

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В.Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра информационных технологий

Утверждено
научно-методическим советом
университета

ОСНОВЫ VISUAL BASIC

Методические указания к выполнению
лабораторных работ по дисциплине "Информационные технологии"
для студентов, обучающихся по направлению подготовки
15.03.02 - Технологические машины и оборудование

Белгород 2015

УДК 004.434
ББК 32.973
О 75

Составитель *А.Ю. Стремнев*, канд. техн. наук, доцент
Белгородский государственный
технологический университет
им. В. Г. Шухова

О 75

Основы Visual Basic: Методические указания к выполнению лабораторных работ по дисциплине "Информационные технологии" для студентов, обучающихся по направлению подготовки 15.03.02 - Технологические машины и оборудование. – Белгород: Изд-во БГТУ, 2015. – 205 с.

Методические указания предназначены для приобретения студентами базовых навыков по работе в среде программирования Visual Basic for Application и содержат теоретический материал и задания к выполнению семи лабораторных работ.

Методические указания предназначены для студентов, обучающихся по направлению подготовки 15.03.02 - Технологические машины и оборудование.

УДК 004.434
ББК 32.973

© Белгородский государственный
технологический университет
(БГТУ) им. В.Г. Шухова, 2015

СОДЕРЖАНИЕ

Содержание.....	3
Лабораторная работа № 1. Знакомство со средой программирования Visual Basic	7
Цель работы	7
Теоретические сведения	7
Задание	15
Состав отчета	41
Лабораторная работа № 1. Знакомство со средой программирования Visual Basic (для MS Office 2007-2010)	43
Цель работы	43
Теоретические сведения	43
Задание	51
Состав отчета	77
Лабораторная работа № 2. Переменные. Операторы. Встроенные функции.....	79
Цель работы	79
Теоретические сведения	79
Переменные и константы	79
Встроенные операторы и функции.....	88
Функция MsgBox.....	96
Функция InputBox	97
Пример.....	99
Решение	100
Задание	102
Состав отчета	109
Контрольные вопросы	109
Лабораторная работа № 3. Условные операторы.....	110
Цель работы	110

Теоретические сведения	110
Операторы If...Then и If...Then...Else	110
Оператор Select Case	116
Пример 1.....	117
Решение примера 1.....	117
Пример 2.....	120
Решение примера 2.....	120
Задание	122
Состав отчета	128
Контрольные вопросы	129
Лабораторная работа № 4. Операторы циклов	130
Цель работы	130
Теоретические сведения	130
Оператор For...Next	130
Оператор Do...Loop	132
Пример.....	134
Решение	134
Задание	138
Состав отчета	142
Контрольные вопросы	142
Лабораторная работа № 5. Массивы.....	143
Цель работы	143
Теоретические сведения	143
Пример.....	146
Решение	146
Задание	150
Состав отчета	150
Контрольные вопросы	151
Лабораторная работа № 6. Процедуры и функции	152
Цель работы	152

Теоретические сведения	152
Процедуры и функции	152
Область видимости переменных	158
Пример.....	161
Решение	162
Задание	166
Состав отчета.....	166
Контрольные вопросы	167
Лабораторная работа № 7. Формы и элементы управления	168
Цель работы	168
Теоретические сведения	168
Формы: свойства, методы, обработка событий	168
Надпись (Label).....	173
Поле ввода (TextBox).....	174
Кнопка (CommandButton).....	175
Флажок (CheckBox).....	176
Переключатель (OptionButton).....	177
Список (ListBox).....	178
Полоса прокрутки (ScrollBar)	179
Картинка (Image)	181
Пример.....	182
Решение	183
Задание	189
Состав отчета.....	190
Контрольные вопросы	190
Лабораторная работа № 8. Численные методы	192
Цель работы	192
Теоретические сведения	192
1. Решение уравнений.....	192
Метод Ньютона	192

Метод деления отрезка пополам	193
2. Решение систем уравнений	194
3. Расчет определенных интегралов.....	196
Метод прямоугольников.....	197
Метод трапеций.....	198
Метод Симпсона (парабол)	199
Задание	200
Состав отчета	204
Контрольные вопросы	205

ЛАБОРАТОРНАЯ РАБОТА № 1. ЗНАКОМСТВО СО СРЕДОЙ ПРОГРАММИРОВАНИЯ VISUAL BASIC

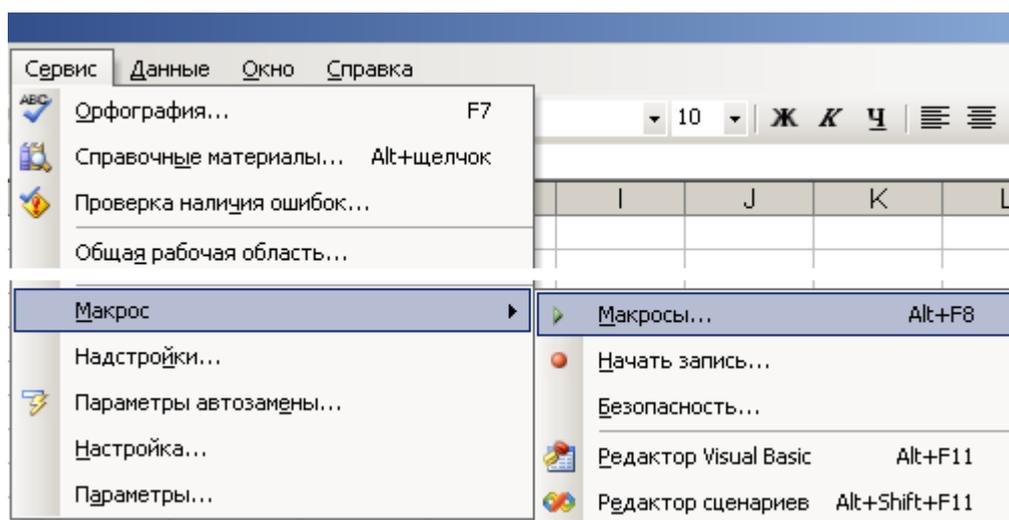
Цель работы

Получить представление о принципах работы в среде программирования Visual Basic (VB).

Теоретические сведения

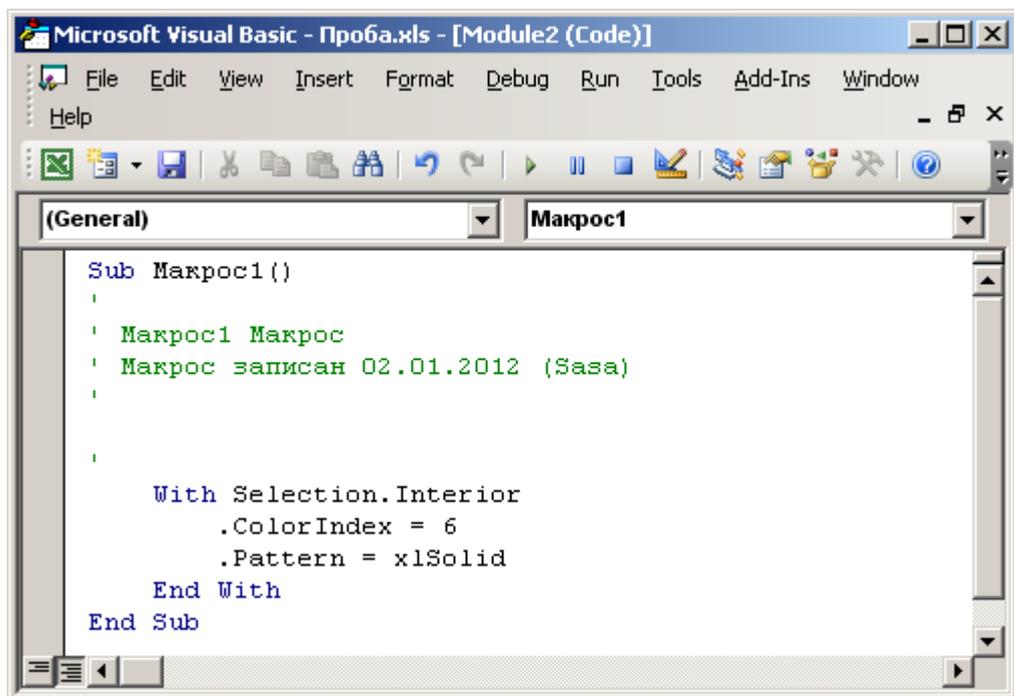
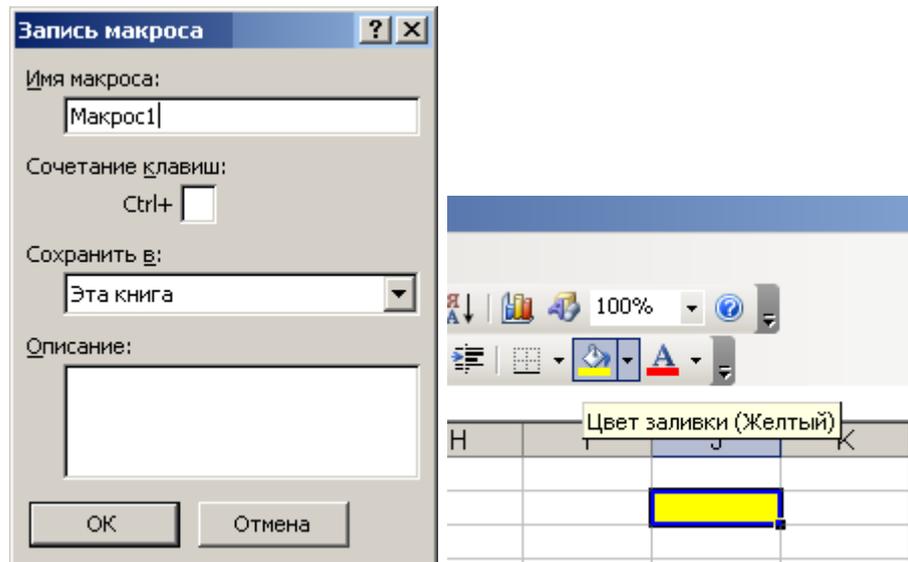
Разработка специалистом конкретной предметной области программных продуктов может быть оправдана при использовании языков программирования, имеющих не только широкие функциональные возможности, но и синтаксис, близкий к естественному математическому и логическому представлению объектов рассматриваемой области. К таким языкам относится VB. В виде интегрированного средства (Visual Basic for Application (VBA)) он входит в пакет MS Office, а многие программы позволяют использовать собственные объекты с помощью библиотек VB.

Для работы в VBA в любом приложении MS Office предназначена группа команд *Service (Сервис) > Macros (Макрос)*.

