радио и связь,1984. – 304 с.

14.Шураков В.В. Надежность программного обеспечения систем обработки данных/ В.В. Шураков. - М.:Финансы и статистика, 1987. – 272 с.

15.Стандартизация языков программирования/Под ред. Е.Л. Ющенко.– Киев.: Техника, 1989. – 160 с.

16.ДСТУ 2844-94. Программные средства ЭВМ. Обеспечение качества. Термины. – Киев, 1994.-12 с.

17.ДСТУ 2850-94. Программные средства ЭВМ. Показатели и методы оценки качества. – Киев,1994. – 20 с.

18.ДСТУ 2853-94. Программные средства ЭВМ. Подготовка и проведение испытаний. – Киев,1994. – 18 с.

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)

Заказ №_____от «_____»_____2006_____.Тираж_____экз.
Изд-во СевНТУ

Create PDF files without this message by purchasing novaPDF printer (http://www.novapdf.com)