Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА

ЧИСЛЕННОЕ РЕШЕНИЕ ТИПОВЫХ ЗАДАЧ С ПОМОЩЬЮ VBA

Методические указания к курсовой работе для студентов направления 27.03.01 «Стандартизация и метрология»

САНКТ-ПЕТЕРБУРГ

2017

УДК 004.424
ИНФОРМАТИКА. ЧИСЛЕННОЕ РЕШЕНИЕ ТИПОВЫХ ЗАДАЧ С ПОМОЩЬЮ VBA: Методические указания к курсовой работе / Санкт-Петербургский горный университет. Сост.: *Косарев О.В.,* СПб.: 2017, 36 с.

Рассмотрены основные приемы программирования на языке Visual Basic for Applications. Показаны примеры выполнения: математических расчетов, вычисления определенного интеграла методом квадратур Гаусса, решения системы линейных уравнений с помощью VBA и построения графика. Приведены задания по вариантам.

Научный редактор: заведующий кафедрой ИиКТ Маховиков А.Б.

Ил. 25, Библиогр. 9 назв.

ВВЕДЕНИЕ

Эта курсовая работа посвящена основам программирования. Что такое программирование? Ответ на этот вопрос можно найти в словаре, в википедии или учебнике. В двух словах – это создание программ. Программ для домашнего компьютера, для компьютера на работе, для смартфона или даже для умного дома. Программы окружают нас повсюду. Практически в любой сфере деятельности используется автоматизация рутинных операций. Не всегда для этого требуется специальная программа. Довольно часто нужно уметь просто написать небольшую последовательность команд. Например, для проверки вводимых данных в табличку Excel. Конечно, профессиональный программист гораздо лучше (наверное) справится с написанием программы. Но как вы сможете с ним договориться, если не будете понимать, как это работает? Программирование развивает абстрактное мышление, учит однозначно описывать задачу на универсальном языке, точно описывать ожидаемый результат. Это как раз те навыки, которые требуются от дипломированного специалиста.

В качестве примера в этой курсовой работе выбран язык программирования Visual Basic for Applications. Выбор именно этого языка не случаен. Он прост, но обладает большими возможностями, как и более сложные языки программирования. Для этого языка написано множество учебных пособий и существует множество форумов с примерами решений. Но самое главное его достоинство – это доступность. Он доступен на каждом компьютере, на котором установлен пакет Microsoft Office.

1 ИСХОДНЫЕ ДАННЫЕ

Необходимо запрограммировать на языке Visual Basic for Applications (VBA) **решение задач**:

1. Вычислить значение математического выражения
2. Вычислить значение определенного интеграла методом квадратур Гаусса
3. Решить систему линейных алгебраических уравнений с помощью обратной матрицы и метода Крамера
4. Построить график функции

Решение задачи **должно содержать**:

1. Описание метода решения задачи
2. Блок-схему решения задачи, оформленную в виде рисунка
3. Код программы с комментариями
   В коде программы необходимо показать применение операторов ввода-вывода, цикла, проверку вводимых значений на ошибку, использование процедуры.
4. Проверку полученного решения в MS Excel и Mathcad
5. Интерактивную форму выбора задания для просмотра ответа
6. Ввод коэффициентов в программу необходимо делать двумя способами: через считывание из ячеек листа Excel и через форму
7. Промежуточные вычисления (определители, обратная матрица, частичная сумма и пр.) и ответ должны отображаться в форме и выводиться на лист Excel

Варианты заданий определяются по учебному пособию [1]. Пояснительная записка должна быть оформлена в соответствии с [2]-[4]. Блок схемы решения задач должны быть оформлены в соответствии с [5]. Теоретический материал по основам алгоритмов и программированию на VBA изложен в материалах лекций по дисциплине и в [6]-[9].

2 ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЙ

2.1 РЕШЕНИЕ МАТЕМАТИЧЕСКОГО ВЫРАЖЕНИЯ

Дано математическое выражение (1). Необходимо вычислить его значение.

$$\sin(5x+2)-3x^2+3,\ x=3 \qquad (1)$$

Решение задачи предполагает простое вычисление выражения для заданного значения переменной. Для решения задачи необходимо: ввести значение переменной из ячейки Excel (с экрана), выполнить арифметические операции, вывести ответ в ячейку Excel (на экран). Блок схема решения выражения (1) показана на рисунке 1.

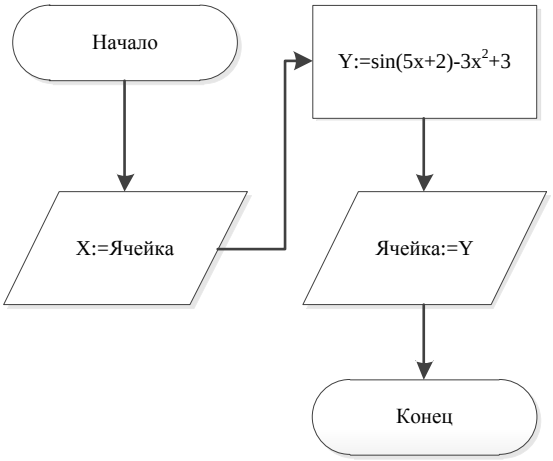


Рис. 1. Блок-схема решения выражения (1)

Сначала запрограммируем ввод значения переменной из ячейки Excel, а затем ввод через диалоговое окно. Определим на листе Excel место для записи выражения и ответа. Поместим значение переменной в ячейке B2 (рисунок 2).

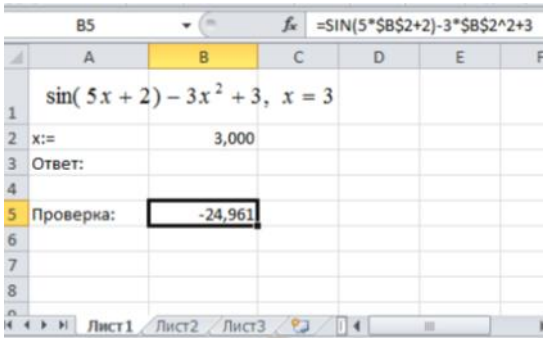


Рис. 2. Исходные данные на листе Excel

Обратите внимание, что в ячейку B5 мы поместили формулу с нашим выражением. Ответ, полученный в макросе VBA, выведем в ячейку B3.

Для создания макроса необходимо сохранить нашу книгу Excel в формате с поддержкой макросов (****.xlsm) и перейти в редактор VBA. В нашем проекте необходимо создать модуль Module 1, в котором будем писать код программы (рисунок 3).

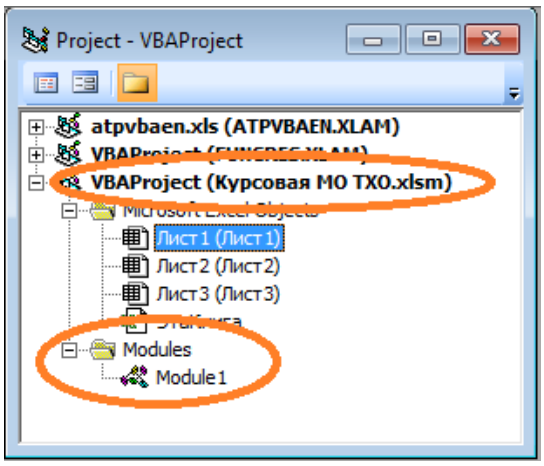


Рис. 3. Новый модуль в проекте VBA

Теперь напишем код программы, которая будет выполнять алгоритм на рисунке 1. Для этого дважды щелкнем по Module1 в редакторе VBA. Откроется окно кода модуля. Обратите внимание на то, где именно писать код. Код можно писать в проекте, в листе и в модуле. Мы будем писать код именно в модуле. Окно кода модуля и код для задания 1 показаны на рисунке 4.