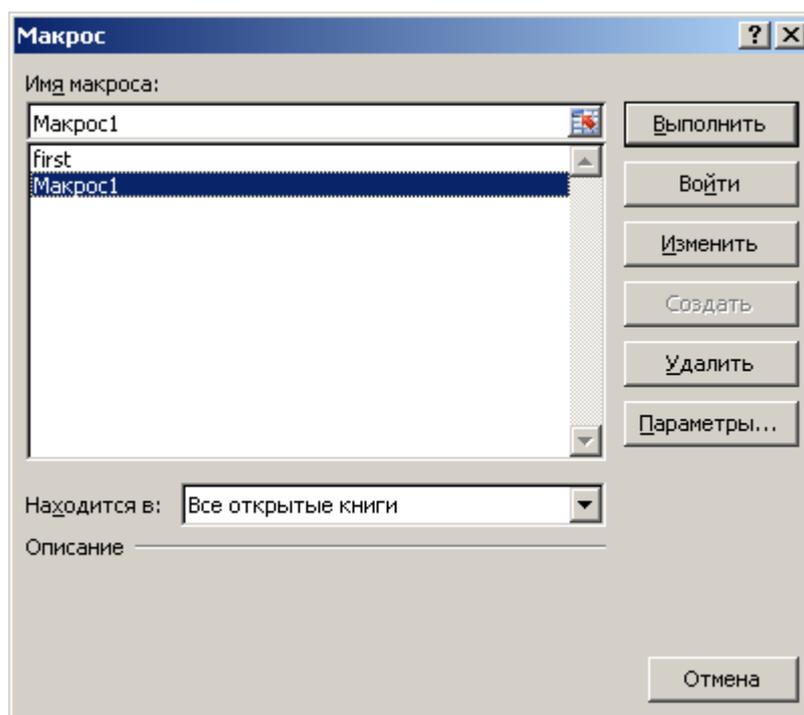


Команды: *Macros (Макросы)*, *Record (Начать запись)* и *Security (Безопасность)* предназначены для автоматизации работы с программными приложениями на VB. *Record (Начать запись)* – команда, позволяющая

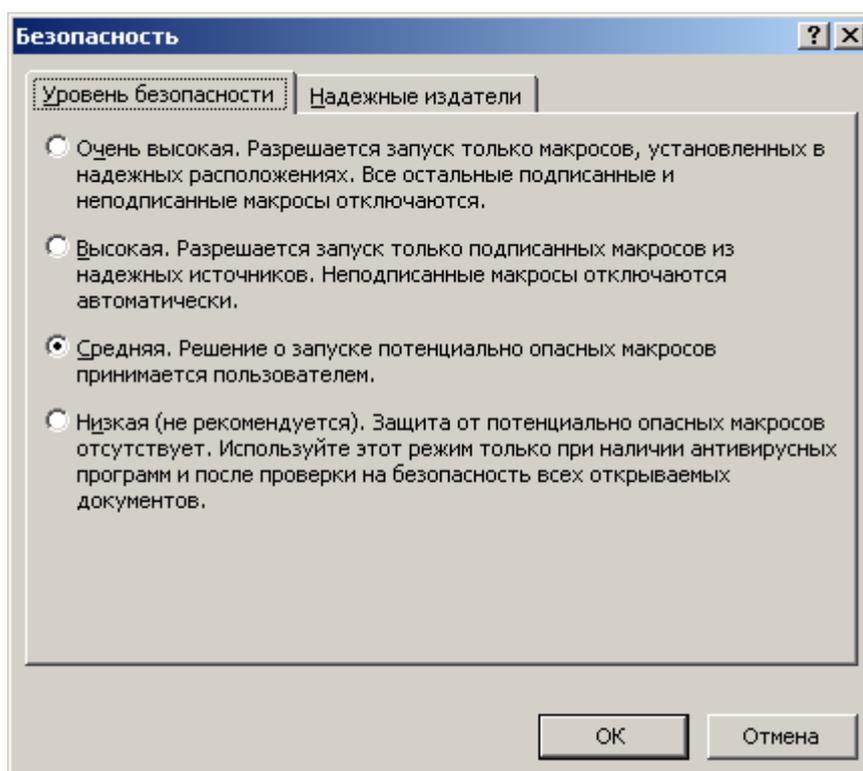
записать последовательность действий пользователя в приложении MS Office в виде программного кода VB – макроса.



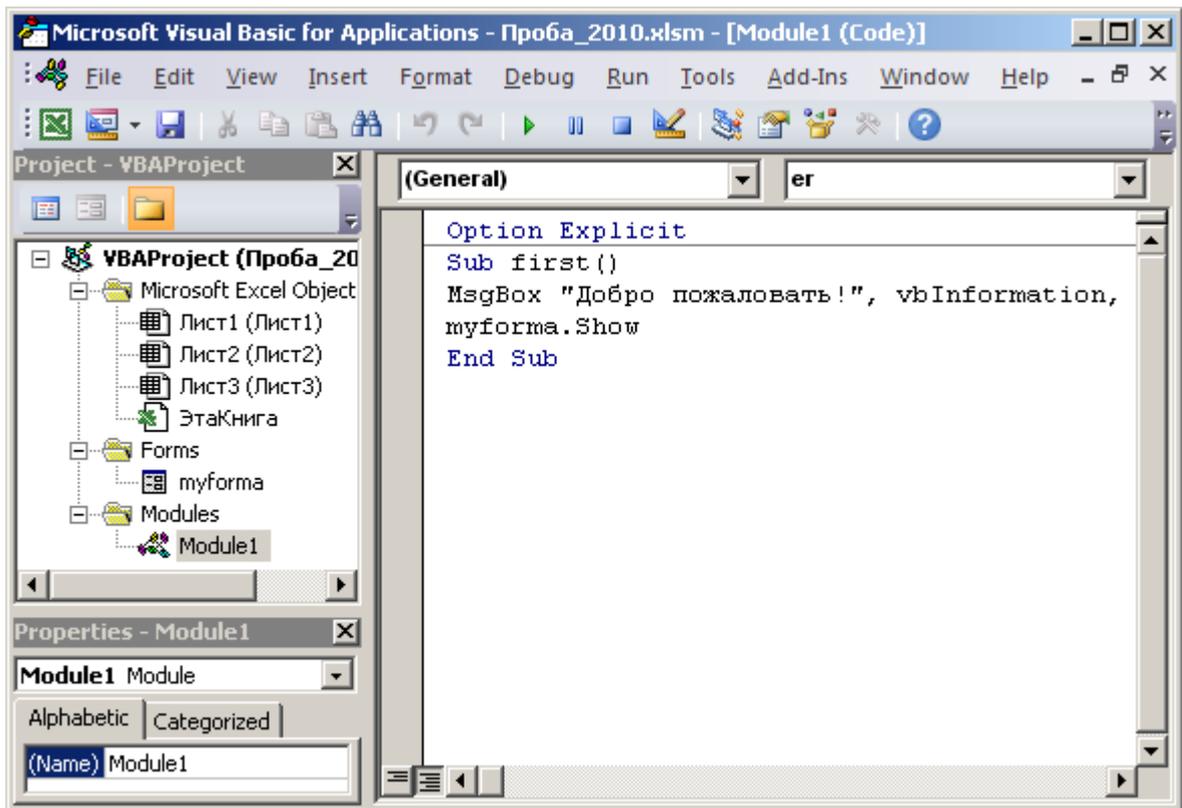
Записанные макросы можно использовать (запускать) многократно (*Macros (Макрос) > Macros (Макросы) > Run (Выполнить)*) и редактировать (*Macros (Макрос) > Macros (Макросы) > Edit (Изменить)*).



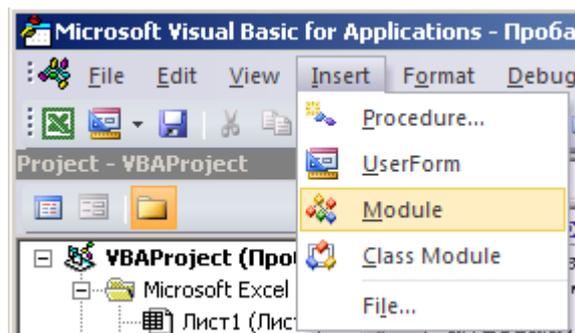
Команда **Security (Безопасность)** позволяет установить уровень защиты от запуска макросов (н.п., источник содержащего их файла неизвестен), так как некоторые из них могут выполнять опасные (нежелательные) действия.



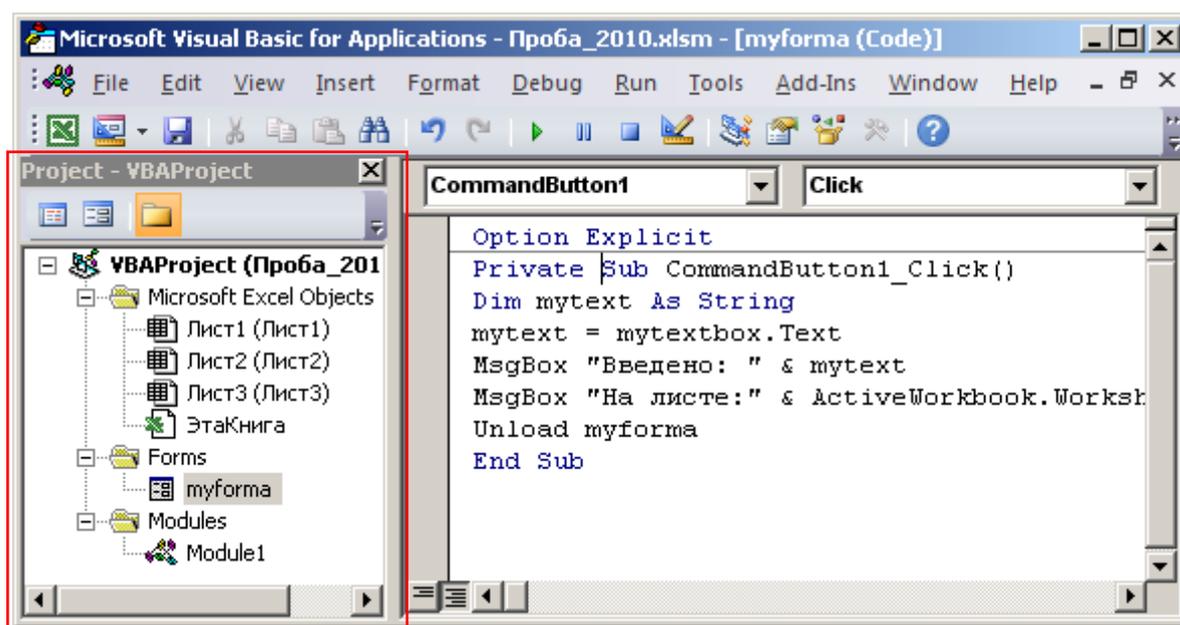
Команда **Visual Basic Editor (Редактор Visual Basic)** открывает оболочку для создания и редактирования программ VB (в т.ч. макросов).



Оболочка VB позволяет создавать программные проекты на базе модулей (*Insert (Вставка) > Module (Модуль)*), электронных форм (*Вставка (Вставка) > User Form (Пользовательская форма)*) и модулей пользовательских объектов (*Insert (Вставка) > Class Module (Модуль класса)*).



Контролировать состав проекта VB и осуществлять навигацию между всеми его компонентами можно в окне проекта (*View (Вид) > Project Explorer (Окно проекта)*), снабженного кнопками отображения объектов  (*View Object*) для форм и программного кода  (*View Code*) для форм и модулей.



Окно проекта (Project Explorer)

Модуль – это лист с текстом программы, вставленный в документ MS Office (записанные макросы добавляются именно в модули). Программный проект VB может состоять из нескольких модулей. Модули могут находиться в разных документах MS Office.

Структура программы в модуле VB следующая:

- 1) ключевое слово – тип программного фрагмента: функция, процедура, объявление переменных, объявление типа данных, объявление свойств;
- 2) имя программного фрагмента;
- 3) опции программного фрагмента: параметры, переменные;
- 4) объявления и инициализация переменных для функций и процедур;
- 5) программный код, реализующий необходимый пользовательский алгоритм;
- 6) завершение программы: выходные результирующие значения, ключевое слово.

