Дисциплина: Разработка спецификации требований к ПО

Лабораторная работа №1 (2 часа)

Тема: Возможности Visual Basic для генерации программного кода.

Цель работы: **Изучить и освоить навыки генерации программного кода с помощью Visual Basic.**

1. Теоретические сведения

1.1. *Формальный метод* в разработке программы – это математически обоснованная методика описания свойства программы. Альтернативой ему является неформальный метод, основанный, как правило, на опыте и интуиции конкретного программиста. Используя формальный метод, разработчик программы может специфицировать, спроектировать, написать и проверить программу некоторым систематическим образом в противовес использованию неформального метода, чреватому ошибками, двусмысленностями и недомолвками.

Формальная спецификация — это системная спецификация, записанная на языке, словарь, синтаксис и семантика которого определены формально.

Формальный метод обычно реализуется в виде некоторого *языка спецификаций*. К настоящему времени разработан ряд *функциональных* и *императивных* языков спецификаций. Функциональные языки обычно основываются на теории алгебр, а императивные языки – на методах преобразования алгебр, представляющих различные состояния описываемой динамической системы (программы). Необходимость формального определения языка предполагает, что этот язык основывается на математических концепциях.

Как правило, в самом начале разработки программы, создаётся *Видение программы (vision statement)* или *Концепция программы*. Это краткое описание результатов, которые планируется достичь в ходе программы. Концепция, в свою очередь, не просто описывает выгоды от реализации программы и свойства продукта. Она содержит в себе то, как должна измениться система в сравнении с нынешним состоянием. И вот эта «будущая система» и должна реализовать выгоды от программы.

Концепция обычно может быть реализована многими программами, но только небольшое количество из них имеет практическое значение. Эта ситуация изображена на рис. 1.

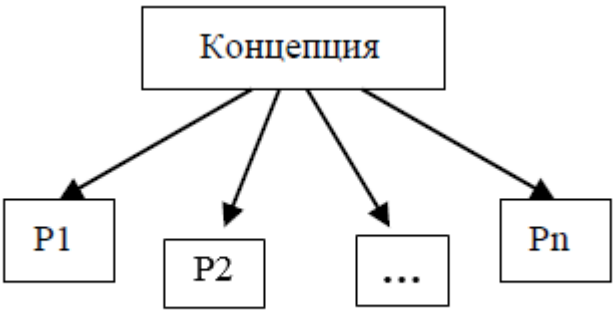


Рис. 1. Концепция и множество реализующих ее программ

В текущей практике концепция обычно формулируется неформально (на естественном языке) и поэтому независимо от используемого метода проверки правильности реализующей ее программы (обычно используется тестирование) результат применения этого метода может формулироваться только в неформальных терминах.

При формальных методах между концепцией и программой вводится промежуточное звено – спецификация. Цель этого звена –обеспечить математическое описание концепции и позволить установить правильность программы путем доказательства эквивалентности программы этой спецификации. Эта ситуация изображена на рис. 2.

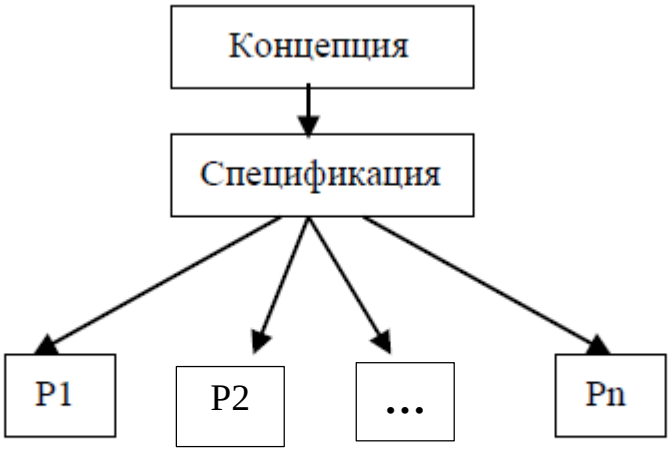


Рис. 2. Концепция, ее формальная спецификация и множество реализующих ее программ

Формальные спецификации играют также важную роль во время построения программы. Широко признано, что спецификацию того, что должна делать программа, следует иметь до того как программа кодируется, поскольку спецификация помогает лучше понять концепцию и повышает вероятность того, что программа действительно будет реализовывать именно ее. Далее формальная спецификация исключительно важна во время использования и модификации программы.