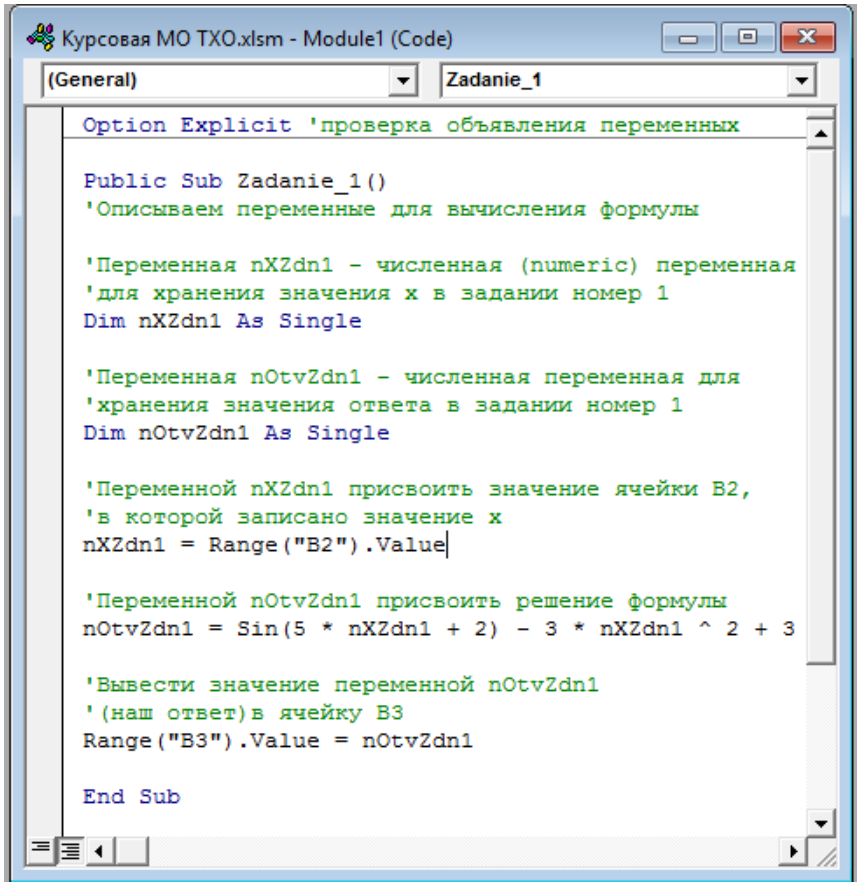


Рис. 4. Код программы для Задания 1

Первая строчка кода – команда Option Explicit. Эта команда проверяет, все ли переменные к программе объявлены. Это не обязательная команда, программа будет работать и без неё. Далее нам необходимо создать новую процедуру. Назовем ее **Zadanie_1**. Процедура начинается командой **Public Sub Zadanie_1()** и заканчивается командой **End Sub**. Вручную писать эти команды не надо, вставка новой процедуры делается через меню Insert\Procedure. Следующие две строки кода – это описание имен и типов переменных. В первом задании две переменных – x и ответ. Переменная для хранения значения x – **nXZdn1**, где n – тип переменной (численный), Zdn1 - принадлежность переменной к заданию 1. Переменная для хранения ответа - **nOtvZdn1**. Тип переменных – Single, как наименьший формат, поддерживающий дробные числа. Принцип составления имени переменной **nOtvZdn1** такой же, как и в предыдущем случае. Такое наименование не обязательное, можно использовать любые другие имена. Однако следование такому правилу записи имен переменных (венгерской нотации) позволяет лучше ориентироваться в коде при большом количестве переменных. При решении задач курсовой работы необходимо следовать этому правилу.

В следующей строке мы записываем в переменную **nXZdn1** значение x, указанное на листе Excel в ячейке B2. Для этого используем команду **nXZdn1 = Range("B2").Value**. Здесь Range – объект Excel, описывающий диапазон ячеек на листе. В данном случае диапазон включает всего одну ячейку B2. Свойство Value используется для присвоения переменной значения ячейки.

Далее мы присваиваем переменной nOtvZdn1 значение выражения «$\sin(5x+2)-3x^2+3,\ x=3$». Для этого используется команда **nOtvZdn1 = Sin(5*nXZdn1+2) - 3*nXZdn1^2 + 3**.

После того, как выражение посчитано, необходимо вывести ответ из переменной **nOtvZdn1** в ячейку B5 листа Excel. Для этого воспользуемся командой **Range("B3").Value = nOtvZdn1**. Эта команда аналогична команде считывания из ячейки, только работает наоборот. Результат выполнения нашей программы (макроса) показан на рисунке 5. В ячейке B3 появился результат вычисления выражения (1). Он совпадает с расчетом непосредственно в ячейке B5.

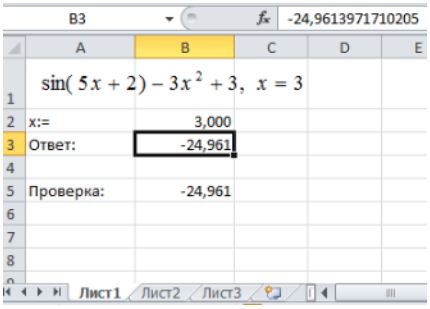


Рис. 5. Результат выполнения программы Zadanie_1

Теперь напишем код программы, который будет запрашивать значение x у пользователя и выводить ответ на экран. Для организации диалога программы с пользователем будем использовать команды **InputBox** (блок ввода) и **MsgBox** (блок вывода). Код программы показан на рисунке 6. Это модифицированный код предыдущего решения, написанный в модуле Module 2.

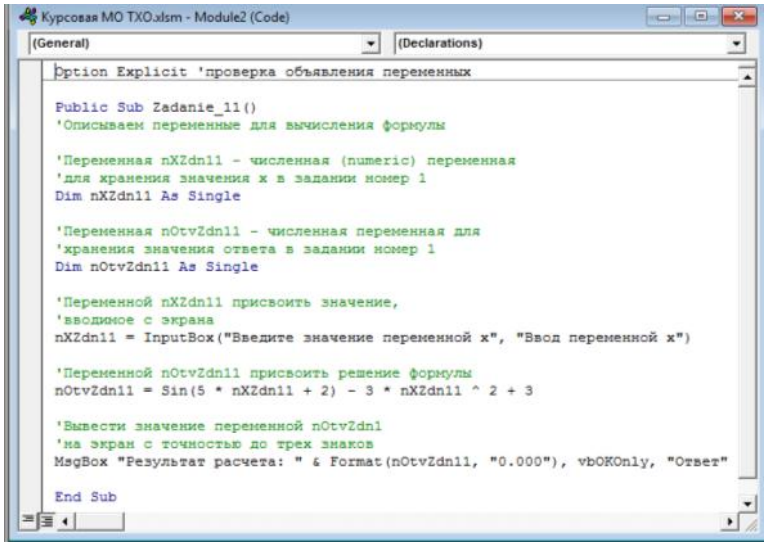


Рис. 6. Код программы Zadanie_11

В этом коде мы изменили название процедуры на **Zadanie_11**. Это все еще первое задание, но второй вариант решения. Все переменные также получили в названии дополнительную единицу. Сам код написан в созданном модуле Module 2, что видно в шапке программы. Значение x вводится с экрана через форму. Для этого используется команда **nXZdn11 = InputBox("Введите значение переменной x", "Ввод переменной x")**. Переменной **nXZdn11** присваивается введенное в форму значение. Экранная форма ввода создается функцией **InputBox**. Эта функция позволяет написать пояснительный текст над окном ввода ("Введите значение переменной x") и подписать шапку окна ввода ("Ввод переменной x"). Результат выполнения это функции показан на рисунке 7.

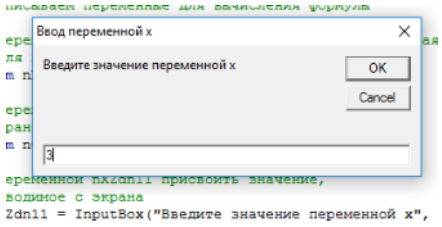


Рис. 7. Результат выполнения функции InputBox

Вычисленное по формуле значение записывается в переменную **nOtvZdn11**. Затем значение этой переменной выводится на экран с помощью функции **MsgBox "Результат расчета: " & Format(nOtvZdn11, "0.000"), vbOKOnly, "Ответ"** (рисунок 8).

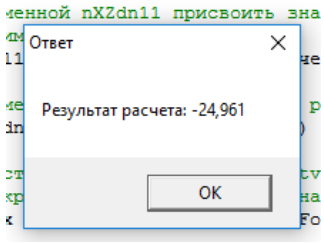


Рис. 8. Результат выполнения функции MsgBox

Функция **MsgBox** выводит на экран текстовое сообщение «**Результат расчета:** » и значение переменной **nOtvZdn11** в одну строку с помощью оператора конкатенации строковых данных **&**. В данном случае ответ меняет тип с численного на числеПный. предварительно значение переменной **nOtvZdn11** приводится к виду трех знаков после запятой (форматируется) с помощью функции **Format(nOtvZdn11, "0.000")**. В этой функции выражение **"0.000"** определяет вид ответа. Параметр функции **vbOKOnly** определяет наличие кнопки «**Ок**» на форме вывода результата. Параметр **"Ответ"** определяет название формы с ответом. Это название видно в шапке формы на рисунке 8.

Вводить дробные числа в окно формы **InputBox** следует через запятую. Такая запись соответствует американскому формату чисел. Если вместо запятой поставить точку, программа выдаст ошибку. Аналогичная ошибка возникнет, если вместо цифр ввести текст. Целесообразно предусмотреть проверку типа вводимых данных. Для проверки типа вводимых данных будем использовать функцию **IsNumeric(nXZdn11)**. Эта функция проверяет, является ли введенное значение числом. Содержимое **InputBox** хранится в переменной **nXZdn11.** Поэтому необходимо чтобы эта переменная могла хранить не только числовые значения, но и текстовые. Иначе программа будет останавливаться из-за несоответствия типов переменной и вводимых данных. Для этого переменной **nXZdn11** необходимо присвоить тип **Variant**. Этот тип позволяет хранить в переменной любые данные. Функция **IsNumeric** возвращает логическую истину (True) если введено числовое значение, или логическую ложь (False) в противном случае. Если в **InputBox** будет введено текстовое значение, программа выдаст соответствующее предупреждение и прекратит работу. Проверка условия введенного значения организуется с помощью функции **If…Then** (Если условие выполняется, То…). Выход из процедуры (выход из нашей программы **Zadanie_11**) организован при помощи команды **Exit Sub.** Код программы показан на рисунке 9. Обратите внимание, на изменение типа переменной **nXZdn11**, организацию проверки условия и выход из процедуры (на рисунке показан стрелкой). Сообщение об ошибке при вводе числа с точкой показано на рисунке 10.