Пример программы VB в форме процедуры, выводящей на экран текстовое сообщение (' – символ-метка комментария):

Sub qwerty()

'Sub – ключевое слово для программы-процедуры

' по имени "qwerty", () – без входных параметров

Dim usertitle As String

'Dim – объявление переменной "usertitle"

' типа String (строка)

usertitle = "qwerty1234"

'инициализация переменной

' "usertitle" значением "qwerty1234"

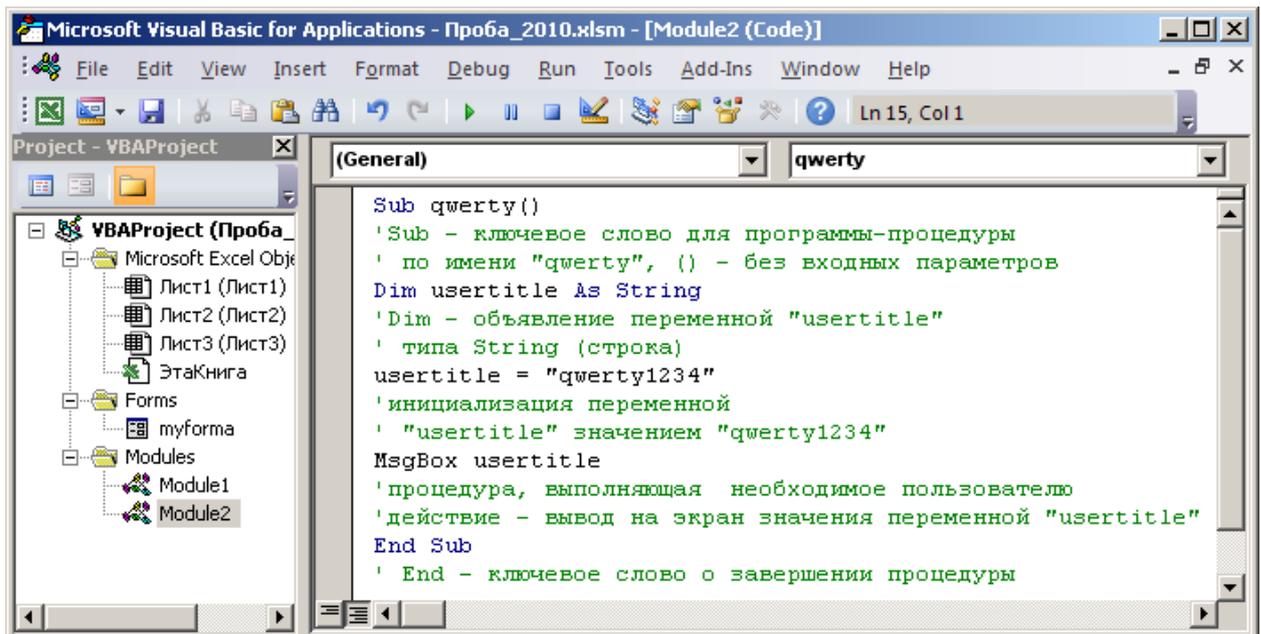
msgBox usertitle

'процедура, выполняющая необходимое пользователю

'действие – вывод на экран значения переменной "usertitle"

End Sub

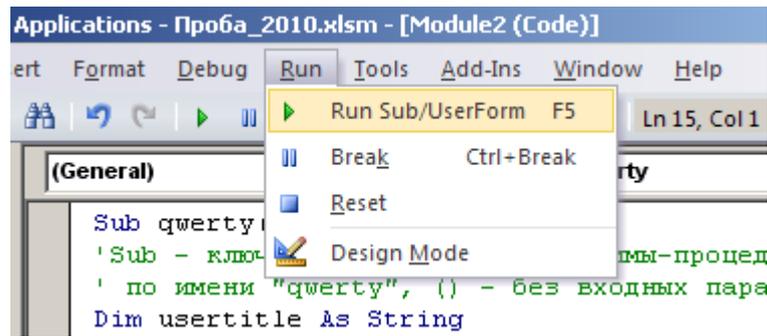
' End – ключевое слово о завершении процедуры



В оболочке VB MS Office имеется команда для сохранения программы в составе документа соответствующего приложения (**File (Файл) > Save... (Сохранить...)**) или отдельного компонента – модуля VB для последующего использования (**File (Файл) > Export File (Экспорт)**).

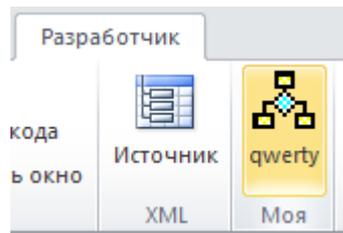
Запуск программы производится несколькими методами:

1) в оболочке VB команда **Run (Запуск) > Run Sub/UserForm (Запуск Процедуры/Пользовательской формы)**;



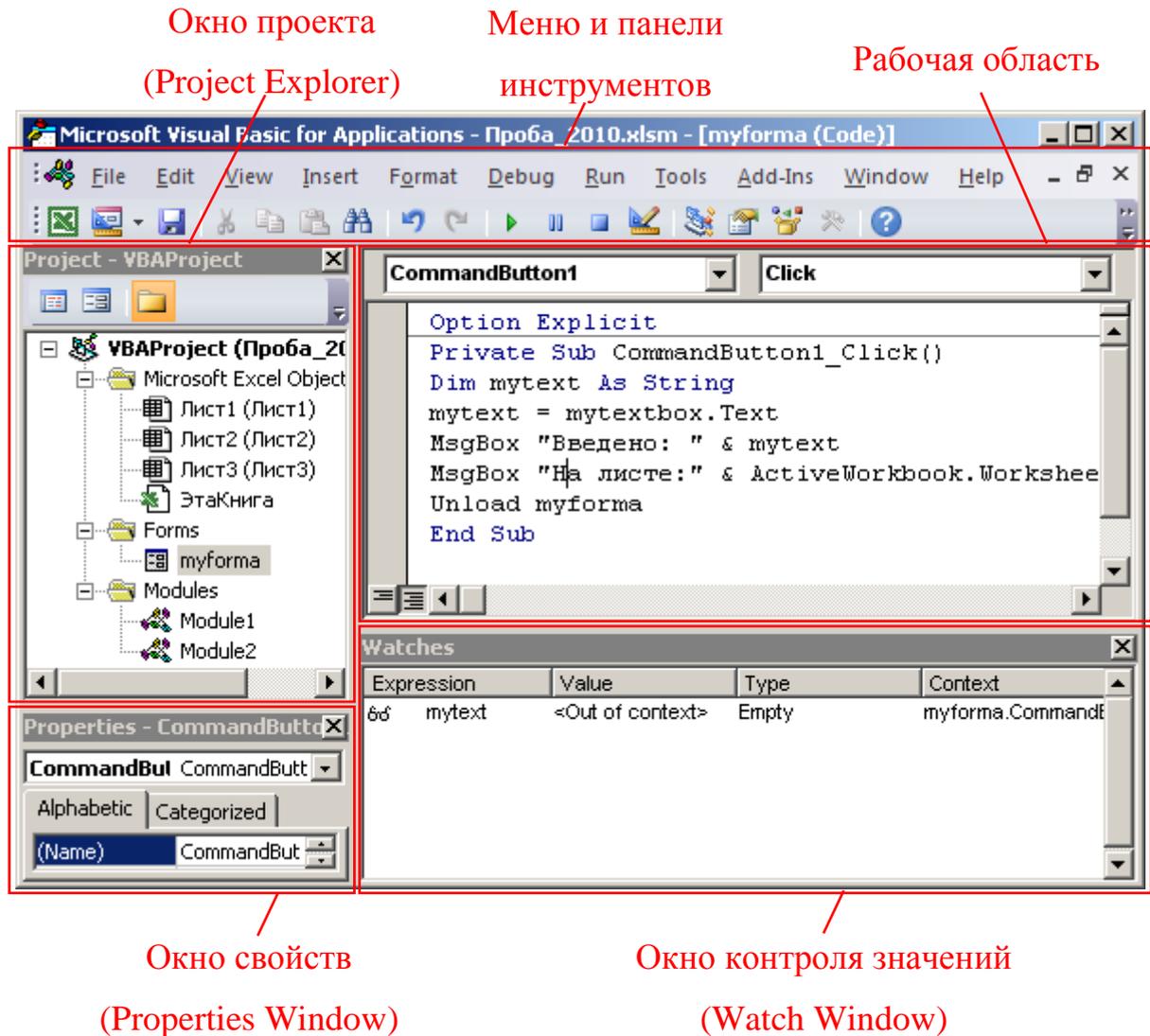
2) в оболочке соответствующего приложения MS Office: *Service (Сервис) > Macros (Макрос) > Macros (Макросы) > Run (Выполнить)*;

3) в оболочке соответствующего приложения MS Office по нажатию кнопки панели инструментов: *View (Вид) > Toolbars (Панели инструментов) > Edit (Настройка) > New... (Создать...), Commands (Команды) > Macros (Макросы) > ...*



Оболочка VB состоит из следующих основных частей:

- 1) панель меню (содержит все команды среды программирования VB);
настраиваемые панели инструментов (тематические наборы часто используемых команд);
- 2) рабочая область (ввод и редактирование текста программы);
- 3) вспомогательные окна: свойств объектов (*Properties Window*), состава программного проекта (*Project Explorer*), доступных программных компонентов (*Object Browser*), отладка (*Immediate Window*), контроля значений при выполнении программы (*Watch Window*).



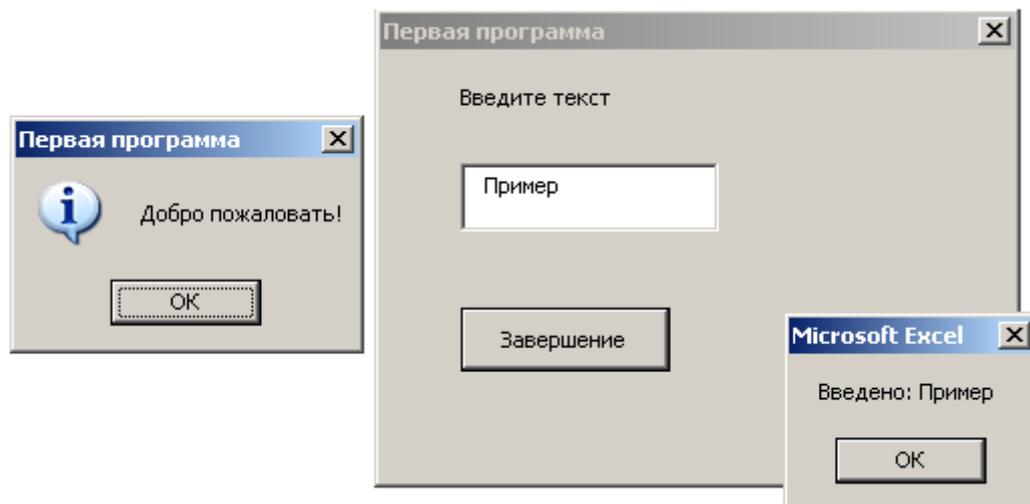
Панель меню содержит подменю:

- 1) **File (Файл)**: сохранение, экспорт, импорт программных компонентов, печать форм и текстов программных модулей, выход в приложение-контейнер (компонент MS Office);
- 2) **Edit (Редактирование)**: команды редактирования текста программ (операции с буфером обмена, возврат действий, табулирование текста, закладки, контекстные справки и шаблоны синтаксиса);
- 3) **View (Вид)**: переключение между окнами кода и объектов (форм), переход в окно приложения-контейнера, отображение структуры проекта VB (модулей, форм) и доступных программных библиотек и их компонентов, настройка панелей инструментов оболочки VB, отображение окон отладки (переменных, операторов);

- 4) **Insert (Вставка)**: добавление в программу процедур, функций, свойств, модулей, форм, файлов других проектов;
- 5) **Format (Формат)**: редактирование форм (расположение и размеры объектов);
- 6) **Debug (Отладка)**: проверка синтаксиса программ, выполнение программ по шагам (строкам кода), по точкам прерывания (остановки), просмотр текущих значений выражений при выполнении (контрольные значения);
- 7) **Run (Запуск)**: запуск выполнения программных модулей или диалогов, прерывание и сброс выполнения, переход в режим разработки;
- 8) **Tools (Инструменты)**: работа с макросами, загрузка дополнительных программных библиотек, настройка параметров оболочки и проекта VB;
- 9) **Add-Ins (Надстройки)**: добавление системных объектов в проект;
- 10) **Window (Окно)**: управление расположением окон VB;
- 11) **Help (Помощь)**: справка по синтаксису VB.