Таким образом, использование формальной спецификации вовлекает в рассмотрение несколько прагматических аспектов: кто использует ее, для чего она используется, когда она используется и как она используется.

Категории пользователей спецификаций

Основными пользователями формальной спецификации являются:

1) спецификатор – лицо, составляющее спецификацию;

2) реализатор – лицо, пишущее программу (программист);

3) верификатор – лицо, доказывающее соответствие программы спецификации;

4) клиент – лицо, использующее программу.

Взаимодействие этих лиц (как один из вариантов) изображено на Рис.3.

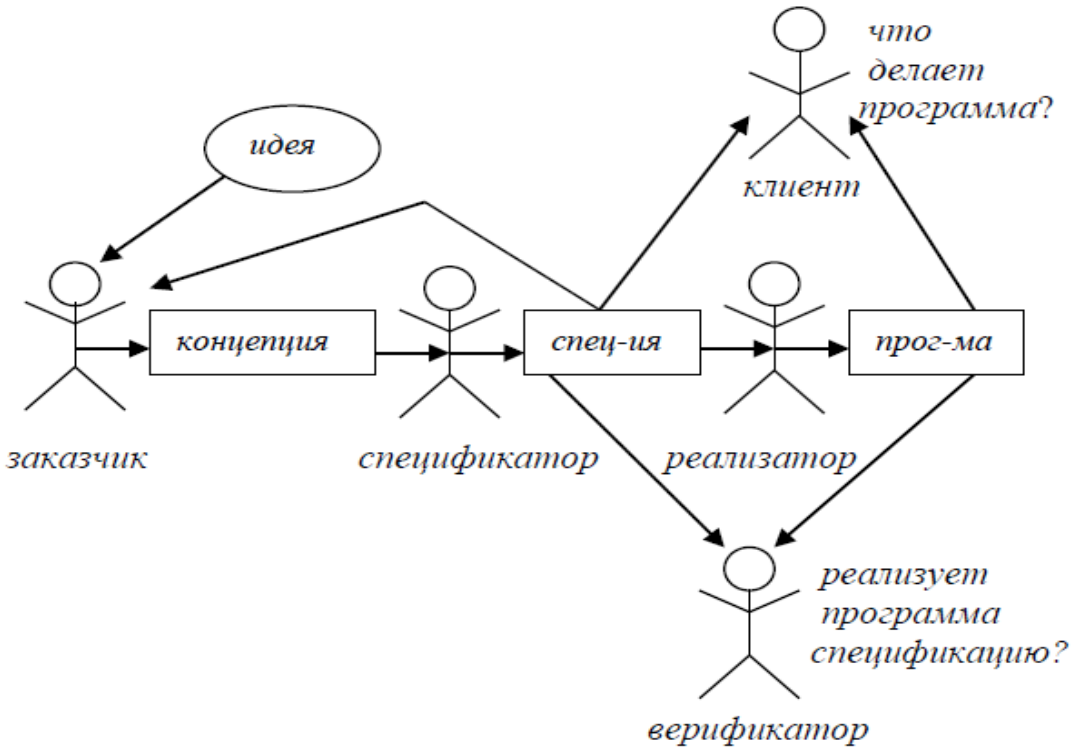


Рис. 3. Взаимодействие пользователей спецификации

Назначение спецификации

1. Описание концепции точным и недвусмысленным образом. Составляя формальное описание концепции, спецификатор вынужден анализировать ее до мельчайших деталей, которые обычно игнорируются при неформальном описании.

2. Привлечение инструментов для выявления двусмысленностей и противоречий в описании концепции. Неформальное описание не может быть

формально проанализировано, и, как следствие, программа зачастую реализует концепцию неправильно. Формальная спецификация служит средством общения между заказчиком и реализатором программы для того, чтобы быть уверенным, что реализатор правильно понимает желания заказчика. Формальная спецификация служит также средством общения между несколькими реализаторами, если программа реализуется в виде нескольких модулей, кодируемых разными лицами.

3. Достижение однозначного понимания программы ее реализатором и пользователями. Таким образом устраняются споры между пользователем и реализатором по поводу правильности исполнения программой ее функций.