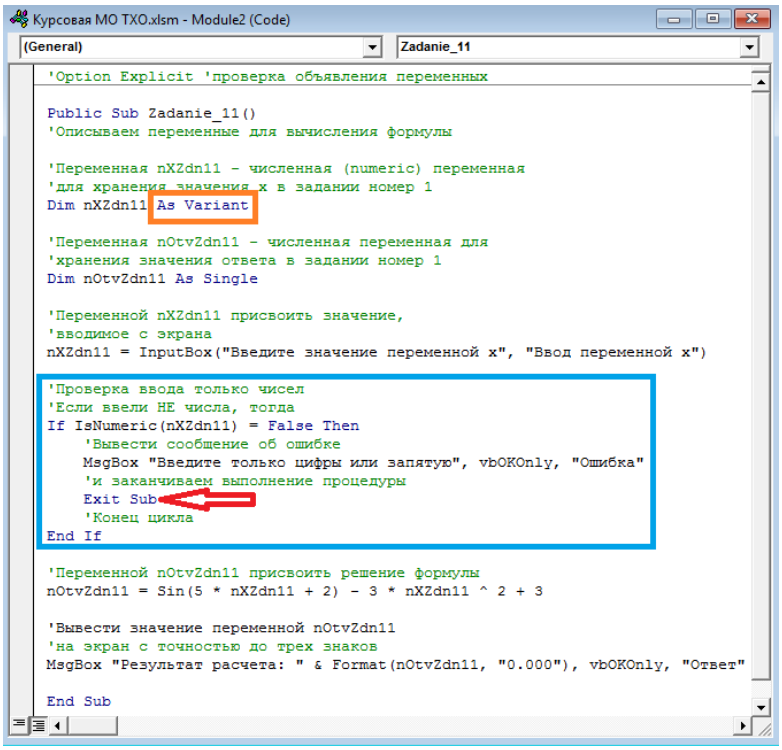


Рис. 9. Использование функции IsNumeric и условия If…Then

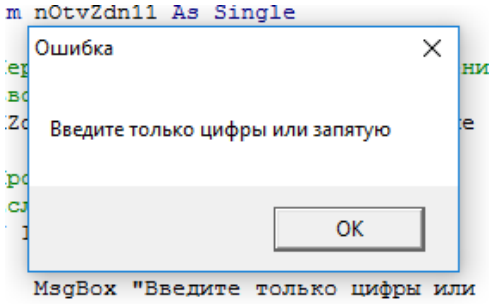


Рис. 10. Сообщение об ошибке

При обнаружении ошибки ввода нужно, чтобы программа запрашивала ввод данных заново. Для этого программа должна возвращаться на выполнение в точку запроса ввода данных. Такой алгоритм работы возможно организовать при помощи цикла. Конструкция (содержимое) цикла такова:

1. показать окно ввода переменной пока
2. если введено число, выйти из цикла
3. если введено НЕ число, вывести сообщение об ошибке и вернуться на шаг 1

Цикл в VBA организуется при помощи функции **Do … Loop**. Функция **Do … Loop** работает так: тело цикла (код между оператором **Do** и **Loop**) выполняется до тех пор, пока внутри цикла не будет дана команда на выход из цикла. Если такой команды не будет, то цикл будет выполняться бесконечно. Выход из цикла задается командой **Exit Do**. Остальные функции для организации тела цикла (пункты 1-3 выше) нам уже знакомы. Запишем конструкцию цикла еще раз, но с учетом новых функций и команд:

1. показать окно ввода переменной пока (**Do**, **InputBox**)
2. если введено число, выйти из цикла (If IsNumeric… Then Exit Do)
3. если введено НЕ число, вывести сообщение об ошибке и вернуться на шаг 1 (**MsgBox**, **Loop**)

Код программы с возможностью обработки ошибки ввода показан на рисунке 11. Обратите внимание на функцию **IsNumeric(nXZdn11) = True**. Параметр **True** не является обязательным, функция возвращает его по умолчанию. Поэтому часть функции «**= True**» можно опустить (не печатать в коде), программа будет работать точно так же. По сравнению с кодом на рисунке 9 имеются отличия в функции **If…Then**. На рисунке 9 функция заканчивается командой **End If**. На рисунке 11 такой команды нет. Если команда, которую нужно выполнить при истинности проверяемого условия, расположена сразу после **Then**, то **End If** можно не писать. В случае

нескольких команд или продолжении проверки условия через конструкцию **Else**/**ElseIf**, окончание **End If** обязательно.

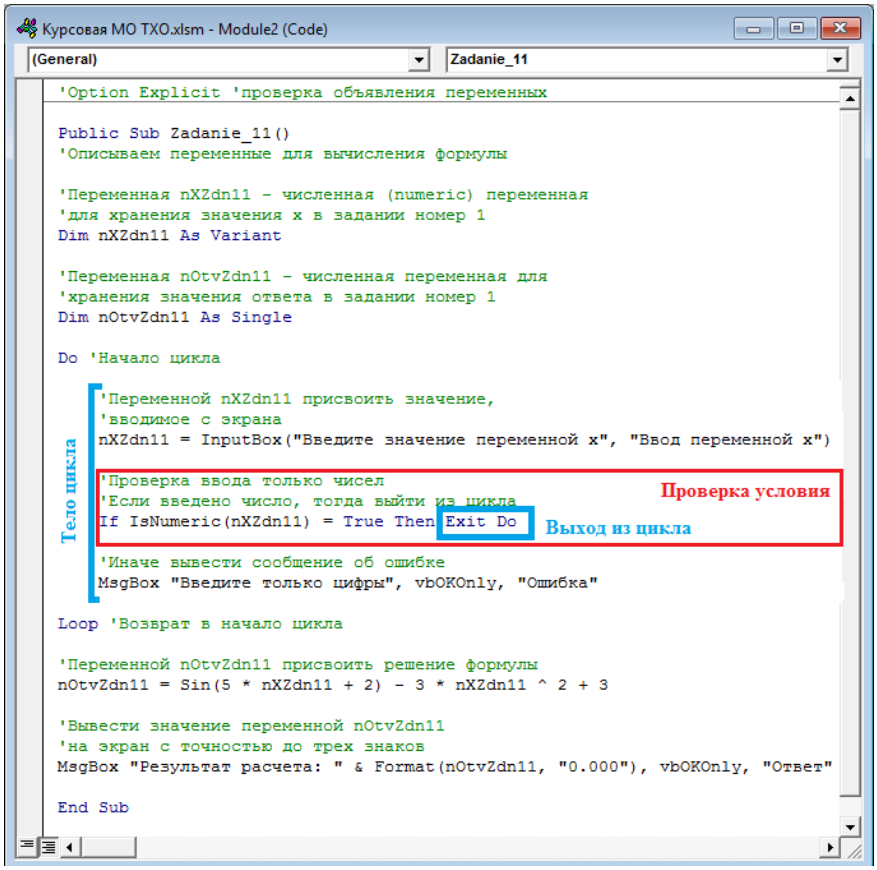


Рис. 11. Обработка ошибки ввода

В процессе написания кода возникают ситуации, когда программа зацикливается. Например, невозможно выйти из приглашения ввести переменную **x**, нажав кнопку **Cancel**. В таких случаях (когда программа зацикливается) выйти из программы можно при помощи комбинации клавиш **Ctrl+Break**. В нашем конкретном слу-

чае нажатие кнопки **Cancel** на форме ввода можно обработать с помощью функции **StrPtr**. Эта функция относится к недокументированным (не описанным в руководстве по VBA) возможностям. Разбирать подробно принцип работы этой функции мы не будем. Используем только то свойство, что она возвращает значение **False** при нажатии кнопки **Cancel.** Если после запуска программы пользователь не хочет вводить данные и решил выйти из программы, то он нажмет кнопку **Cancel.** Рассмотрим алгоритм обработки такого события.