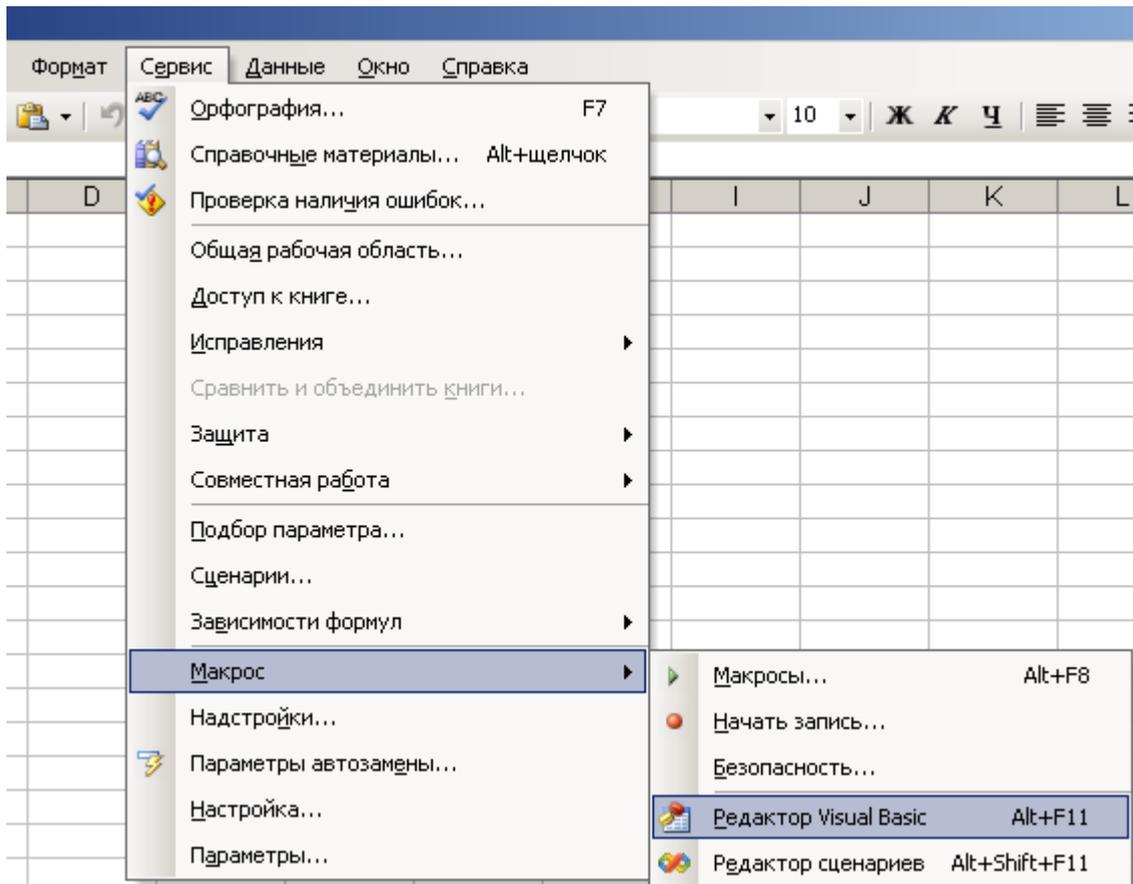
Задание

Итогом выполнения этой работы должна быть программа на VB, запускаемая в созданном пользователем файле электронных таблиц MS Excel. Программа будет выводить на экран сообщение о начале своей работы, а затем выводить диалоговое окно с полем ввода и кнопкой.

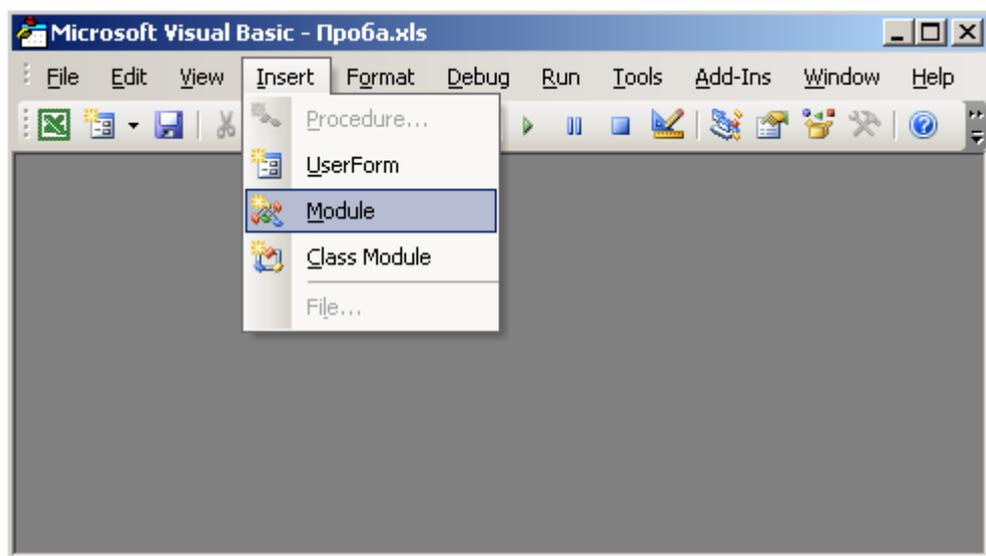


При нажатии кнопки программа отобразит значение, введенное пользователем в поле ввода, и закроет диалоговое окно.

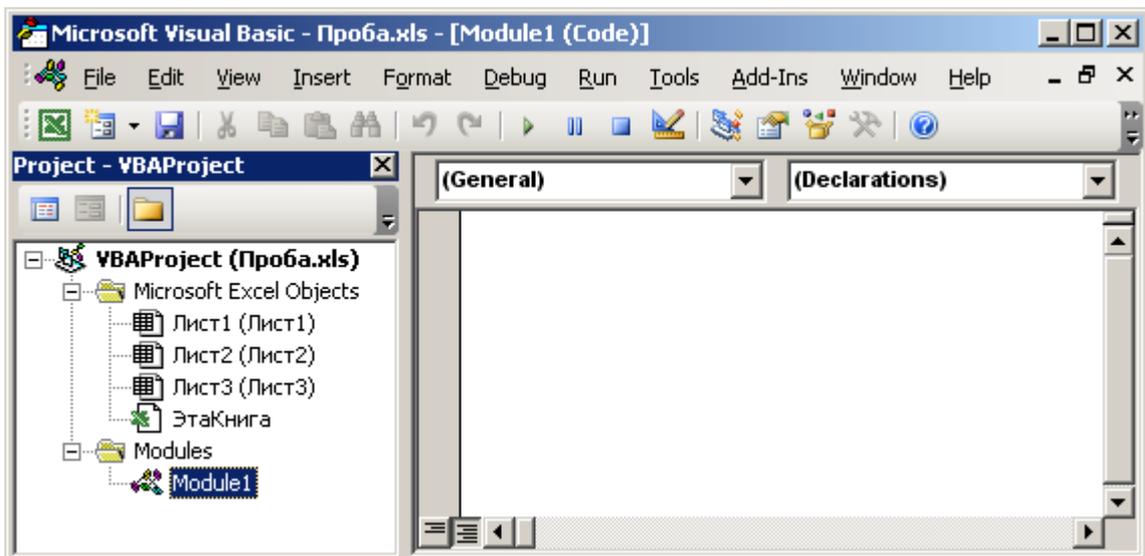
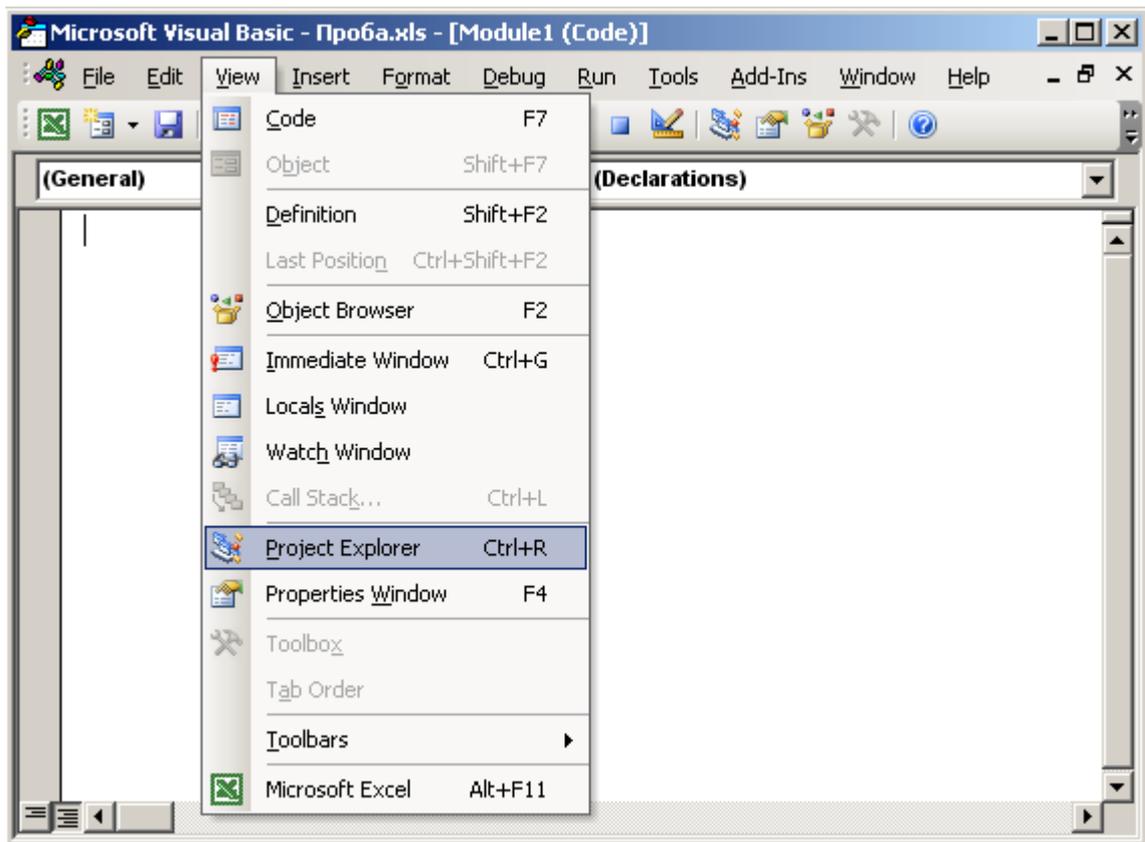
1. Открыть MS Excel, создать и сохранить электронную таблицу.
2. Запустить среду VBA: *Сервис > Макрос > Редактор Visual Basic*.



3. Добавить программный модуль VB: *Вставка (Insert) > Module*.



4. Открыть (если оно скрыто) окно проекта (*Project-VBA Project*): *Вид (View) > Окно проекта (Project Explorer)* и найти добавленный модуль в структуре текущего файла *Excel (VBAProject (...xls) > Модули (Modules) > Модуль1 (Module1)*.



5. Дважды щелкнуть на имени модуля в окне проекта и в открывшемся окне (окне программы) набрать текст процедуры:

Option Explicit

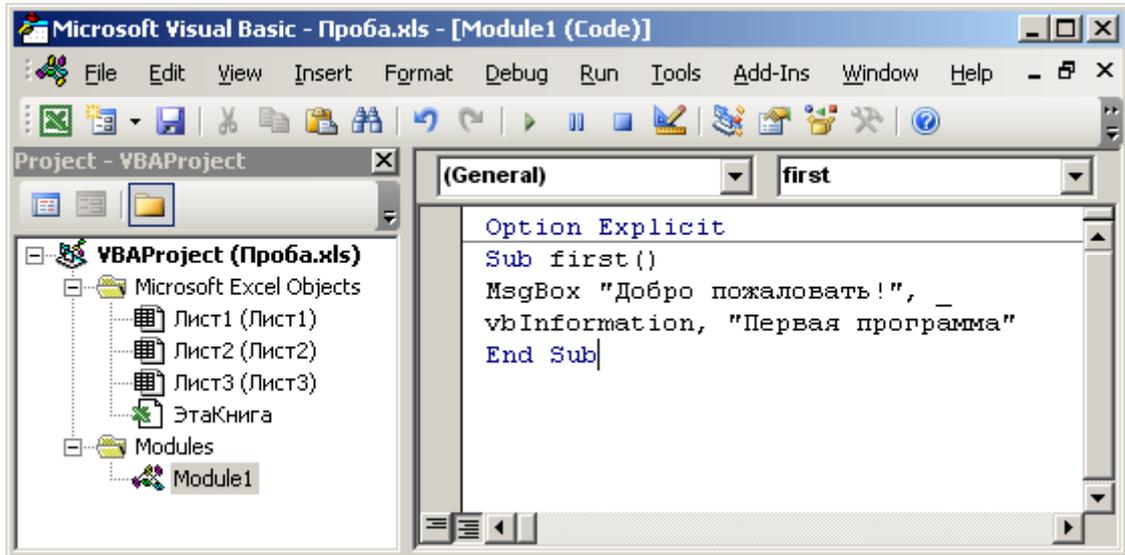
' Инструкция для обязательного объявления переменных

