Лабораторная работа по курсу «Инструментальные средства и технологии программирования»

Тестирование программы методами черного и белого ящиков»

Цель: проанализировать методы тестирования программ, оценить различные методы с точки зрения детективности и покрывающей способности тестов.

Подготовка к лабораторной работе:

1. Ознакомиться с лекционным материалом по теме "Тестирование программ" учебной дисциплины "Инструментальные средства и технологии программирования".
2. Изучить соответствующие разделы в литературе, приведенной в методических описаниях к лабораторной работе.

Теория:

1. Тестирование по принципу «черного ящика»

Одним из способов проверки программ является стратегия тестирования, называемая стратегией "черного ящика" или тестированием с управлением по данным. В этом случае программа рассматривается как "черный ящик" и такое тестирование имеет целью выяснения обстоятельств, в которых поведение программы не соответствует спецификации.

Для обнаружения всех ошибок в программе необходимо выполнить исчерпывающее тестирование, т.е. тестирование на всех возможных наборах данных. Для тех же программ, где исполнение команды зависит от предшествующих ей событий, необходимо проверить и все возможные последовательности.

Очевидно, что построение исчерпывающего входного теста для большинства случаев невозможно. Поэтому, обычно выполняется "разумное" тестирование, при котором тестирование программы ограничивается прогонами на небольшом подмножестве всех возможных входных данных. Естественно при этом целесообразно выбрать наиболее подходящее подмножество (подмножество с наивысшей вероятностью обнаружения ошибок).

Правильно выбранный тест подмножества должен обладать следующими свойствами:

1) уменьшать, причем более чем на единицу число других тестов, которые должны быть разработаны для достижения заранее определенной цели «приемлемого» тестирования:

2) покрывать значительную часть других возможных тестов, что в некоторой степени свидетельствует о наличии или отсутствии ошибок до и после применения этого ограниченного множества значений входных данных.

Стратегия "черного ящика" включает в себя следующие методы формирования тестовых наборов:

- эквивалентное разбиение;
- анализ граничных значений;
- анализ причинно-следственных связей;
- предположение об ошибке.

Эквивалентное разбиение

Основу метода составляют два положения:

1. Исходные данные программы необходимо разбить 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 тест какого-либо класса обнаруживает ошибку, то предполагается, что все другие тесты этого класса эквивалентности тоже обнаружат эту ошибку и наоборот
2. Каждый тест должен включать по возможности максимальное количество различных входных условий, что позволяет минимизировать общее число необходимых тестов.

Первое положение используется для разработки набора "интересных" условий, которые должны быть протестированы, а второе - для разработки минимального набора тестов.

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа:

- выделение классов эквивалентности;
- построение тестов.

1.1.2.Выделение классов эквивалентности

Классы эквивалентности выделяются путем выбора каждого входного условия (обычно это предложение или фраза из спецификации) и разбиением его на две или более групп. Для этого используется таблица следующего вида:

Входное условие

Правильные классы эквивалентности

Неправильные классы эквивалентности

Правильные классы включают правильные данные, неправильные классы - неправильные данные.

Выделение классов эквивалентности является эвристическим процессом, однако при этом существует ряд правил:

- Если входные условия описывают область значений (например «целое данное может принимать значения от 1 до 999»), то выделяют один правильный класс 1<=X<=999 и два неправильных X<1 и X>999.
- Если входное условие описывает число значений (например, «в автомобиле могут ехать от одного до шести человек»), то определяется один правильный класс эквивалентности и два неправильных (ни одного и более шести человек).
- Если входное условие описывает множество входных значений и есть основания полагать, что каждое значение программист трактует особо (например, «известные способы передвижения на АВТОБУСЕ, ГРУЗОВИКЕ,

ТАКСИ, МОТОЦИКЛЕ или ПЕШКОМ»), то определяется правильный класс эквивалентности для каждого значения и один неправильный класс (например «на ПРИЦЕПЕ»).

- 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).
- Если есть любое основание считать, что различные элементы класса эквивалентности трактуются программой неодинаково, то данный класс разбивается на меньшие классы эквивалентности.

1.1.3.Построение тестов

Этот шаг заключается в использовании классов эквивалентности для построения тестов. Этот процесс включает в себя:

- Назначение каждому классу эквивалентности уникального номера.
- Проектирование новых тестов, каждый из которых покрывает как можно большее число непокрытых классов эквивалентности, до тех пор, пока все правильные классы не будут покрыты (только не общими) тестами.
- Запись тестов, каждый из которых покрывает один и только один из непокрытых неправильных классов эквивалентности, до тех пор, пока все неправильные классы не будут покрыты тестами.