Программа запускается и выводит приглашение на ввод значения переменной (рисунок 7). Пользователь передумал и нажимает кнопку **Cancel** на форме приглашения**.** Программа должна проверить, какое значение вернула функция **StrPtr** после выполнения команды **InputBox.** Аргумент функции **StrPtr** – это переменная, которой присваивается значение функции **InputBox.** Если функция вернула значение **False,** то необходимо выйти из программы. Наша программа – это процедура **Sub.** Следовательно, выйти из программы – означает выйти из процедуры. Выход из процедуры реализуется командой **Exit Sub.** Код обработки события «Нажата кнопка **Cancel**» будет в нашем случае будет выглядеть так: **If StrPtr(nXZdn11) = False Then Exit Sub.** Эту строку кода необходимо поместить вначале цикла, сразу после строки ввода значения переменной **x.** Фрагмент модифицированной программы (только цикл) показан на рисунке 12. Теперь при нажатии клавиши **Cancel** наша программа будет корректно заканчивать свою работу.

Выполнение программы на VBA зависит от версии пакета MS Office и подключенных библиотек функций и подпрограмм. Зацикливание программы при нажатии кнопки **Cancel** может возникать не всегда. В некоторых случаях код программы может выдавать ошибки синтаксиса. Возможным решением проблемы синтаксиса может быть указание полного имени функции. Например, вместо **IsNumeric** указать **VBA.IsNumeric.** Общим правилом написания программы должен быть принцип «От простого – к сложному». Принцип означает, что сначала нужно написать код для простых действий: описать переменные, организовать ввод-вывод, запрограммировать математические выражения. Нужно добиться, чтобы

программа работала без ошибок. После этого можно переходить к более сложным вещам, например к циклам или массивам. Обработку редких ошибок и улучшение интерфейса взаимодействия с пользователем следует оставить на последний шаг. Еще раз: **сначала программируем основную функцию задачи, затем улучшаем код.**

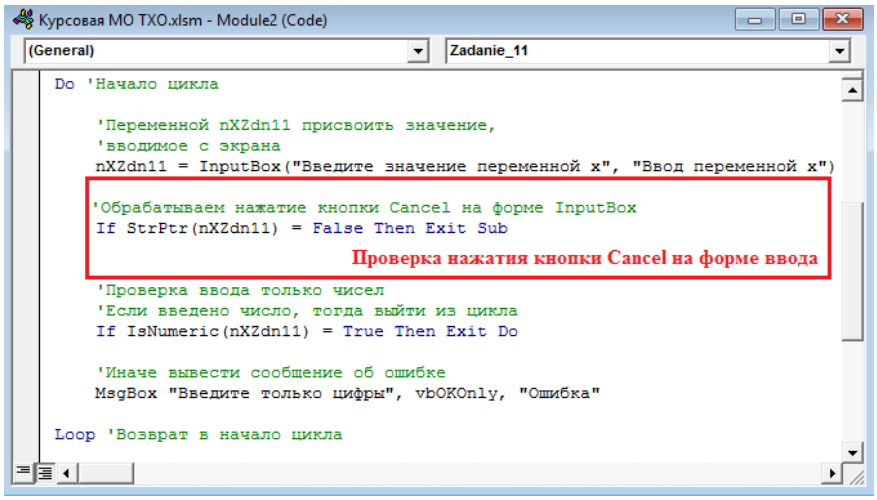


Рис. 12. Обработка нажатия клавиши Cancel на форме InputBox

2.2 ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА

В этом разделе для вычисления определенного интеграла будем использовать массивы. Вычислим определенный интеграл вида (2) методом квадратур Гаусса четвертого порядка:

$$\int_{0.2}^{2.4} \frac{\sqrt{x^2+1}}{x+2}\,dx \qquad (2)$$

Порядок решения этой задачи в Excel подробно указан в [1] с.29. для приближенного вычисления используется выражение (3) и (4).

$$\int_a^b f(x)dx \cong \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i) \qquad (3)$$

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} t_i \qquad (4)$$

где: $f(x)$ - подынтегральная функция,

b, a - верхний и нижний пороги и интегрирования,

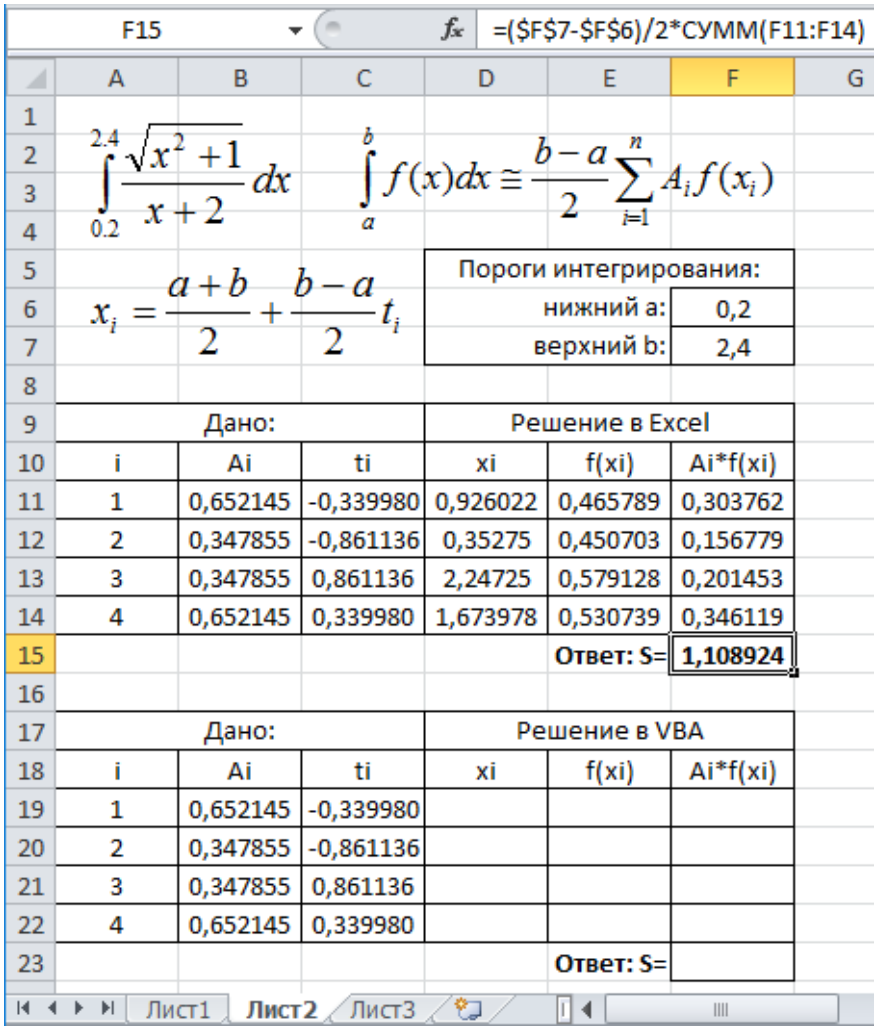
n - порядок метода,

A_i, t_i - заданные коэффициенты для метода порядка n.

Создадим на втором листе нашего файла Excel таблицу с коэффициентами ***Ai*** и ***ti***. Остальные значения вычислим с помощью Excel и VBA. На рисунке 13 показан Лист2 с заготовкой для вычисления определенного интеграла. Решение в Excel уже известно, поэтому описывать его не будем, просто приведем для сравнения. В программе будем вычислять правую часть таблицы. При этом табличные коэффициенты будем считывать в программу из левой части таблицы в массив в программе VBA.

Итак, для решения интеграла в VBA нам понадобится много переменных. Почти все переменные будут массивами. В нашей таблице есть пять столбцов с коэффициентами и результатами вычислений. Каждый столбец – это массив размерностью 4 строки на 1 столбец. В VBA массив задается таким же способом, как и переменная, но в скобках имени массива указывается его размерность: Имя(строки, столбцы). Пример: **Dim Massiv(2,2) As Single.** В этой строке кода мы объявили переменную **Massiv**, которая имеет размер 3 строки (от 0-й до 2-й) на 3 столбца (от 0-го до 2-го). В данном случае мы указали верхний предел номера строки и столбца, считая с

нулевой строки и нулевого столбца. По умолчанию индексы массива в VBA начинается с нуля. Это значение можно изменить на 1 с помощью команды **Option Base 1**.