Sub first()

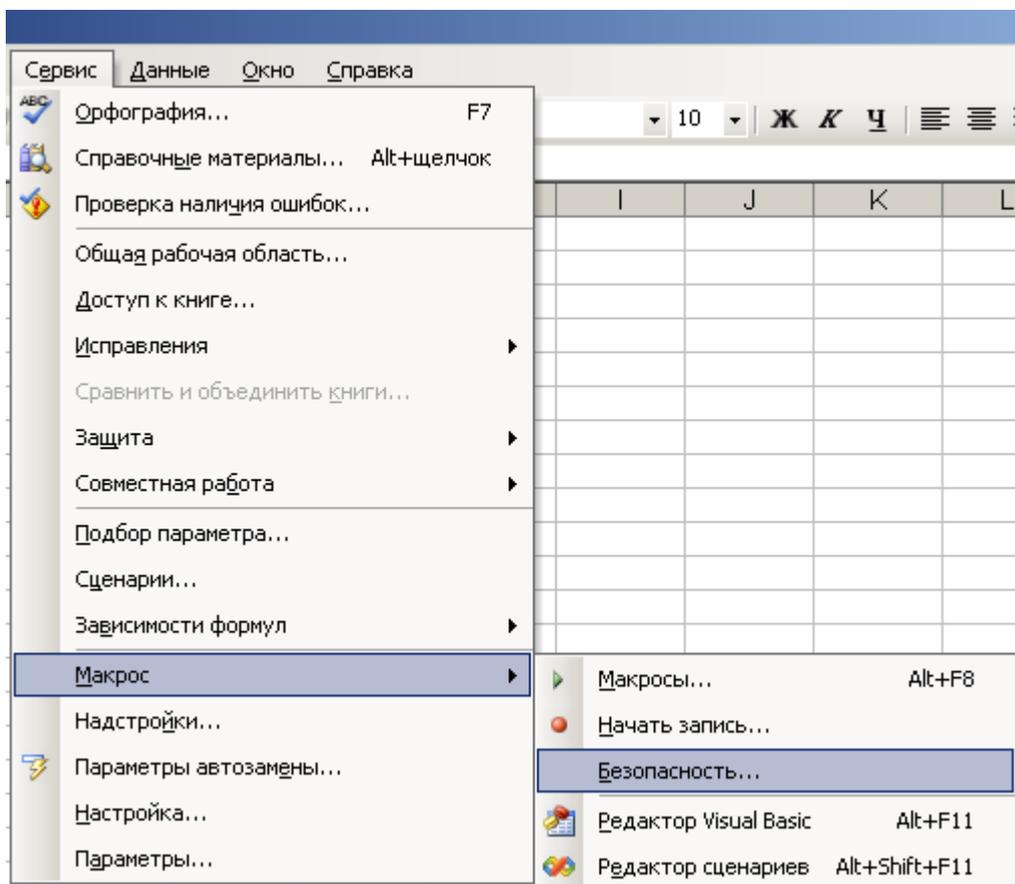
' Начало процедуры-программы по имени first

MsgBox "Добро пожаловать!", vbInformation, "Первая программа"

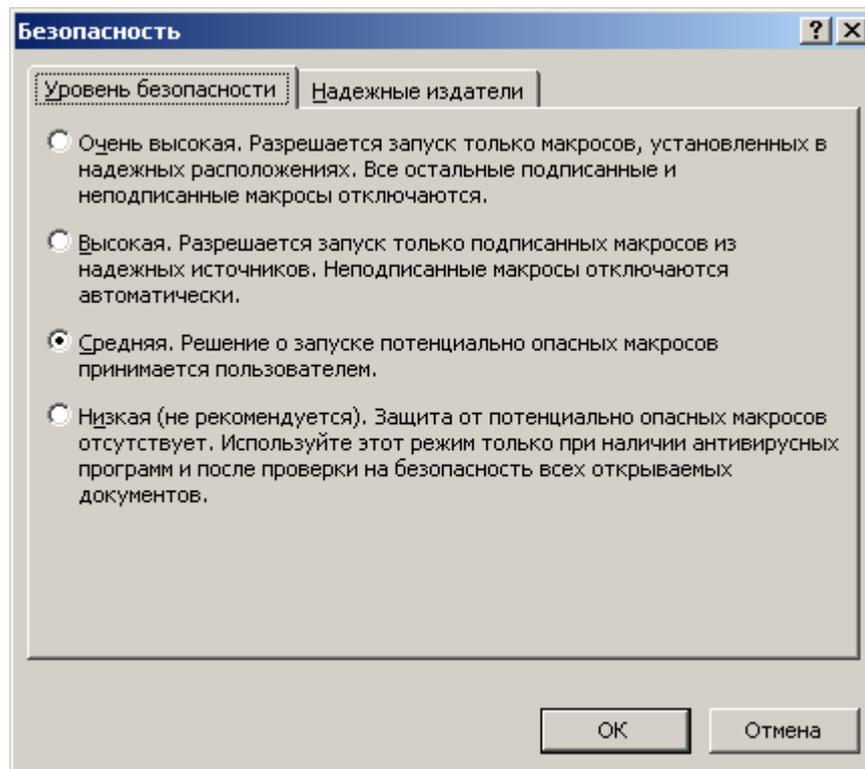
- ' Вывод окна-сообщения с текстом "Добро пожаловать"
- ' и заголовком "Первая программа"
- End Sub
- ' Завершение процедуры-программы по имени first



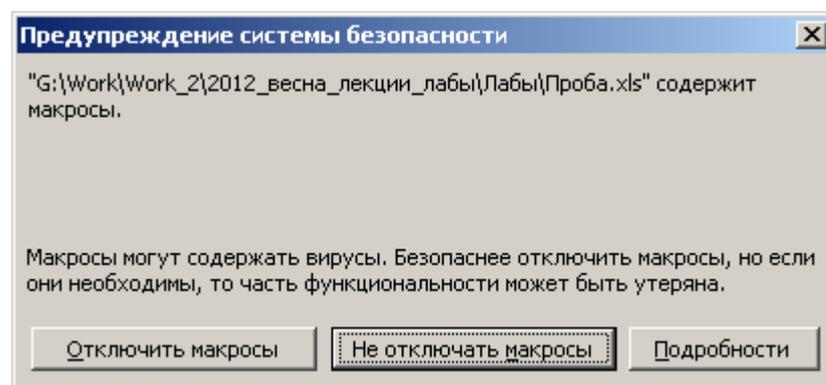
6. Перейти в MS Excel командой *Вид (View) > Microsoft Excel* и проверить уровень безопасности при запуске программ-макросов командой *Сервис > Макрос > Безопасность*.



Если установлен уровень *Средний* (*Решение о запуске потенциально-опасных макросов принимается пользователем*), нажать **ОК** и сохранить файл командой **Файл > Сохранить**.

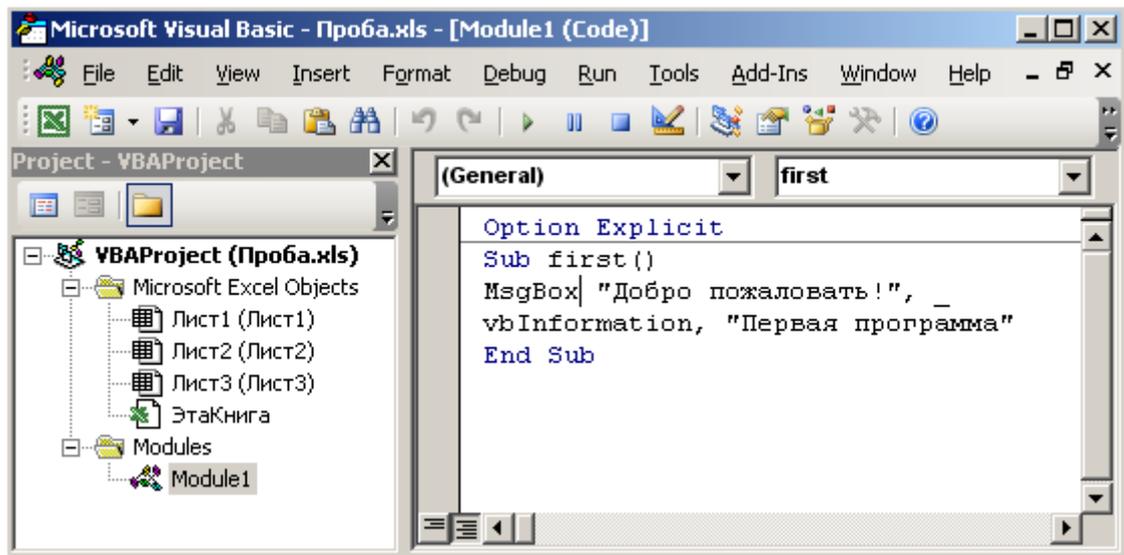


Если режим безопасности *Средний* не установлен, установить его, нажать **ОК**, а затем сохранить файл (**Файл > Сохранить**), закрыть его (**Файл > Закр~~ыть~~**) и повторно открыть (**Файл > Откр~~ыть~~** или **Файл >** первый файл в списке последних открытых). На *Предупреждение системы безопасности* при открытии файла отвечать **Не отключать макросы**.

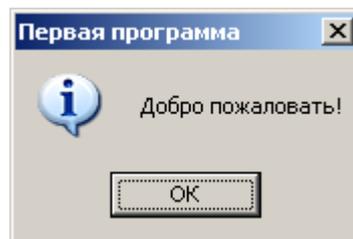
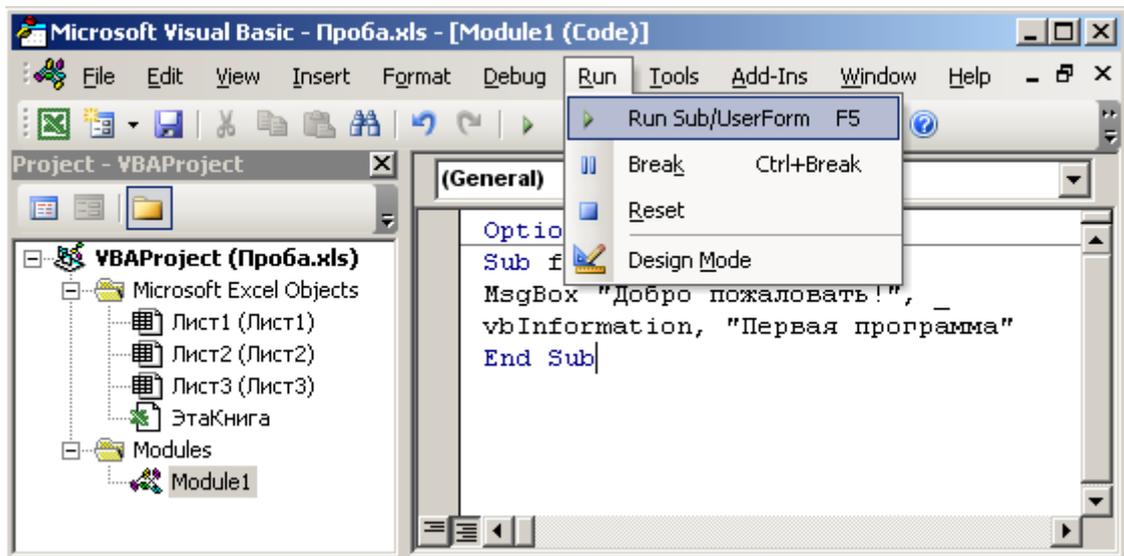


7. Открыть среду VBA (**Сервис > Макрос > Редактор Visual Basic**) и перейти в модуль **Module1**, щелкнув по нему в *Окне проекта* (**Project**

Explorer). Установить курсор в любом месте процедуры **first** (**Sub first()...End Sub**).