Разработка индивидуальных тестов для неправильных классов эквивалентности обусловлено тем, что определенные проверки с ошибочными входами скрывают или заменяют другие проверки с ошибочными входами.

Недостатком метода эквивалентных разбиения в том, что он не исследует комбинации входных условий.

2. Анализ граничных значений.

Граничные условия - это ситуации, возникающие на, выше или ниже границ входных классов эквивалентности. Анализ граничных значений отличается от эквивалентного разбиения следующим:

- Выбор любого элемента в классе эквивалентности в качестве представительного при анализе граничных условий осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.
- При разработке тестов рассматриваются не только входные условия (пространство входов), но и пространство результатов.

Применение метода анализа граничных условий требует определенной степени творчества и специализации в рассматриваемой проблеме. Тем не менее, существует несколько общих правил этого метода:

- Построить тесты для границ области и тесты с неправильными входными данными для ситуаций незначительного выхода за границы области, если входное условие описывает область значений (например, для области входных значений от -1.0 до +1.0 необходимо написать тесты для ситуаций -1.0, +1.0, -1.001 и +1.001).

- Построить тесты для минимального и максимального значений условий и тесты, большие и меньшие этих двух значений, если входное условие удовлетворяет дискретному ряду значений. Например, если входной файл может содержать от 1 до 255 записей, то проверить 0, 1, 255 и 256 записей.
- Использовать правило 1 для каждого выходного условия. Причем, важно проверить границы пространства результатов, поскольку не всегда границы входных областей представляют такой же набор условий, как и границы выходных областей. Не всегда также можно получить результат вне выходной области, но, тем не менее, стоит рассмотреть эту возможность.
- Использовать правило 2 для каждого выходного условия.
- Если вход или выход программы есть упорядоченное множество (например, последовательный файл, линейный список, таблица), то сосредоточить внимание на первом и последнем элементах этого множества.
- Попробовать свои силы в поиске других граничных условий.

Анализ граничных условий, если он применен правильно, является одним из наиболее полезных методов проектирования тестов. Однако следует помнить, что граничные условия могут быть едва уловимы и определение их связано с большими трудностями, что является недостатком этого метода. Второй недостаток связан с тем, что метод анализа граничных условий не позволяет проверять различные сочетания исходных данных.

3. Анализ причинно-следственных связей.

Метод анализа причинно-следственных связей помогает системно выбирать высокорезультативные тесты. Он дает полезный побочный эффект, позволяя обнаруживать неполноту и неоднозначность исходных спецификаций.

Для использования метода необходимо понимание булевской логики (логических операторов - и, или, не). Построение тестов осуществляется в несколько этапов.

1) Спецификация разбивается на «рабочие » участки, так как таблицы причинно-следственных связей становятся громоздкими при применении метода к большим спецификациям. Например, при тестировании компилятора в качестве рабочего участка можно рассматривать отдельный оператор языка.

2) В спецификации определяются множество причин и множество следствий. Причина есть отдельное входное условие или класс эквивалентности входных условий. Следствие есть выходное условие или преобразование системы. Каждым причине и следствию приписывается отдельный номер.

3) На основе анализа семантического (смыслового) содержания спецификации строится таблица истинности, в которой последовательно перебираются все возможные комбинации причин и определяются следствия каждой комбинации причин. Таблица снабжается примечаниями, задающими ограничения и описывающими комбинации причин и/или следствий, которые являются невозможными из-за синтаксических или внешних ограничений.

Аналогично, при необходимости строится таблица истинности для класса эквивалентности.

Примечание. При этом можно использовать следующие приемы:

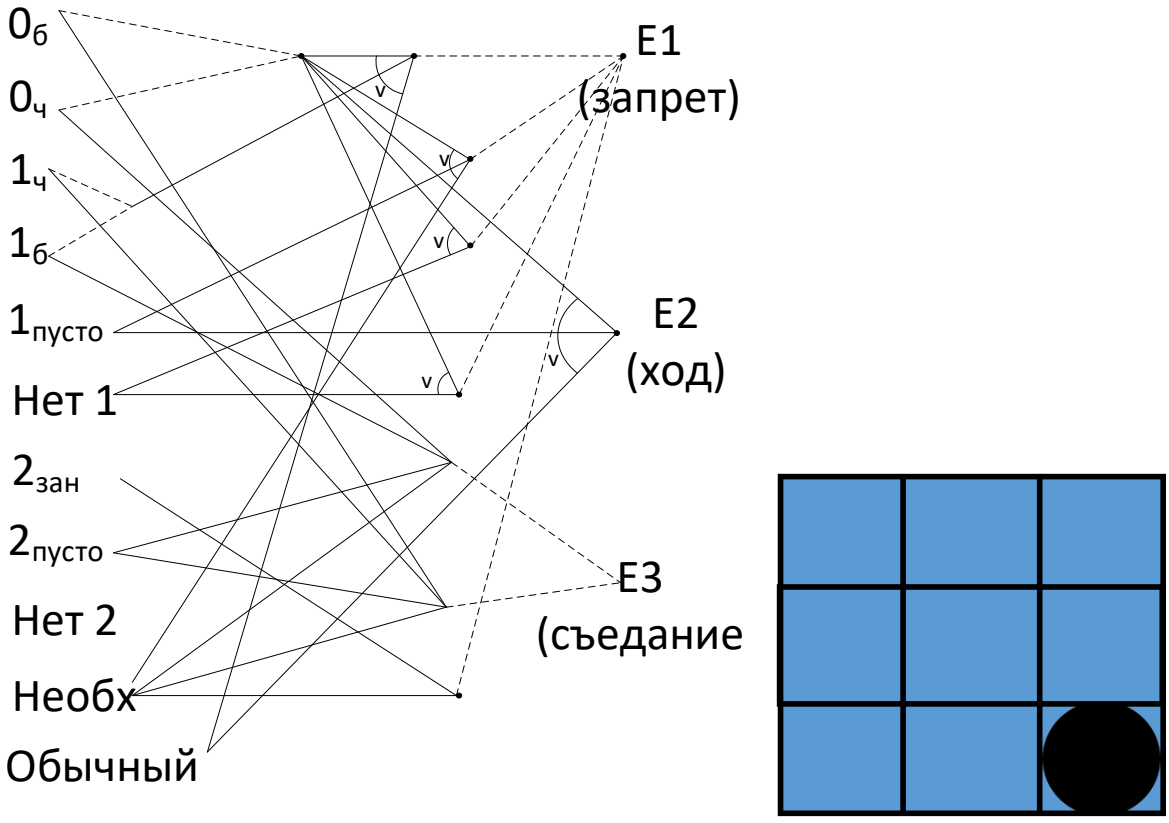
- По возможности выделять независимые группы причинно-следственных связей в отдельные таблицы.
- Истина обозначается "1". Ложь обозначается "0". Для обозначения безразличных состояний условий применять обозначение "X", которое предполагает произвольное значение условия (0 или 1).

4) Каждая строка таблицы истинности преобразуется в тест. При этом:

- по возможности следует совмещать тесты из независимых таблиц;
- для классов эквивалентности входных условий дополнительно необходимо

Недостаток метода - неадекватно исследует граничные условия.

Для анализа причинно-следственных связей также может быть использовано И-ИЛИ дерево. Пример И-ИЛИ дерева для отслеживания причинно-следственных связей при определении хода в игре шашки (дуга на пересечении условий обозначает «ИЛИ»).



4. Предположение об ошибке.

Часто программист с большим опытом выискивает ошибки "без всяких методов". При этом он подсознательно использует метод "предположение об ошибке". Процедура метода предположения об ошибке в значительной степени основана на интуиции. Основная идея метода состоит в том, чтобы перечислить в некотором списке возможные ошибки или ситуации, в которых они могут появиться, а затем на основе этого списка составить

тесты. Другими словами, требуется перечислить те специальные случаи, которые могут быть не учтены при проектировании.

Пример применения методов тестирования «черным ящиком»

Пусть необходимо выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. Попутно, она должна определять параллельность прямой одной их осей координат.

В основе программы лежит решение системы линейных уравнений:

$Ax + By = C$ и $Dx + Ey = F$.

2.1. Используя метод эквивалентных разбиений, получаем для всех коэффициентов один правильный класс эквивалентности (коэффициент - вещественное число) и один неправильный (коэффициент - не вещественное число). Откуда можно предложить 7 тестов:

1) все коэффициенты - вещественные числа;

2)- 7) поочередно каждый из коэффициентов - не вещественное число.