F15 | =(F7-F6)/2*СУММ(F11:F14)

$$\int_{0.2}^{2.4} \frac{\sqrt{x^2+1}}{x+2}\,dx \qquad \int_a^b f(x)dx \cong \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i)$$

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}t_i$$

	Пороги интегрирования:	
	нижний a:	0,2
	верхний b:	2,4

	Дано:			Решение в Excel	
i	Ai	ti	xi	f(xi)	Ai*f(xi)
1	0,652145	-0,339980	0,926022	0,465789	0,303762
2	0,347855	-0,861136	0,35275	0,450703	0,156779
3	0,347855	0,861136	2,24725	0,579128	0,201453
4	0,652145	0,339980	1,673978	0,530739	0,346119
				Ответ: S=	**1,108924**

	Дано:		Решение в VBA		
i	Ai	ti	xi	f(xi)	Ai*f(xi)
1	0,652145	-0,339980			
2	0,347855	-0,861136			
3	0,347855	0,861136			
4	0,652145	0,339980			
				Ответ: S=	

Лист1 | Лист2 | Лист3

Рис. 13. Вычисление определенного интеграла в Excel

Второй способ задания массива – указать начальные и конечные индексы строк и столбцов: **Dim Massiv(1 To 3, 1 To 3) As Single.** В примере кода ниже использован именно такой способ. Для решения интеграла нам понадобятся переменные-массивы для каждого столбца и обычные переменные для: верхнего и нижнего порогов интегрирования, для индекса массива и для результата интегрирования. Переменные-массивы будут иметь размерность 4 строки (по числу строк коэффициентов в таблице Excel) на 1 столбец. Тип переменной – **Single.** В столбце ***f(xi)*** будет вычисляться подынтегральная функция. Она содержит квадратный корень. Функция вычисления квадратного корня в VBA **Sqr()** возвращает значение типа **Double**. Следовательно, переменные для столбцов ***f(xi)*** и ***Ai*f(xi)*** также будут типа **Double**. Для переменных верхнего и нижнего порогов и значения ***xi*** достаточно типа **Single.** Для переменной индекса массива достаточно типа **Integer.** А вот переменная для результата интегрирования должна быть типа **Double**, т.к. в нее будет записана сумма элементов массива ***Ai*f(xi)*** с типом **Double**.

Теперь можно переходить непосредственно к вычислению интеграла. Порядок действий такой: записать в массивы значения коэффициентов ***Ai*** и ***ti*** с листа Excel (для перебора элементов массивов используем цикл), считать в переменные значения порогов интегрирования, рассчитать значения ***xi, f(xi)*** и ***Ai*f(xi)*** (для перебора элементов массивов используем цикл). Затем просуммировать элементы массива ***Ai*f(xi)***, умножить на полуразность порогов интегрирования и записать результат в ячейку F23 на листе Лист2. Блок-схема программы показана на рисунке 14. В программе должно бвть организовано три цикла. В первом цикле заполняются массивы переменных с табличными коэффициентами. В одном цикле создаются сразу две переменных-массива. Во втором цикле заполняется уже три массива, по числу столбцов в таблице на рисунке 13, которые нужно заполнить рассчитанными значениями. В этом же цикле заполняются ячейки на листе Excel. За один такт (круг) цикла последовательно рассчитываются значения ячеек таблицы Excel в одной строке. Так как количество повторений цикла 4, то за четыре последовательных прохода заполняются все строки таблицы. В принципе

содержимое первого и второго циклов (тело цикла) можно было объединить в один цикл. Это немного уменьшило бы объем кода. Но с точки зрения универсальности это неудобно. Каждый цикл выполняет свою функцию и не зависит один от другого. В случае изменения подынтегрального выражения менять код нужно будет только во втором цикле.

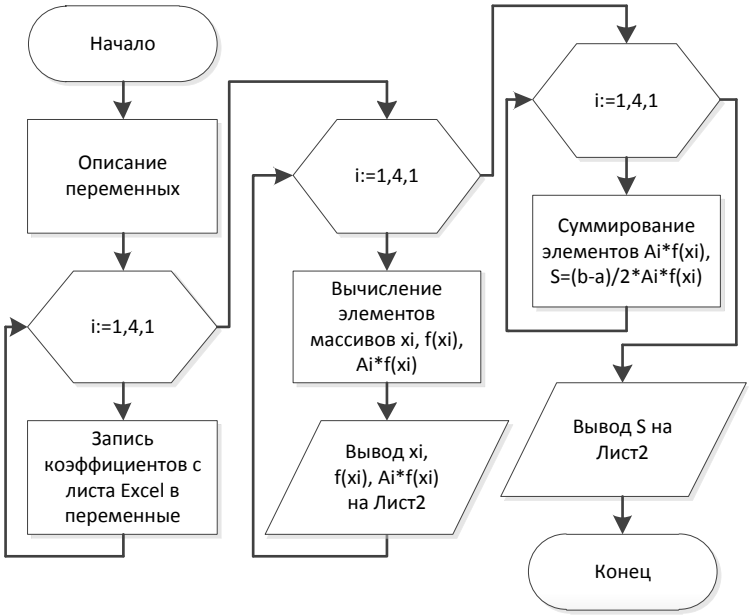


Рис. 14. Блок-схема программы Zadanie_2

Фрагмент кода, где объявлены переменные, показан на рисунке 15. Обратите внимание на имена переменных. В именах появились буквы «arr» - от слова «array», т.е. массив. Также в именах переменных отмечено, что они относятся к заданию №2 – буквы «Zdn2». Рассмотрим переменную **narrAiZdn2.** Это численная переменная, буква «n» в названии. Это массив (array), поэтому в названии буквы «arr». Переменная предназначена для записи коэффици-

ентов ***Ai***. В имени переменной присутствуют буквы «Ai». Переменная из кода программы для выполнения задания №2, поэтому в названии присутствуют буквы «Zdn2». Полное описаниес масива выглядит так: **Dim narrAiZdn2(1 To 4, 1 To 1) As Single**. Эта строка задает переменную с именем **narrAiZdn2**, это массив из 4-х строк и одного столбца, элементы массива имеют тип **Single**. Остальные переменные заданы аналогично.

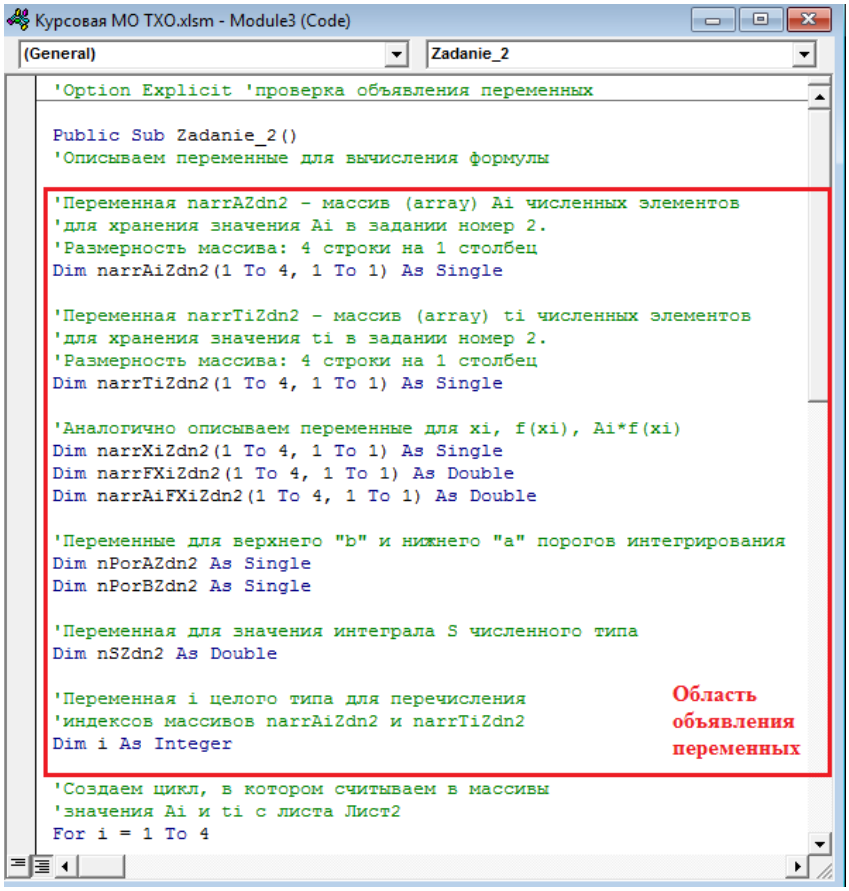


Рис. 15. Объявление переменных для вычисления интеграла

Перейдем теперь к основному коду. Основной код получился большой, поэтому будем рассматривать его по частям. Первая часть соответствует первому циклу нашей блок-схемы. Она показана на рисунке 16.