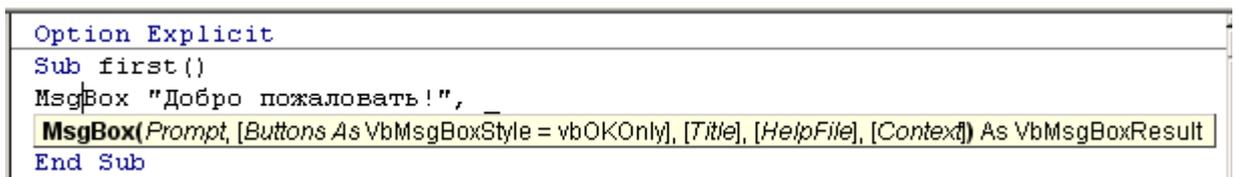
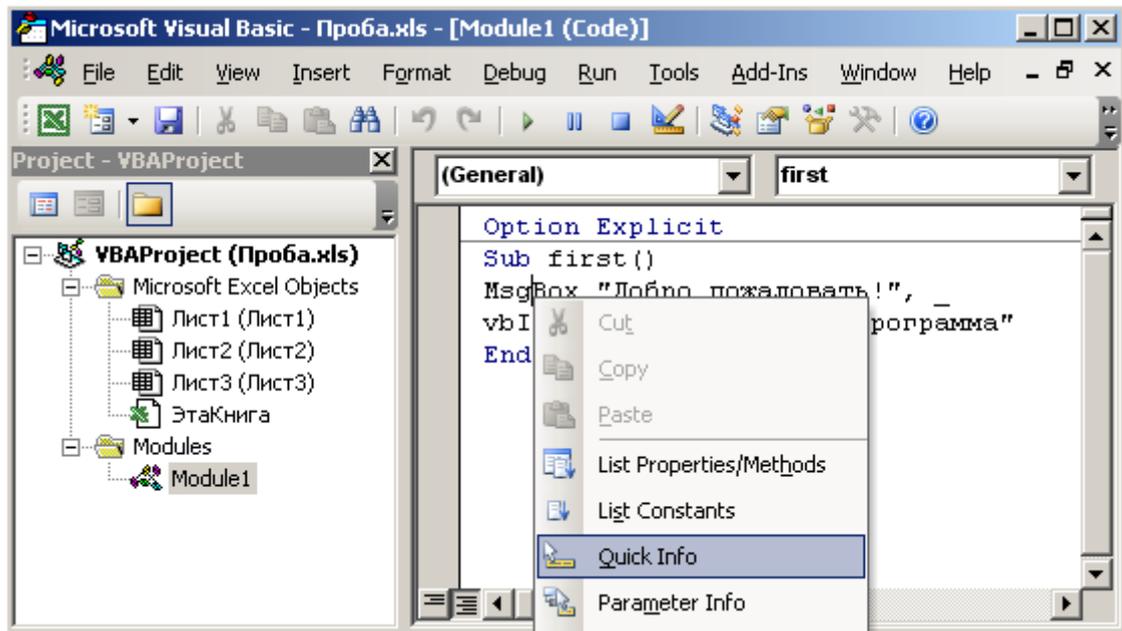


8. Сохранить файл (*Файл (File) > Сохранить... (Save...)*) и запустить модуль с процедурой **first** на выполнение командой: *Запуск (Run) > Запуск подпрограммы (Run Sub или F5)*.

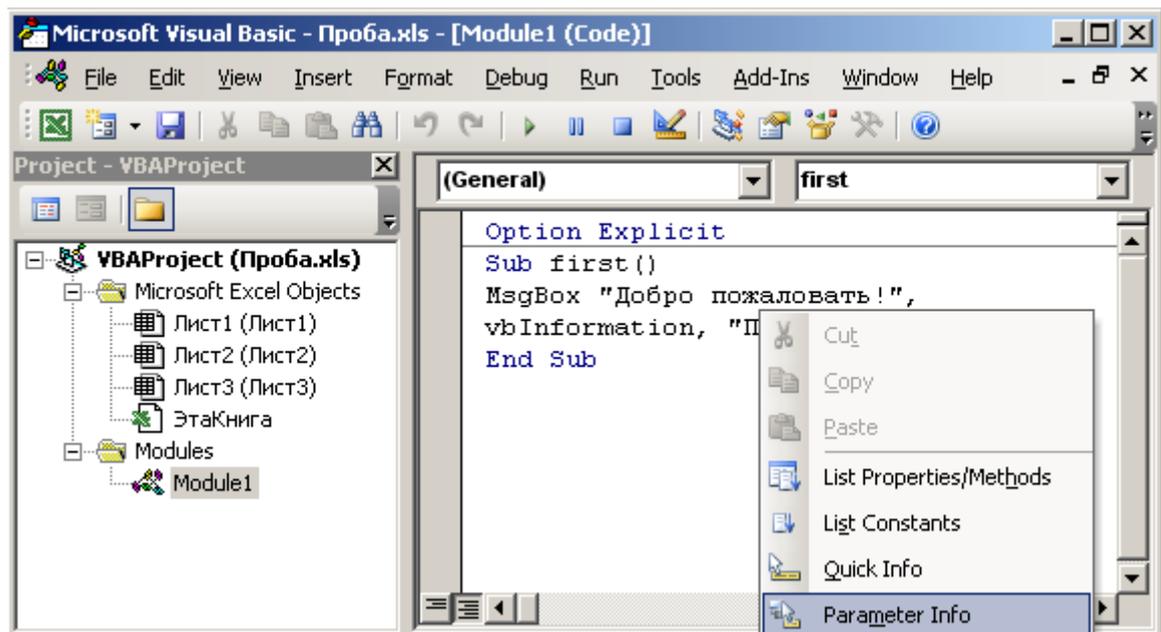


9. После завершения программы (нажатие *ОК* в окне *Первая программа*) в окне модуля установить курсор на ключевом слове **MsgBox** и выполнить

команду: *Правка (Edit) > Сведения (Quick Info)* – информация о синтаксисе функции или процедуры.

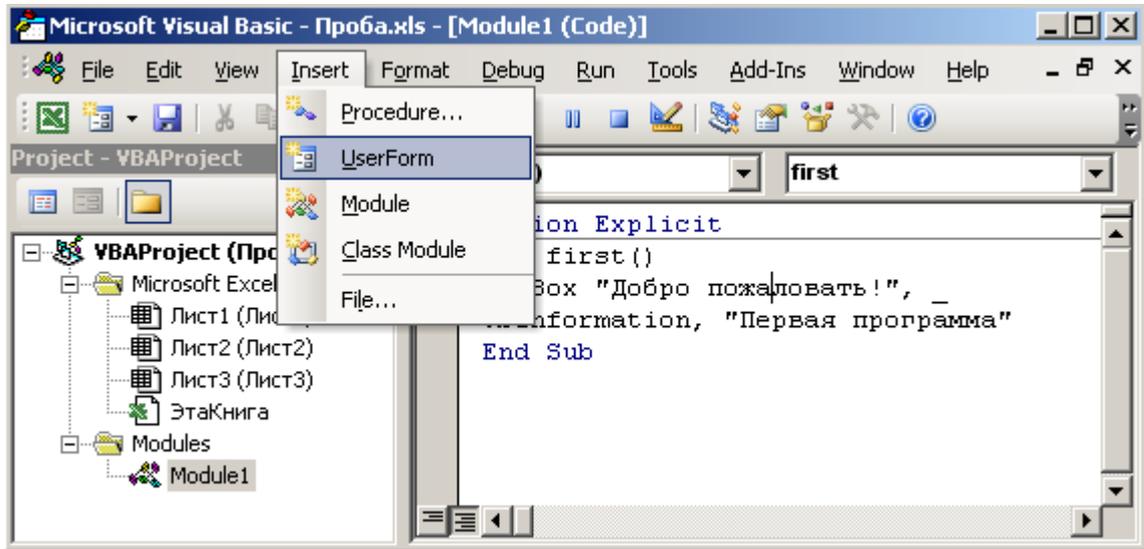


10. Установить курсор на тексте **"Добро пожаловать"** и выполнить команду: *Правка (Edit) > Параметры (Parameter Info)* – информация о текущем параметре функции или процедуры.

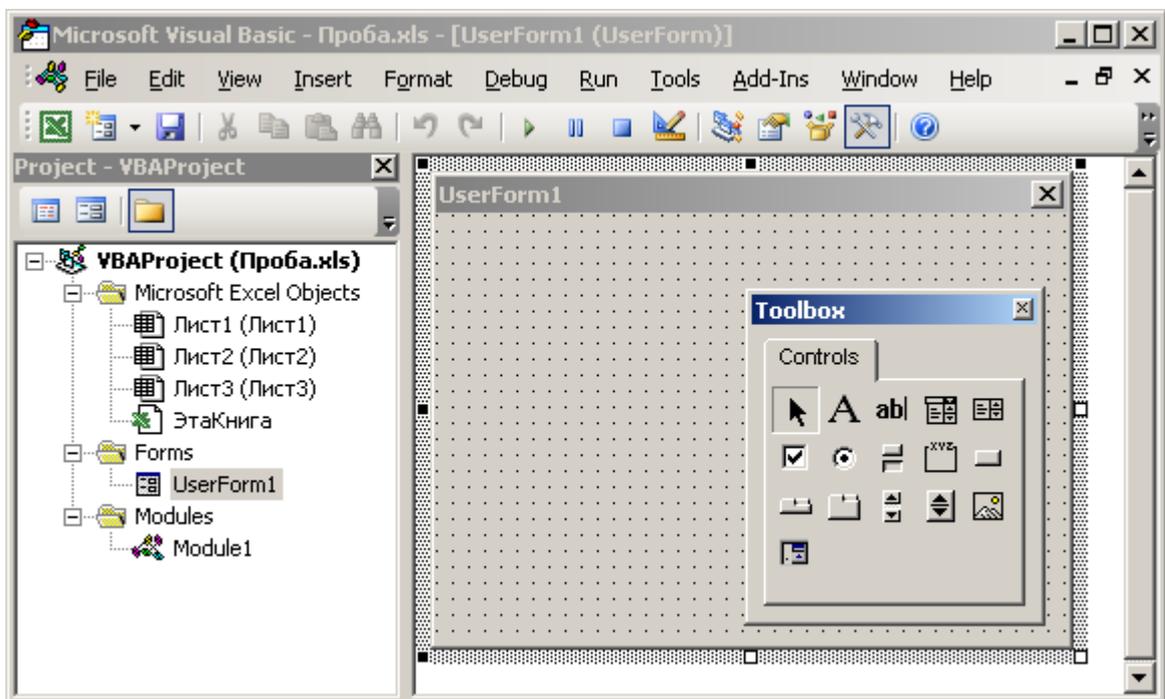


```
Option Explicit
Sub first()
MsgBox "Добро пожаловать!", _
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
End Sub
```

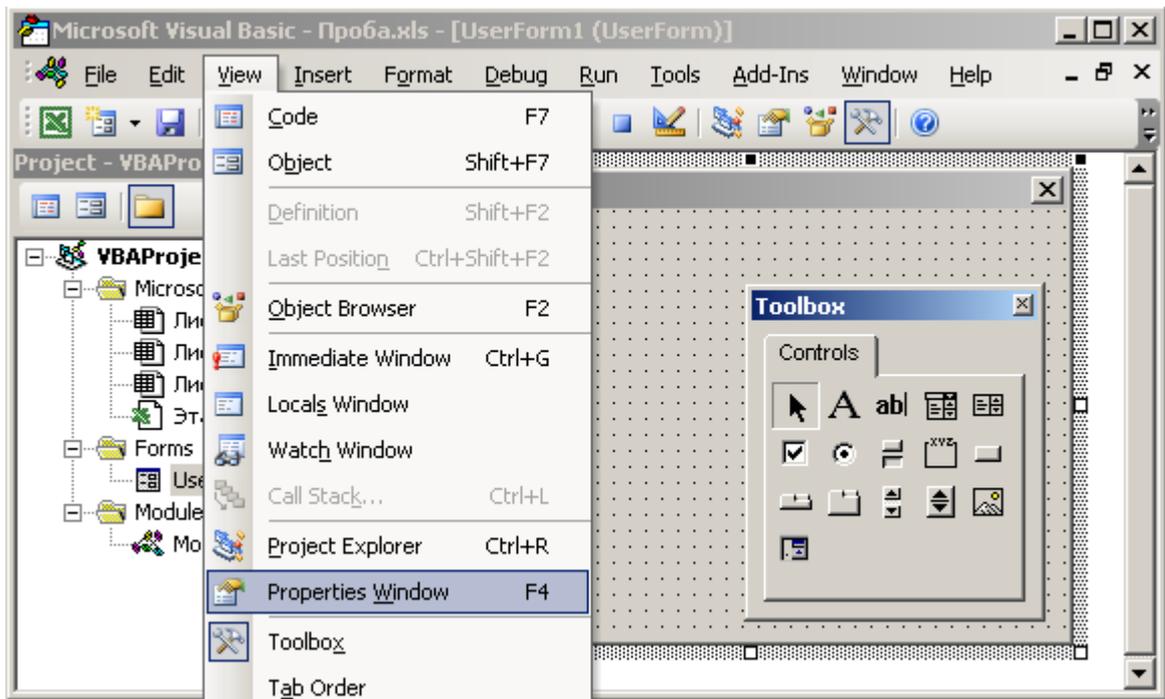
11. Добавить к проекту окно формы командой: **Вставка (Insert) > UserForm**.