2.2. По методу граничных условий:

можно считать, что для исходных данных граничные условия отсутствуют (коэффициенты - "любые" вещественные числа);

для результатов - получаем, что возможны варианты: единственное решение, прямые сливаются (множество решений), прямые параллельны (отсутствие решений). Следовательно, можно предложить тесты, с результатами внутри области:

1. результат - единственное решение ($\delta \neq 0$);
2. результат - множество решений ($\delta = 0$ и $\delta_x=\delta_y=0$);
3. результат - отсутствие решений ($\delta = 0$, но $\delta_x\neq0$ или $\delta_y\neq0$);

и с результатами на границе:

1. $\delta = 0{,}01$;
2. $\delta = -0{,}01$;
3. $\delta = 0$, $\delta_x = 0{,}01$, $\delta_y = 0$;
4. $\delta = 0$, $\delta_y = -0{,}01$, $\delta_x = 0$.

2.3. По методу анализа причинно-следственных связей:

Определяем множество условий.

а) для определения типа прямой:

$a = 0$
$b = 0$
$c = 0$ - для определения типа и существования первой прямой;
$d = 0$
$e = 0$
$f = 0$ - для определения типа и существования второй прямой;

б) для определения точки пересечения:

$\delta = 0$
$\delta_x = 0$
$\delta_y = 0$

Выделяем три группы причинно-следственных связей (определение типа и существования первой линии, определение типа и существования второй линии, определение точки пересечения) и строим таблицы истинности.

A=0	B=0	C=0	Результат
0	0	X	прямая общего положения
0	1	0	прямая, параллельная оси OX
0	1	1	ось OX
1	0	0	прямая, параллельная оси OУ
1	0	1	ось OУ
1	1	X	множество точек плоскости

Такая же таблица строится для второй прямой.

$\delta = 0$	$\delta_x = 0$	$\delta_y = 0$	Ед. реш.	Мн.реш.	Реш. нет
0	X	X	1	0	0
1	0	X	0	0	1
1	X	0	0	0	1
1	1	1	0	1	0

Каждая строка этих таблиц преобразуется в тест. При возможности (с учетом независимости групп) берутся данные, соответствующие строкам сразу двух или всех трех таблиц.

В результате к уже имеющимся тестам добавляются:

1. проверки всех случаев расположения обеих прямых - 6 тестов по первой прямой вкладываются в 6 тестов по второй прямой так, чтобы варианты не совпадали, - 6 тестов;
2. выполняется отдельная проверка несовпадения условия $\delta x = 0$ или $\delta y = 0$ (в зависимости от того, какой тест был выбран по методу граничных условий) - тест также можно совместить с предыдущими 6 тестами;

2.4. По методу предположения об ошибке добавим тест:

все коэффициенты - нули.

Всего получили 20 тестов по всем четырем методикам. Если еще попробовать вложить независимые проверки, то возможно число тестов можно еще сократить. (Не забудьте для каждого теста заранее указывать результат!).

3.Общая стратегия тестирования.

Все методологии проектирования тестов могут быть объединены в общую стратегию. Это оправдано тем, что каждый метод обеспечивает создание определенного набора тестов, но ни один из них сам по себе не может дать полный набор тестов. Приемлемая стратегия состоит в следующем:

1. Если спецификация состоит из комбинации входных условий, то начать рекомендуется с применения метода функциональных диаграмм.
2. В любом случае необходимо использовать анализ граничных значений.
3. Определить правильные и неправильные классы эквивалентности для входных и выходных данных и дополнить, если это необходимо, тесты, построенные на предыдущих шагах.
4. Для получения дополнительных тестов рекомендуется использовать метод предположения об ошибке.

Порядок выполнения работы:

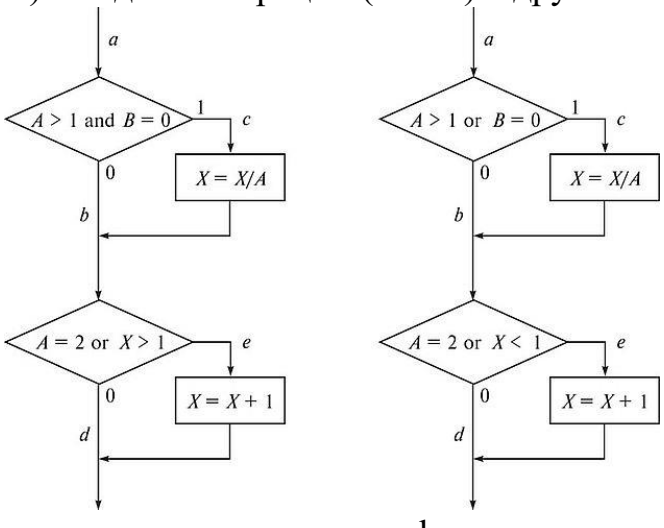
1. Ознакомиться с теоретическими сведениями по стратегиям тестирования.
2. В соответствии с вариантом задачи, подготовить тесты по методикам стратегии"черного ящика".
3. Предлагаемые тесты свести в таблицу.