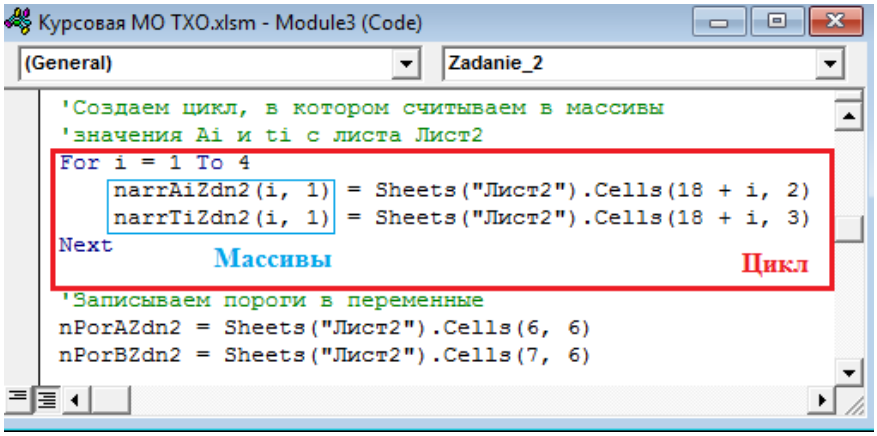


Рис. 15. Формирование массива коэффициентов

Формирование элементов массива происходит в цикле. В предыдущем задании мы уже сталкивались с организацией цикла. В этот раз мы будем использовать «цикл с параметром». Такой цикл создается в VBA при помощи конструкции **For…To…Step…Next**. Внутри этой конструкции указывается диапазон изменения счетчика цикла (**i** изменяется от 1 до 4 в нашем случае). После каждого изменения счетчика последовательно выполняются команды внутри цикла (выполняется тело цикла). После команды **Next** счетчик изменяется на 1 и тело цикла повторяется заново. Величину изменения можно задавать после необязательного параметра **Step** (шаг) – в нашем коде этот параметр опущен. По умолчанию, шаг цикла равен единице, и его можно не указывать. Шаг может быть как положительным, так и отрицательным. Рассмотренная конструкция наиболее предпочтительна при работе с массивами. Внутри цикла элементам массива **narrAiZdn2** присваиваются значения ячеек на листе Excel. В массиве четыре строки и один столбец. Элементы массива

указываются в скобках после имени массива: **narrAiZdn2(i, 1)**. Вместо номера строки элемента массива в скобках стоит переменная **i** (счетчик), значение которой изменяется на каждом шаге цикла от 1 до 4. Таким образом, счетчик цикла используется для последовательного перебора элементов массива. В этом и заключается весь смысл применения конструкции **For…To…Step…Next** для работы с массивами. Столбец в нашем массиве всего один, поэтому достаточно указать его обычным числом. Если бы количество столбцов отличалось от единицы, то нам пришлось бы организовать два вложенных один в другой цикла.

В теле цикла переменным присваивается значение ячеек листа Excel. В предыдущей задаче мы использовали для этого конструкцию **Range("B2").Value**, где **B2** – имя ячейки на листе. Такую\ конструкцию можно использовать, если мы работаем с одним единственным листом в книге (как правило это Лист1) и этот лист активный. Активный лист – это лист в книге Excel, содержимое которого мы видим в текущий момент времени. Второе задание мы делаем на втором листе рабочей книги, поэтому использовать рассмотренную выше конструкцию не правильно. Поскольку у нас уже два листа, на которых есть информация, то активным может оказаться любой из них. В этом случае считывание данных с листа в переменную и обратно может произойти не с того листа. Поэтому необходимо указывать, с какого конкретно листа книги (а в общем случае и из какой рабочей книги!) необходимо считывать информацию. На рисунке 15 видно, что для этого используется команда **narrAiZdn2(i, 1) = Sheets("Лист2").Cells(18 + i, 2)**. Разберем эту команду подробно.

Нас интересует правая часть команды, а именно **Sheets("Лист2").Cells(18 + i, 2)**. В этой команде мы видим указание на лист нашей книги (**Sheets("Лист2")**) и на конкретную ячейку (**Cells(18 + i, 2)**). В указании на лист книги присутствует имя книги, а в указании на ячейку – адрес ячейки на листе. Стиль ссылок – R1C1. Это означает, что адрес ячейки указывается как номер строки (**R**ow) и номер столбца (**C**olumn), на пересечении которых она находится. Нумерация начинается с единицы из левого верхнего угла.

Номер строки указан как некоторое значение плюс счетчик цикла, т.к. мы в теле цикла перебираем строки на листе Excel.

Две последние строки в коде на рисунке 15 – это запись в переменные верхнего и нижнего порогов интегрирования. На этом подготовительная часть программы заканчивается и начинается непосредственное вычисление интеграла. Код для вычисления столбцов ***xi***, ***f(xi)*** и ***Ai*f(xi)*** показан на рисунке 16.

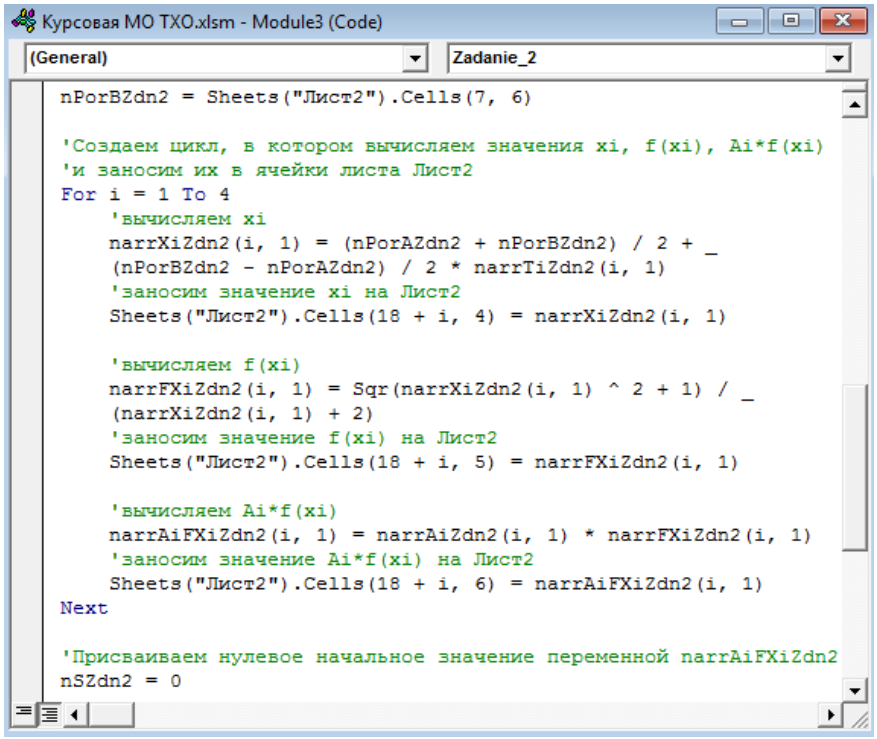


Рис. 16. Вычисление столбцов *xi*, *f(xi)* и *Ai*f(xi)*

В этом коде используются уже знакомые нам конструкции. Последовательно вычисляются значения ***xi***, ***f(xi)*** и ***Ai*f(xi)*** по формулам (3) и (4). Полученные значения записываются в ячейки на

листе Excel. Последний шаг – просуммировать столбец ***Ai*f(xi)***, умножить полученную сумму на полуразность порогов интегрирования и записать ответ в ячейку Excel. Заключительный фрагмент кода показан на рисунке 17.

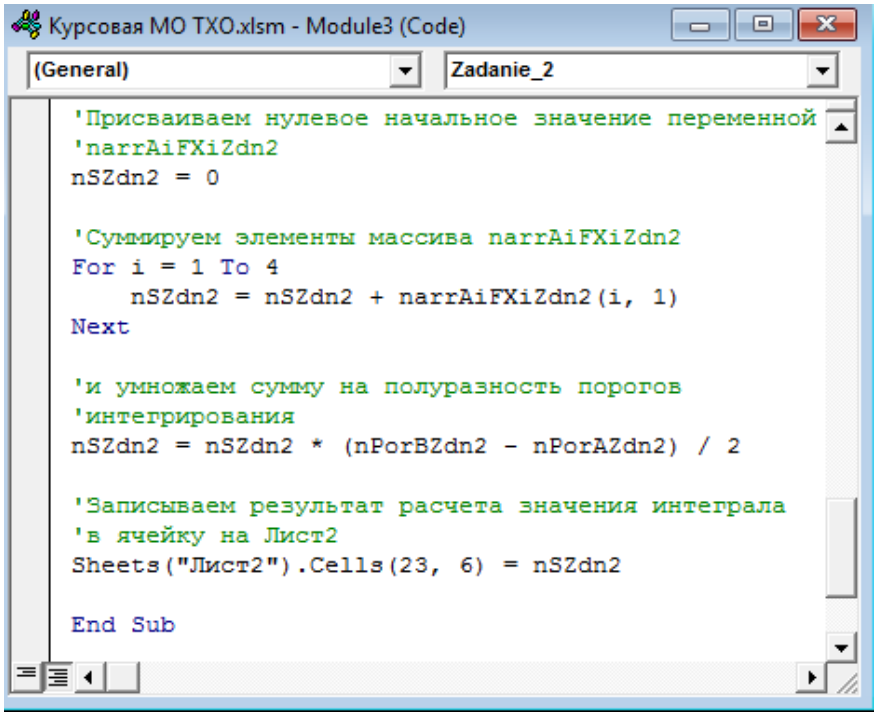


Рис. 17. Заключительный фрагмент кода вычисления определенного интеграла

При решении задачи мы считывали коэффициенты и пороги интегрирования с листа Excel. Организуем теперь ввод этих величин через окна ввода. В предыдущей задаче мы использовали для этого функцию InputBox. Теперь воспользуемся вводом с помощью форм пользователя. С помощью форм пользователя можно создать графйический интерфейс своей программы. Форма пользователя добавляется в проект командой **Insert\UserForm**. Создадим форму,

на которой будут размещены: два окна для ввода порогов, кнопка расчета интеграла и окно вывода результата. Форма расчета интеграла в режиме конструктора (сверху) и в режиме выполнения макроса (снизу) показана на рисунке 18.