4. Определение правильности реализации программы и эквивалентности различных реализаций. При этом совсем не обязательно привлечение аналитических методов проверки. Даже если проверка правильности программы осуществляется методом тестирования, возможно применение методик, основывающихся на сравнении формальной спецификации и реализации. Результатом такой методики может быть множество критических тестовых вариантов, которые устанавливают, что программа правильно реализует данную спецификацию. Формальность спецификации означает возможность привлечения вычислительной машины, например, для проверки шагов доказательства правильности программы или для генерирования тестовых вариантов.

5. Подготовка документации программы, необходимой для эксплуатации и модификации программы. В этом случае формальная спецификация способствует пониманию текста программы, написанного другим лицом. При отсутствии формальной спецификации единственный путь программиста, модифицирующего программу, – сравнить текст программы со своим интуитивным представлением о том, что она должна делать. Однако интуиция часто оказывается ненадежным помощником. При наличии формальной спецификации чтение текста программы приобретает вид неформального доказательства, каждый шаг которого основывается на понимании формального описания.

6. Обеспечение средства общения между клиентом, реализатором и спецификатором. Спецификация, полученная в процессе проектирования программы, служит для передачи намерения заказчика программы реализатору и готовой программы – пользователю.

2. Основы программирования на VBA

Использование кода для выполнения операций приложениями

Может показаться, что написание кода — сложный или загадочный процесс, но его базовые принципы основаны на применении повседневной логики и

вполне доступны. Приложения Office 2010 созданы так, чтобы предоставлять сущности, называемые *объектами*, которые могут принимать инструкции. Пользователь может взаимодействовать с приложениями, отправляя инструкции различным объектам приложения. Эти объекты являются многочисленными, разнообразными и гибкими, но у них есть свои ограничения. Они могут делать только то, для чего были разработаны, и выполняют только написанные для них инструкции.

Объекты

Программируемые объекты связаны друг с другом в иерархию, называемую *объектной моделью* приложения. Грубо говоря, объектная модель отражает то, что показывается в интерфейсе пользователя, например, объектная модель Excel содержит, среди многих других, объекты **Application**, **Workbook**, **Sheet** и **Chart**. Объектная модель является общей картой приложения и его возможностей.

Свойства и методы

Управлять объектами можно, задавая их *свойства* и вызывая их *методы*. Задание свойства изменяет некоторое качество объекта. Вызов метода заставляет объект выполнить некоторое действие. Например, у объекта **Workbook** есть метод **Close**, закрывающий книгу, и свойство **ActiveSheet**, представляющее лист, активный в данный момент в книге.

Коллекции

Многие объекты поставляются в версиях единственного и множественного числа — Workbook и Workbooks, Worksheet и Worksheets и т. д. Версии множественного числа называются *коллекциями*. Объекты коллекции используются для выполнения действия над несколькими объектами коллекции. Позднее в данной статье рассматривается, как использовать коллекцию **Worksheets** для изменения имени каждого листа книги.

3. Макросы и редактор Visual Basic

Теперь, познакомившись с предоставлением объектной модели приложения Microsoft Excel, можно попробовать вызвать методы объекта и задать его свойства. Для этого необходимо написать свой код таким образом, чтобы он распознавался в Microsoft Excel. Обычно это делается с помощью редактора Visual Basic. Несмотря на то, что он установлен по умолчанию, многие пользователи не знают о его наличии, пока этот редактор не будет включен его на ленте.

3.1. Вкладка "Разработчик"

Все приложения Microsoft Excel используют ленту. Одной из вкладок на ленте является вкладка **Разработчик**, на которой можно вызвать редактор Visual Basic и другие инструменты разработчика. Поскольку в Microsoft Excel вкладка **Разработчик** не показана по умолчанию, необходимо вывести ее на экран, выполнив следующую процедуру.

3.2. Включение вкладки "Разработчик"

1. На вкладке **Файл** выберите **Параметры**, чтобы открыть диалоговое окно **Параметры Excel**.
2. Щелкните **Настройка ленты** в левой части диалогового окна.
3. В разделе **Выбрать команды из**, расположенном слева в окне, выберите **Популярные команды**.
4. В разделе **Настройка ленты**, который находится справа в диалоговом окне, выберите **Основные вкладки**, а затем установите флажок **Разработчик**.
5. Нажмите кнопку **ОК**.