Номер теста Назначение теста Значения исходных данных Ожидаемый результат Реакция программы Вывод

4. Разработать программу.
5. Выполнить тестирование. Занести в таблицу результаты.

Тестирование методом «белого ящика»

Тестирование методом «белого ящика» применяется к той программе, для которой тестировщику известна её структура. Ее базой служит потоковый граф программы, в котором каждый узел ассоциируется с оператором или программным блоком, а дуги – с передачей управления (или данных) от одной операции (блока) к другой.



a b

Метод «белого ящика» можно разделить на методы покрытия решений (покрытия переходов), при котором набор тестов должен покрывать все переходы; а также метод покрытия условий. В соответствии с методом покрытия условий записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.

Метод покрытия решений

Согласно методу покрытия решений каждое направление перехода должно быть реализовано, по крайней мере, один раз. Этот метод включает в себя критерий покрытия операторов, так как при выполнении всех направлений переходов выполнятся все операторы, находящиеся на этих направлениях.

Для программы, приведенной на рис. Л5.1, покрытие решений может быть выполнено двумя тестами, покрывающими пути {ace, abc}, либо {acc, abe}. Для этого выберем следующие исходные данные; {A = 3, B= O, X=3} — в первом случае и {A = 2, B= 1, X= 1} — во втором. Однако путь, где X не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо условия X> 1 записано X<1, то ошибка не будет обнаружена двумя тестами.

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
A = 3, *B*= 0, *X=2*	*X* = 1	*X* = 1	Неуспешно
A = 2, B = 1, *X*= 1	*X=2*	X=2	Успешно

Методы покрытия условий

В рассматриваемом примере имеем четыре условия: {A> 1, 5=0}, {A = 2, X> 1}. Следовательно, требуется достаточное число тестов, такое, чтобы реализовать ситуации, где A > 1, A < 1, 5 = 0 и X#0 в точке a и X = 2, A =2, X> и T< 1 в точке b. Тесты, удовлетворяющие критерию покрытия условий и соответствующие им пути:

а) A = 2, B = 0, X=4 ace;

б) A = 1, B = 1, X=0 abc!.

Тест	Ожидаемый	Фактический	Результат
	результат	результат	тестирования
A = 2, *B*= 0, *X= 4*	*X= 4*	*X= 3*	Неуспешно
A= 1, *B*= 1, *X=0*	*X= 1*	*X*= 1	Успешно

Порядок выполнения работы:

1. Ознакомиться с теоретическими сведениями по стратегиям тестирования.
2. Создать потоковый граф программы
3. Определить цикломатическое число программы

4. Составить тесты, максимально покрывающих программу (оформляется в виде таблицы)

5. Выполнить тестирование. Занести в таблицу результаты.

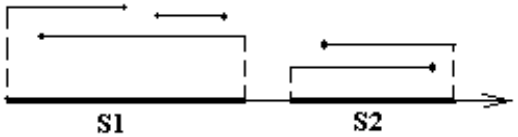
Варианты заданий:

Вариант 1.

Разработать программу решения уравнения , где a, b, c - любые вещественные числа.

Вариант 2.

Разработать программу определения суммарной длины тени, которую отбрасывают на ось OX отрезки, параллельные этой оси и заданные координатами x начала и конца отрезка:



S=S1+S2

Вариант 3.

Разработать программу исследования уравнений второго порядка с двумя неизвестными $Ax^2+2Bxy+Cy^2+2Dx+2Ey+F=0$. Программа должна определять вид графика: эллипс, парабола, гипербола, две пересекающиеся прямые, две параллельные прямые, две мнимые прямые.

Примечание. Вид прямой второго порядка определяется по двум дискриминантам

большому: $\Delta = \begin{vmatrix} A & B & D \\ B & C & E \\ D & E & F \end{vmatrix}$ и малому $\delta = \begin{vmatrix} A & B \\ B & C \end{vmatrix}$.

Малый дискиминант для эллипса положителен, для гиперболы отрицателен, для параболы равен нулю. Если большой дискриминант равен нулю, то линия второго порядка распадается на две прямых:

для эллиптического вида - пересекающиеся мнимые прямые (точка), для гиперболического вида - пара пересекающихся действительных прямых, для параболического вида - две параллельные прямые.