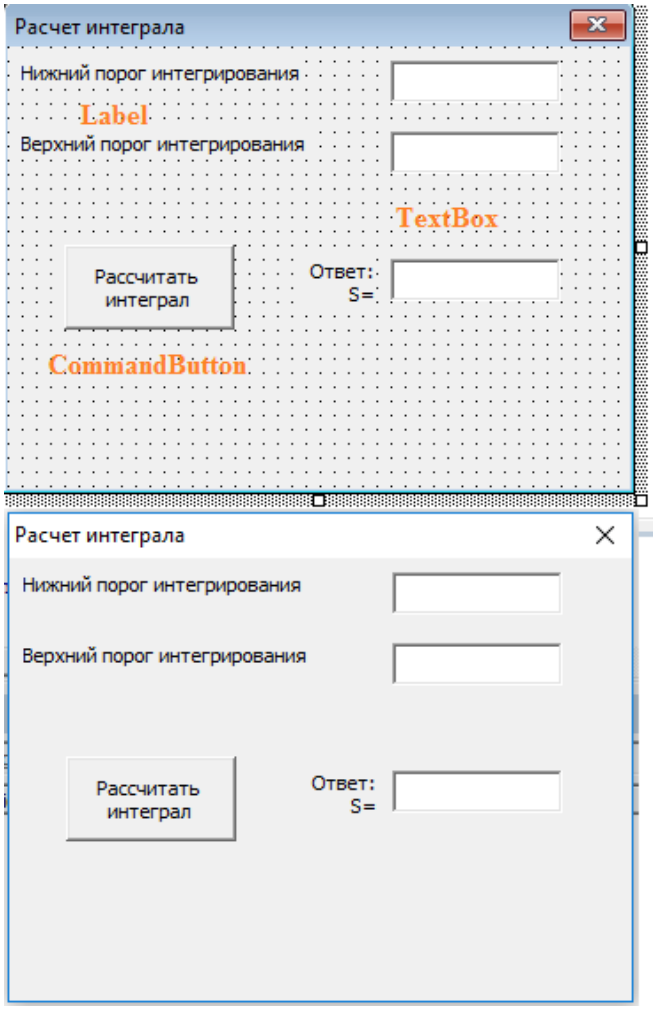


Рис. 18. Пользовательская форма для вычисления определенного интеграла

Содержимое формы создается с помощью перетаскивания элементов формы из окна **Toolbox**. Окно появляется автоматически при создании формы либо командой **View\Toolbox**. Форма на рисунке 18 содержит: три элемента **Label**, три элемента **TextBox**, один элемент **CommandButton**. Элемент **Label** (Надпись) – это просто область формы, в которой находится просто какая-то надпись. Элемент **TextBox** (Текстовое поле) – это область формы, в которой пользователь может вводить данные (и наоборот). Элемент **CommandButton** (Кнопка, кнопка команды) – управляющий элемент формы. Один из наиболее часто используемых элементов. Применяется для управления работой кода формы. Элементы формы располагаются на форме произвольно, расположение задается логикой пользования программой.

Для перехода к коду формы необходимо дважды щелкнуть по полю формы или элементу формы. Форма и каждый элемент формы создает в коде процедуру **Private Sub…End Sub**. На рисунке 19 показан код формы до описания элементов формы (по умолчанию). Такой код получается, если на форме последовательно щелкнуть мышкой по каждому элементу формы. Запуск программы вычисления определенного интеграла происходит по нажатию кнопки **CommandButton1** (Рассчитать интеграл). Поэтому весь код программы будет расположен в процедуре **Private Sub CommandButton1_Click()…End Sub**.

Нам необходимо перенести код программы, написанный для вычисления определенного интеграла, в процедуру кнопки. Внесем некоторые изменения. Зададим коэффициенты ***Ai*** и ***ti*** непосредственно в программе. Это постоянные коэффициенты, они одинаковые для любого интеграла, вычисляемого таким методом. Код записи коэффициентов показан на рисунке 20. Вторым изменением будет ввод величин порогов интегрирования не с листа Excel, а из формы. Введенные в текстовые поля пороги интегрирования необходимо считать в соответствующие переменные. Считывание из текстового поля формы в переменную делается командой **nPorAZdn21 = Val(TextBox1.Text)**. Значения, введенные в текстовые поля, имеют строковый тип.

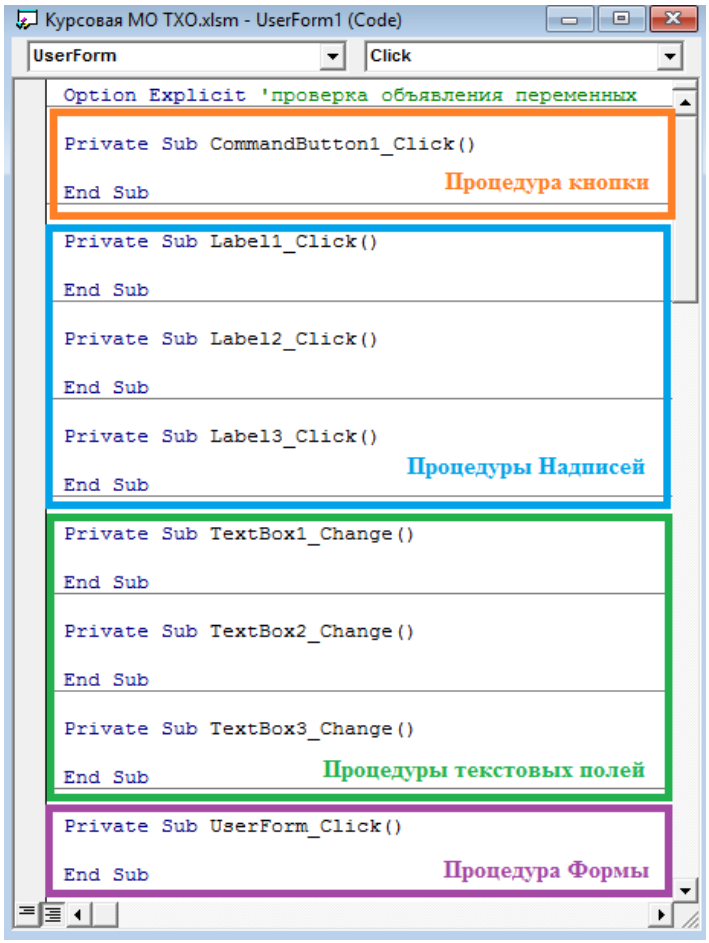


Рис. 19. Код формы пользователя до описания элементов

Для арифметических вычислений необходимо преобразовать текст в число. Такое преобразование выполняется функцией **Val** (Value). Код считывания порогов интегрирования показан на рисунке 21. теперь мы можем менять пороги интегрирования!

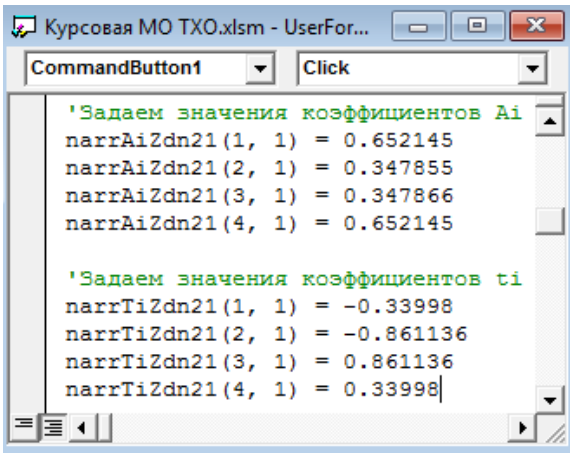


Рис. 20. Запись коэффициентов ***Ai*** и ***ti***

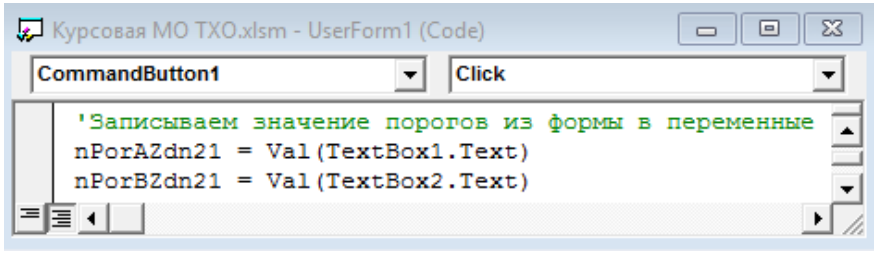


Рис. 21. Считывание порогов интегрирования

Далее код повторяет решение с выводом на лист Excel. Приводить его нет смысла. Все промежуточные вычисления выводятся на лист, окончательный расчет значения интеграла также выводится на лист. Дополнительно значение интеграла выводится в поле **TextBox3** на нашей форме. Фрагмент кода вывода значения переменной в текстовое поле на форме показан на рисунке 22. Результат работы формы показан на рисунке 23. Обратите внимание на то, что ответ отличается от предыдущих результатов в шестом и седьмом знаке после запятой.