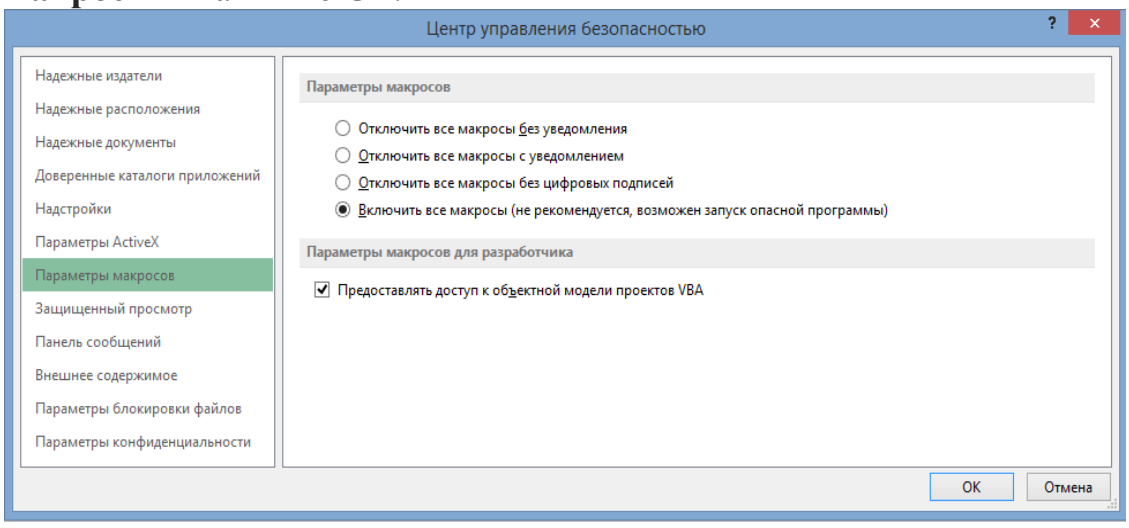
Когда вкладка **Разработчик** появится в интерфейсе Excel, обратите внимание на местонахождение на вкладке кнопок **Visual Basic**, **Макрос** и **Безопасность макросов**.

3.3. Проблемы безопасности

Если при открытии книги между лентой и листом появляется строка **Предупреждение системы безопасности: запуск макросов отключен**, то необходимо обратиться к преподавателю или дежурному администратору.

Кроме того, в качестве мер безопасности, нельзя сохранять макрос в формате файлов Excel, используемом по умолчанию (XLSX-файлы).

Нажмите кнопку **Безопасность макросов**, выберите опцию **Включить все макросы** и нажмите **Ок**.



4. Возможности Visual Basic для генерации программного кода.