Вариант 4.

Разработать программу определения вида треугольника, заданного длинами его сторон: равносторонний, равнобедренный, прямоугольный, разносторонний.

Вариант 5.

Разработать программу определения вида четырехугольника, заданного координатами вершин на плоскости: квадрат, прямоугольник, параллелограмм, ромб, равнобедренная трапеция, прямоугольная трапеция, трапеция общего вида, четырехугольник общего вида.

Вариант 6.

Разработать программу, определяющую взаимное расположение прямых в пространстве: параллельны, пересекаются, скрещиваются и отдельно, расположение каждой прямой (параллельна оси, перпендикулярна плоскости или общего расположения). Прямые задаются координатами двух точек.

Примечание. Две прямые лежат в одной плоскости, если , прямые параллельны если , $\begin{vmatrix} x_1^2 - x_1^1 & y_1^2 - y_1^1 & z_1^2 - z_1^1 \\ l^1 & m^1 & n^1 \\ l^2 & m^2 & n^2 \end{vmatrix} = 0$, прямые параллельны если $\frac{l^1}{l^2} = \frac{m^1}{m^2} = \frac{n^1}{n^2}$, где $l=x_2-x_1$, $m=y_2-y_1$, $n=z_2-z_1$ (верхний индекс соответствует номеру прямой).

Вариант 6.

Ход в шашках

Вариант 7-12

Ход в шахматах ладьёй, ферзем, королем, слоном, конем, пешкой

Вариант 13

Карточная игра «21»

Вариант 14

Ход в игре «Домино» (имеется числа в начале и конце ряда фишек и требуется выяснить, можно ли произвести ход, желательно сделать ход так, чтобы числа на обоих концах ряда были одинаковые, чтобы осложнить следующий ход противнику)

Вариант 15

Пересечение параболы и прямой

Вариант 16

Пересечение гиперболы ($xy=a^2/2$) и прямой

Вариант 17

Пересечение гиперболы ($xy=a^2/2$) и функции $a*|x|$, где a – константа (модуль x)

Вариант 18

Разработать программу решения уравнения , где a, b, c - любые вещественные числа.

Вариант 19

Пересечение двух парабол, заданных формулами $y_1=(x-a_1)^2+b1$ $y_2=(x-a_2)^2+b_2$.

Вариант 20

Пересечение круга и прямой (расстояние от точки до прямой вычисляется по формуле $\text{distance}(ax+by+c=0,(x_0,y_0)) = \frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$, координаты ближайшей к заданной точке точки на прямой, вычисляются по формуле: $x = \frac{b(bx_0-ay_0)-ac}{a^2+b^2}$, $y = \frac{a(-bx_0+ay_0)-bc}{a^2+b^2}$)

Вариант 21

Пересечение двух кругов (входные параметры: координаты центров и радиусы кругов)

Вариант 22

Пересечение графиков $y = \sqrt{(x - a} + b$ и прямой

Вариант 23

Пересечение эллипса ($1 = \frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2}$), где x_0,y_0 – центр эллипса, с осью X

Вариант 24

Пересечение эллипса ($1 = \frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2}$) с осью Y

Вариант 25

Пересечение шара и плоскости (кратчайшее расстояние между точкой и плоскостью вычисляется по формуле: Ax + By + Cz + D = 0, M(Mx, My, Mz)

$d = \frac{|A \cdot M_x + B \cdot M_y + C \cdot M_Z + D|}{\sqrt{A^2 + B^2 + C^2}}$)

Вариант 26

Пересечение прямой с функцией $a*|x|$, где a,b – константы (модуль x).

Вариант 27

Пересечение параболы с прямой, параллельной оси OX.

Вариант 28

Пересечение параболы с функцией |x| (модуль x).

Вариант 29

Пересечение прямой с функцией |x| (модуль x).

Вариант 30

Пересечение круга с функцией |x| (модуль x).

Вариант31

Пересечение прямой с функцией |x| (модуль x) и функции |x-a|+b, где a,b - константы.

Защита отчета по лабораторной работе

Отчет по лабораторной работе должен включать в себя:

1. Задание.
2. Алгоритм программы.
3. Таблицу с результатами тестирования.

Литература

1. Майерс Г. Искусство тестирования программ / Пер. с англ. под ред. Б. А. Позина. - М.: Финансы и статистика,1982.-176с.

2. Иванова Г.С. Технология программирования. – М.: Из-во МГТУ им. Баумана 2002, - 241 с.