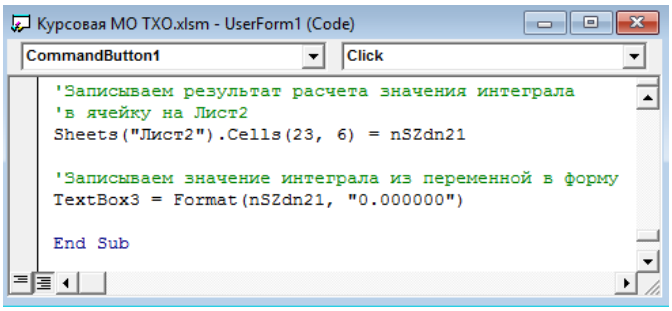


Рис. 22. Вывод значения переменной в текстовое поле формы

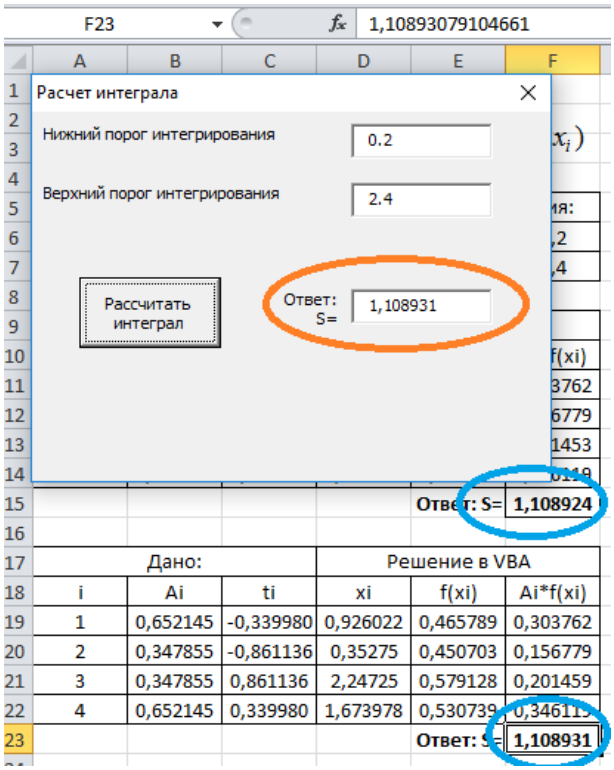


Рис. 23. Расчет определенного интеграла с помощью формы пользователя

2.3 РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Выполнение этого задания не требует новых знаний в программировании и выполняется самостоятельно, аналогично предыдущим примерам.

2.4 ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ

Язык программирования VBA дополняет возможности офисных программ. Для работы с графикой, как правило, используются встроенные средства используемой программы. Мы работаем с Excel, поэтому для построения графика будем использовать возможности Excel. То есть, мы будем строить график функции в Excel известным нам способом, но управлять этим процессом будем с помощью кода на VBA.

В этом задании необходимо продемонстрировать работу с макрорекордером и самостоятельно, с помощью рекомендованной литературы, объяснить значение операторов VBA, полученных в коде с помощью макрорекордера.

Запись макросов в Excel 2010 включается командой **Файл\Макросы\Запись_макроса**.

На рисунке 24 показан произвольный график, созданный на листе Excel. Действия по созданию графика (выбор ячеек, заполнение ячеек, запись в них формул, построение графика встроенными средствами и т.д.) записаны с помощью макрорекордера. Код программы, автоматически созданный при записи макроса, показан на рисунке 25. Необходимо самостоятельно проанализировать каждую строчку кода и построить подобным образом графики для своего варианта. Необходимо убрать лишние строки кода (или добавить недостающие строки) и написать комментарий каждой строке. график должен быть отформатирован. По осям должны быть отложены необходимые значения, график и оси подписаны.

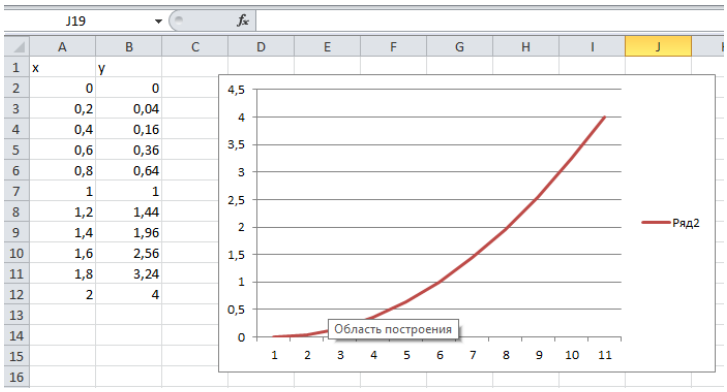


Рис. 24. Построение произвольного графика на листе Excel

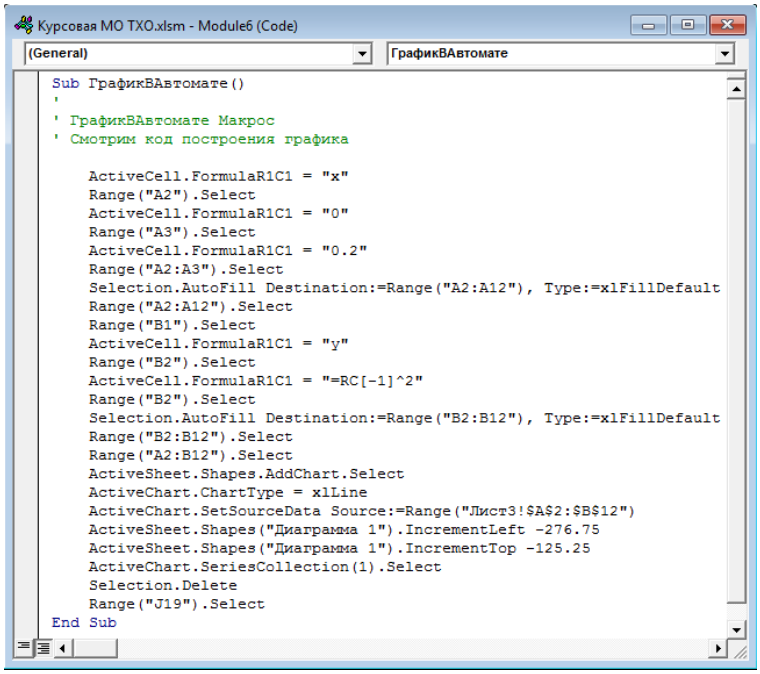


Рис. 25. Код макроса для построения произвольного графика на листе Excel

ЗАКЛЮЧЕНИЕ

Содержание курсовой работы отражает ход мысли и способности студента. Оформление работы демонстрирует отношение студента к процессу обучения, преподавателю и лично к себе. Все разделы работы последовательно раскрывают этапы достижения цели. В работе содержится только информация, непосредственно связанная с этапами работы. Все иллюстрации раскрывают содержание выполненных действий. Текстовые пояснения помогают студенту отвечать на вопросы на защите, а не порождают новые уточняющие вопросы. Выводы по работе содержат личные впечатления студента и констатируют достижение цели работы, а не перечисляют очевидные факты.

Удачи!

СПИСОК ЛИТЕРАТУРЫ

1. ИНФОРМАТИКА: Численное решение задач в пакетах Microsoft Excel, MathCAD и MatLab: Методические указания для самостоятельной работы / Национальный минерально-сырьевой университет «Горный»; Сост.: С.Ю. Кротова, А.Б. Маховиков, И.О. Онушкина. СПб, 2013. 63 с.
2. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления. Минск, 2001, 22 с. (Система стандартов по информ., библ. и изд. делу).
3. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. Минск, 1995, 26 с.
4. ГОСТ Р 7.0.5-2008. Библиографическая ссылка. Общие требования и правила составления. М.: Стандартинформ, 2005, 20 с.
5. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. М.: Стандартинформ, 2010, 22 с.
6. Информатика: Учебник для вузов / А.С. Грошев. – М.-Берлин: Директ-Медиа, 2015. – 484 с.
7. Томас Х. Кормен. Алгоритмы. Вводный курс – М.: «Вильямс», 2013. — 208 с.
8. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е издание – М.:«Вильямс», 2013. — 1328 с.
9. Михеев Р. Н. VBA и программирование в MS Office для пользователей. — СПб.: БХВ-Петербург, 2006. — 384 с.

СОДЕРЖАНИЕ

Учебное издание

Косарев Олег Валерьевич

ИНФОРМАТИКА. ЧИСЛЕННОЕ РЕШЕНИЕ ТИПОВЫХ ЗАДАЧ
С ПОМОЩЬЮ VBA

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Компьютерная верстка О.В. Косарева

Подписано в печать

Формат 60x84 1/16 Печать ризография. Гарнитура «Таймс».

Усл. печ. л. . Уч.-изд. л. . Тираж экз. Заказ № .

г. Санкт-Петербург

36