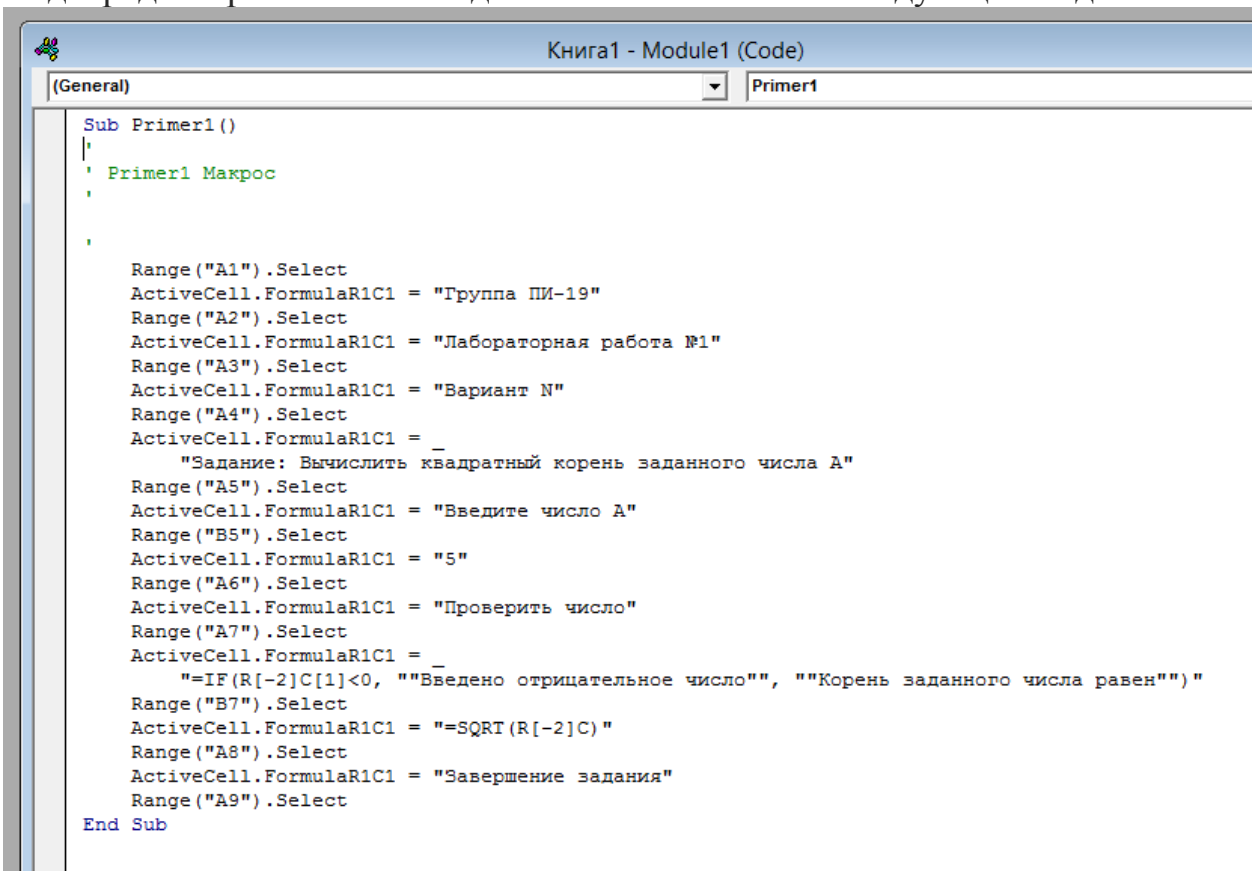
4.1. Использование записи макроса

1. На активном (открытом) листе установите курсор на ячейку A2.
2. На вкладке **Формулы** нажмите кнопку **Показать формулы**
3. На вкладке **Разработчик** нажмите кнопку **Записать макрос** .
4. Назовите макрос **Primer1**.
5. Введите последовательно в каждую ячейку следующий текст:

	A	B
1	Группа ПИ-19	
2	Лабораторная работа №1	
3	Вариант N	
4	Задание: Вычислить квадратный корень заданного числа А	
5	Введите число А	5
6	Проверить число	
7	=ЕСЛИ(B5<0; "Введено отрицательное число"; "Корень заданного числа равен")	=КОРЕНЬ(B5)
8	Завершение задания	
9		
10		
11		

6. Нажать кнопку **Остановить запись**.
7. На вкладке **Разработчик** или **Вид** нажмите кнопку **Макрос** и выберите **Изменить**, чтобы открыть редактор Visual Basic.

Код в редакторе Visual Basic должен быть похож на следующий код.



8. Закройте редактор Visual Basic нажав **Alt+Q.**
9. Откройте новый лист Excel и установите курсор на ячейке А2.
10. На вкладке **Разработчик** или **Вид** нажмите кнопку **Макрос** и выберите **Выполнить.** На рабочем листе появиться следующий текст в результате выполнения макроса **Prim1.**

	A	B	C
1	Группа ПИ-19		
2	Лабораторная работа №1		
3	Вариант N		
4	Задание: Вычислить квадратный корень заданного числа A		
5	Введите число A	5	
6	Проверить число		
7	Корень заданного числа равен	2,236067977	
8	Завершение задания		
9			
10			

5. Выводы

Изложенные в данной лабораторной работе возможности Visual Basic для генерации программного кода дают общее, весьма приближенное представление о возможностях формальных методов по формальному преобразованию формальных спецификаций в программный код. Тем не менее приведенный пример позволяет проиллюстрировать механизм действия формальных методов. Ключевым моментом должно быть понимание того, что имеется однозначное соответствие между элементами формального языка и блоками программного кода.

6. Задание по лабораторной работе.

Задание №1

6.1. Номер варианта соответствует номеру в журнале.

6.2. Согласно заданному варианту подготовить блок-схему алгоритма решения задачи для выполнения на Excel.

6.3. Сгенерировать программный код (макрос) с использованием вкладки **Разработчик.**

6.4. На новом рабочем листе получить результаты выполнения макроса.