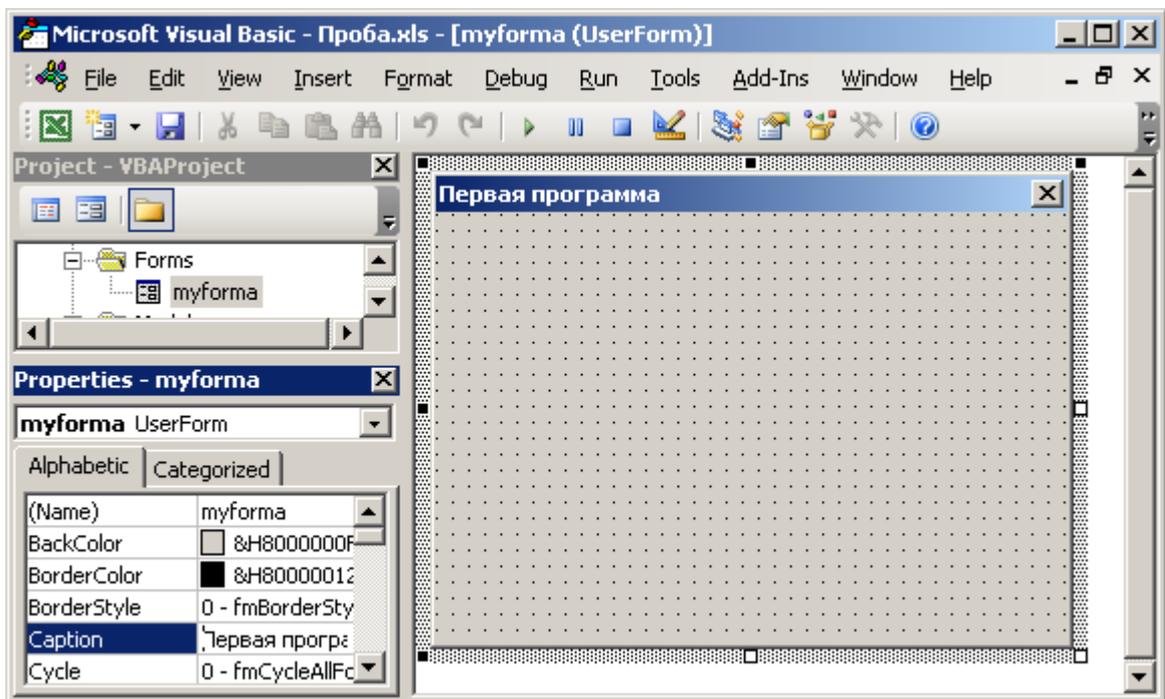
В проект будет добавлена новая форма, окно редактирования которой выводится на экран. В окне проекта будет добавлена группа **Формы (Forms)** с новой формой **UserForm1**.



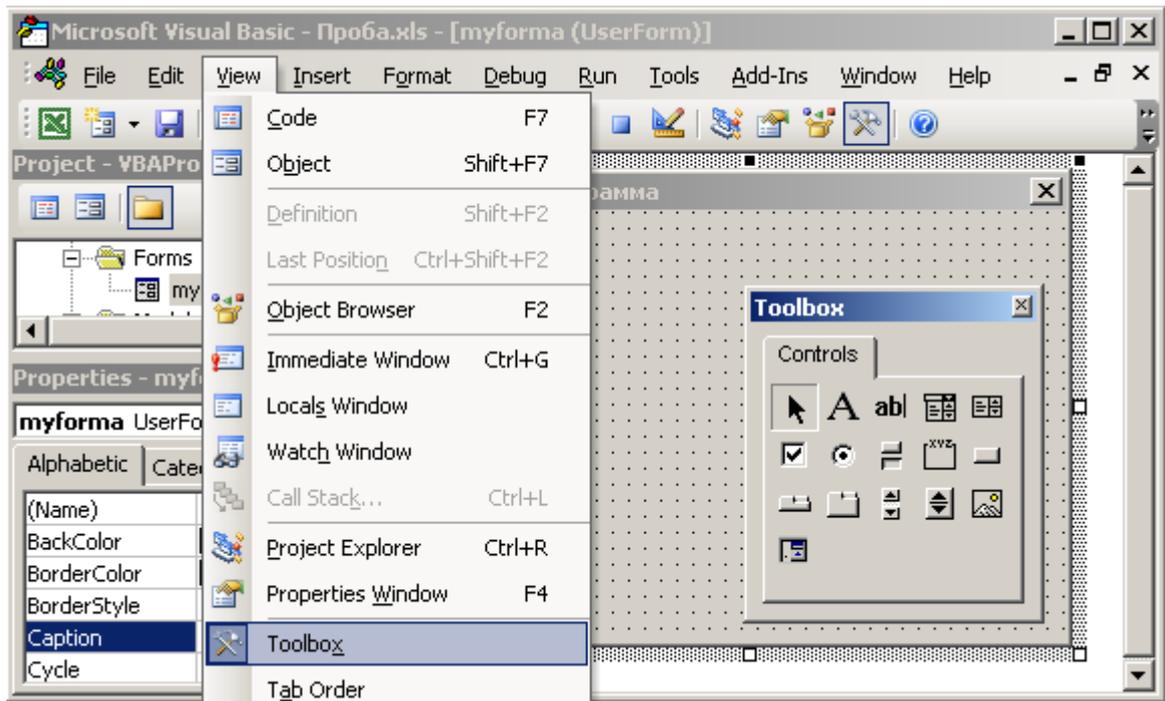
12. Открыть (если оно скрыто) окно свойств (**Properties**): **Вид (View) > Окно свойств (Properties Window)** для добавленной формы.



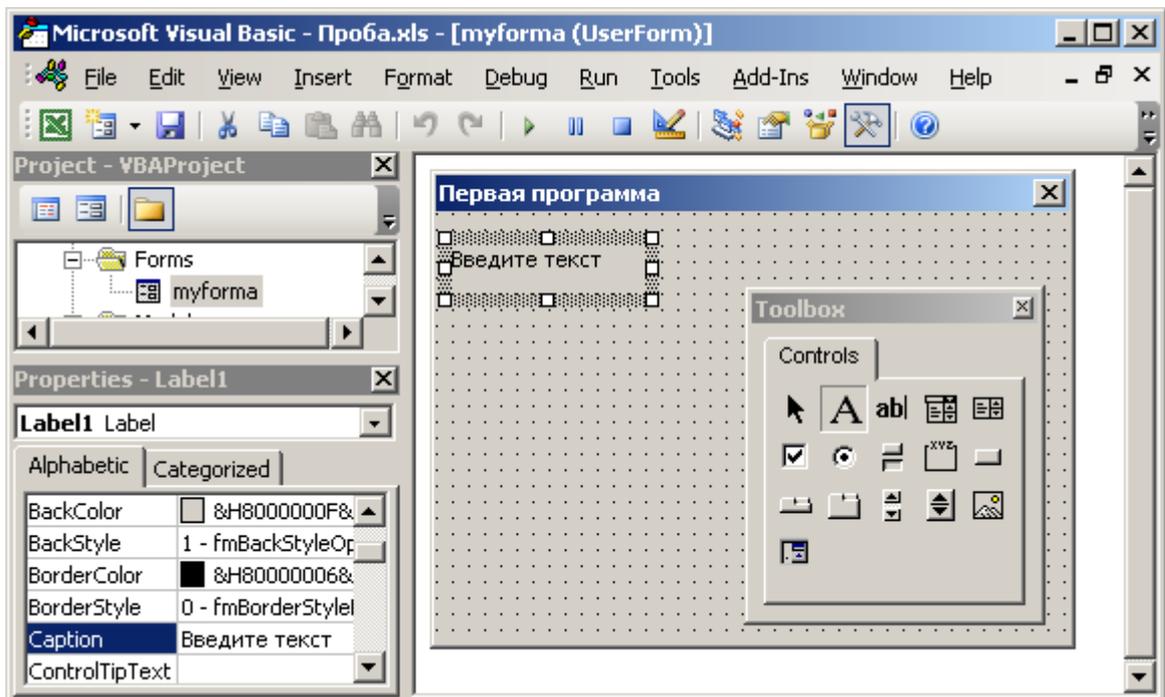
13. В окне свойств в поле *Name* ввести: **myforma** (в окне проекта изменится имя формы), а в поле *Caption* ввести: **Первая программа** (в окне формы изменится ее заголовок).



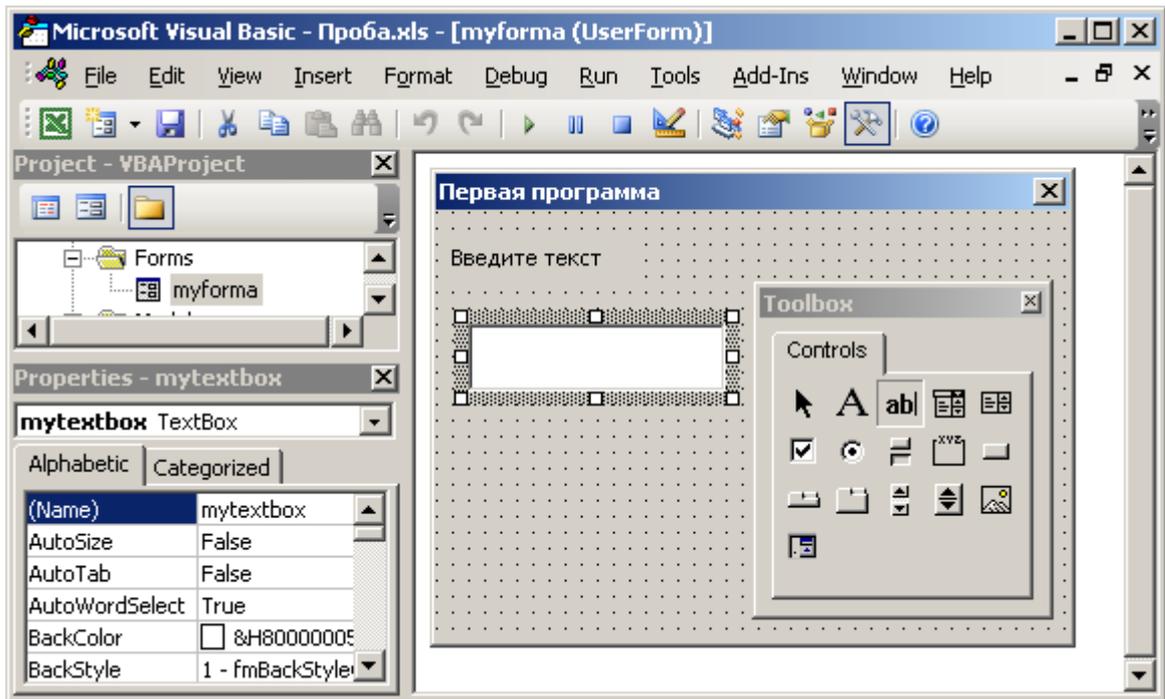
14. Щелчком мыши перейти в окно формы, активировав панель элементов (*ToolBox*). Если *ToolBox* был закрыт, его можно вызвать командой: **Вид (View) > Панель элементов (ToolBox)**.