6.5. Подготовить отчет, включающий:

- описание задачи №1;
- блок-схему алгоритма решения задачи №1;
- фрагмент рабочего листа EXCEL с записанной задачей;
- сгенерированный программный код на Visual Basic (макрос);
- фрагмент рабочего листа с результатом выполнения макроса;

Варианты задания №1:

Вариант 1:

Вычислить значения функции y для любого заданного x:

$$y = \begin{cases} x^2 \text{ при } x > 0 \text{ и } x \leq 1, \\ -2x \text{ при остальных } x. \end{cases}$$

Вариант 2:

Вычислить значения функции y для любого заданного x:

$$y = \begin{cases} e^{2x} \text{ при } x > 1 \text{ и } x \leq 0, \\ 3x \text{ при } x > 0 \text{ и } x < 0{,}5, \\ \ln(2 + x) \text{ при остальных } x. \end{cases}$$

Вариант 3:

Вычислить значения функции y для любого заданного x:

$$y = \begin{cases} Sinx^2 \text{ при } x > 0 \text{ и } x \leq 1, \\ Cos(2 - x) \text{ при остальных } x. \end{cases}$$

Вариант 4:

Вычислить значения функции y для любого заданного x:

$$y = \begin{cases} Logx^2 \text{ при } x > 0 \text{ и } x \leq 1, \\ Sin(1 - 2x) \text{ при остальных } x. \end{cases}$$

Вариант 5 :

$$y = \begin{cases} Lnx^2 \text{ при } x > 0 \text{ и } x \leq 1, \\ tg(2x) \text{ при остальных } x. \end{cases}$$

Вариант 6:

Проверить тождество

$$\frac{1}{1\,2} + \frac{1}{2\,3} + \frac{1}{3\,4} + \cdots + \frac{1}{n(n+1)} = \frac{n}{n+1}.$$

при $n = 2, 5, 6$

Вариант 7:

Проверить тождество
$\frac{1}{1\,3}+\frac{1}{3\,5}+\frac{1}{5\,7}+\cdots+\frac{1}{(2n-1)(2n+1)}=\frac{n}{n+1}$.
при $n=3,4,7$

Вариант 8:

Решить методом обратной матрицы систему лирейных уравнений:

$$\begin{cases} 2x+3y-z=3, \\ 3x-3y+z=2, \\ 2x-3y-z=-3. \end{cases}.$$

Вариант 9:

Решить методом Крамера систему лирейных уравнений:

$$\begin{cases} 2x+3y-z=3, \\ 3x-3y+z=2, \\ 2x-3y-z=-3. \end{cases}.$$

Вариант 10:

Найти матрицу, обратную к матрице

$$\begin{pmatrix} 2 & 3 & -1 \\ 3 & -3 & 1 \\ 2 & -3 & -1 \end{pmatrix}.$$

Вариант 11:

Для равностороннего треугольника с заданной стороной А вычислить его площадь. Вычислить площадь вписанного в этот треугольник круга.

Вариант 12:

Для ромба с заданной стороной А и углом α вычислить его площадь. Вычислить площадь вписанного в этот ромб круга.

Вариант 13:

Для равносторонней пирамиды с заданной стороной А вычислить ее объем. Вычислить объем вписанного в эту пирамиду шара.

Вариант 14:

Для куба с заданной стороной А вычислить его объем. Вычислить объем вписанного в этот куб шара.

Вариант 15:

Найти произведение матриц:

$$\begin{pmatrix} 2 & 3 & 2 \\ -1 & 1 & 1 \\ 3 & 2 & -1 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 \\ 1 & -7 \\ 4 & -5 \end{pmatrix}$$

Вариант 16:

Решить квадратное уравнение:

$x^2 - 6x - 16 = 5$

Вариант 17:

Решить квадратное уравнение:

$x^2 - 5x - 14 = 7$

Вариант 18:

Решить квадратное уравнение:

$2x^2 - 4x - 16 = 3$

Вариант 19:

Решить квадратное уравнение:

$x^2 - 6x + 8 = 15$

Вариант 20:

Решить квадратное уравнение:

$x^2 + 6x - 55 = 26$

Вариант 21:

Решить квадратное уравнение:

$x^2+10x - 39=0$

Вариант 22:

Решить квадратное уравнение:

$x^2 + 17x - 270=0$

Вариант 23:

Решить квадратное уравнение:

$3x^2+15x - 59=10$

Вариант 24:

Решить квадратное уравнение:

$5x^2 + 37x - 170=50$

Вариант 25:

Решить квадратное уравнение:

$13x^2 + 8x - 45=3$