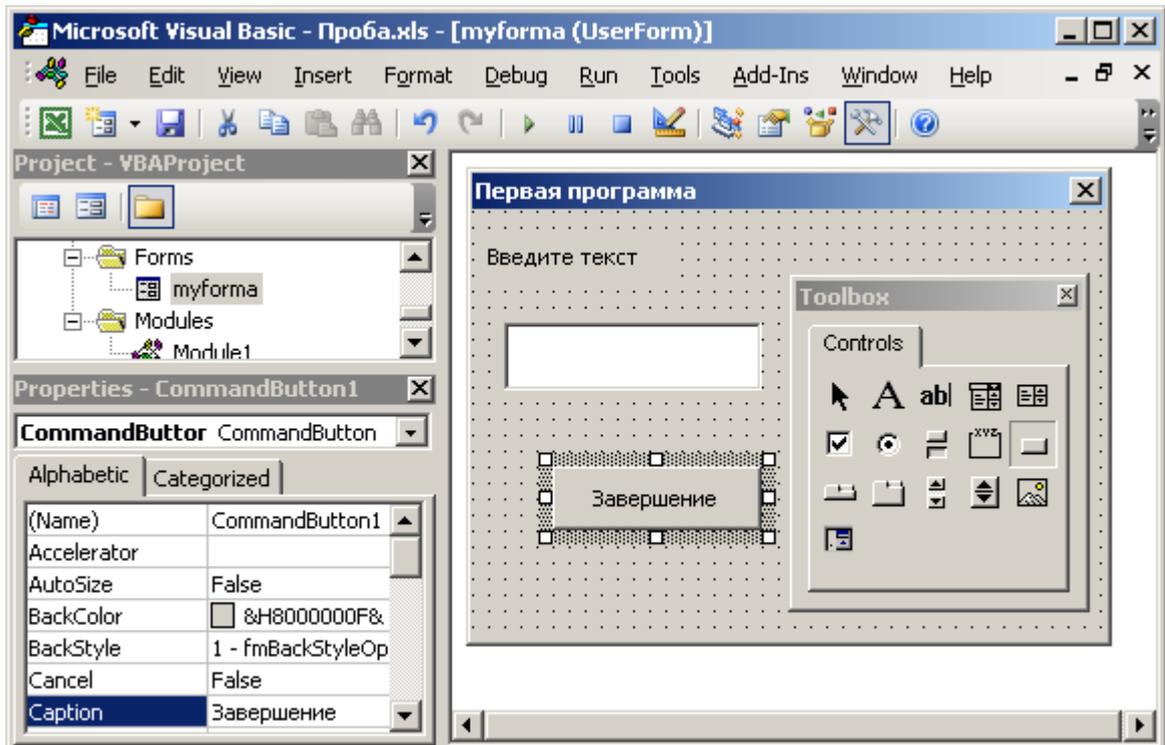
15. На панели элементов щелкнуть на кнопке *Надпись (Label)* и мышью "нарисовать" в форме контур элемента управления. В окне свойств для созданной надписи в поле *Caption* набрать: **Введите текст.**



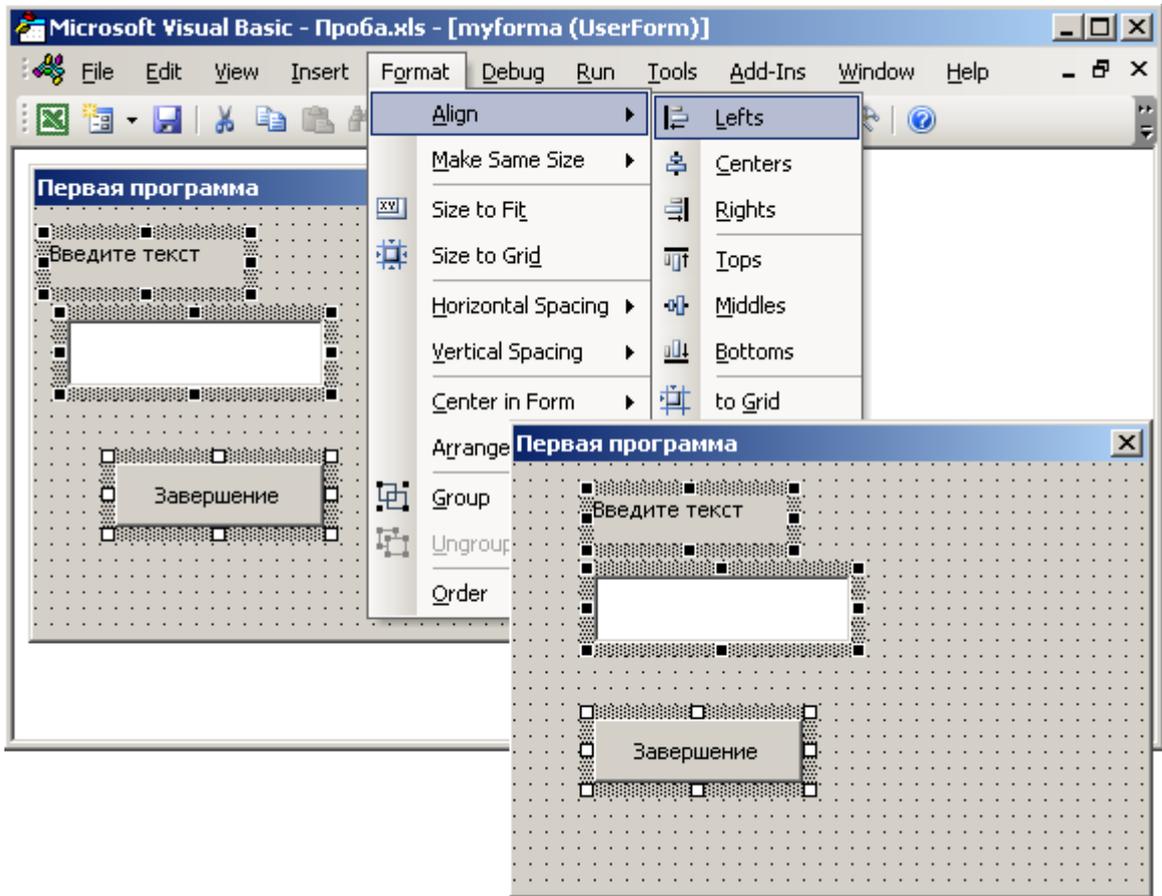
16. На панели элементов щелкнуть на кнопке *Поле (TextBox)* и добавить поле для ввода текста в форму (под надписью). В окне свойств для созданного поля в свойстве *Name* набрать: **mytextbox.**



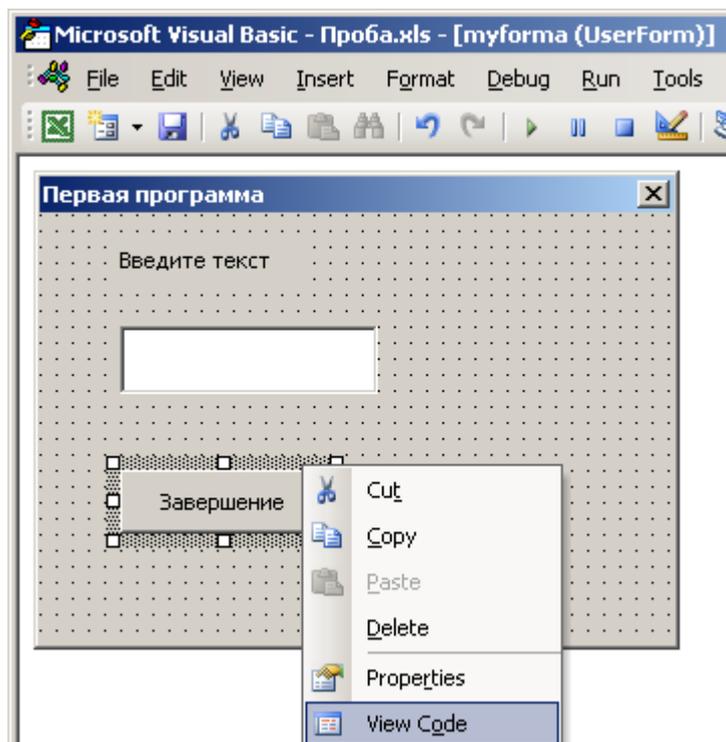
17. На панели элементов щелкнуть на кнопке **Кнопка (CommandButton)** и добавить управляющую кнопку в форму (под текстовым полем). В окне свойств для созданной кнопки в свойстве **Caption** набрать: **Завершение**.



18. Выделить все добавленные элементы в форме (надпись, текстовое поле и кнопку) и применить к ним команду: **Формат (Format) > Выровнять (Align) > По левому краю (Left)**.



19. Выделить в форме добавленную кнопку и вызвать программу для обработки связанных с ней действий командой: **Вид (View) > Программа (Code)** (или **окно проекта > кнопка Программа (View Code)**).



20. В окне программы в открывшемся шаблоне набрать:

Option Explicit

' Инструкция для обязательного объявления переменных

Private Sub CommandButton1_Click()

' Начало процедуры-программы, выполняемой при щелчке

' (клике) по кнопке по имени **CommandButton1**

Dim mytext As String

' Объявление (создание) переменной по имени **mytext**

' для хранения данных типа "строка" (**String**)

mytext = mytextbox.Text

' Запись текста из поля по имени **mytextbox** в переменную

' **mytext**

MsgBox "Введено: " & mytext

' Вывод окна-сообщения со строкой, начинающейся с

' "Введено: " и заканчивающейся текстом из переменной

' **mytext**

MsgBox "На листе:" & ActiveWorkbook.Worksheets(1).Cells(1, 2)

' Вывод окна-сообщения со строкой, начинающейся с

' "На листе: " и заканчивающейся текстом-значением

' из ячейки B1 первого листа текущего файла MS Excel

Unload myforma

' Закрытие формы по имени **myforma**

End Sub

' Завершение программы-процедуры

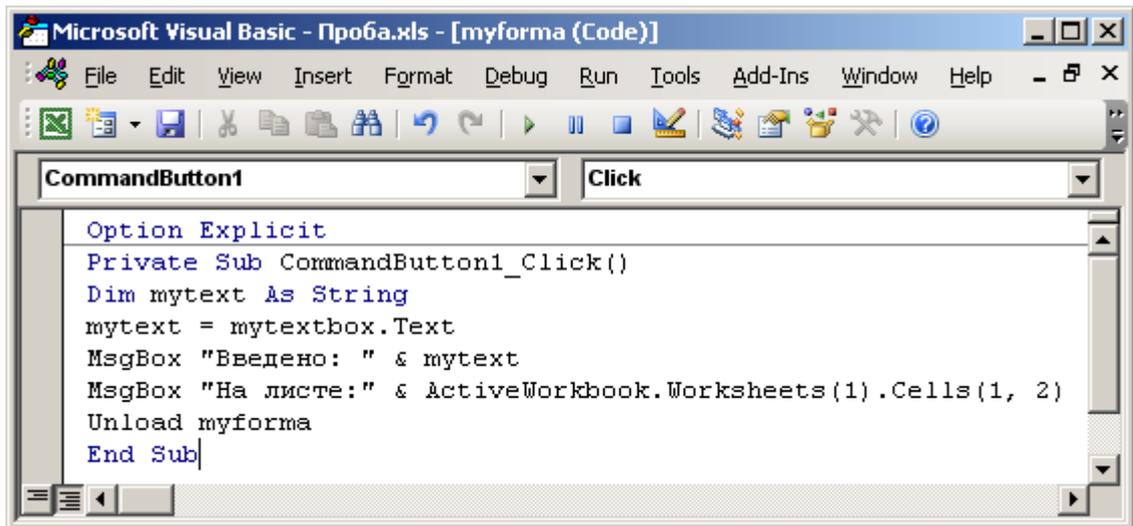
Шаблон процедуры для кнопки (**Private Sub CommandButton1_Click() ...**

End Sub) был добавлен автоматически. Процедура срабатывает при щелчке

(**Click**) по объекту – кнопке (**CommandButton1**). В окне программы

процедуры для различных объектов выбираются из двух списков (*Объект*

(*Object*) и *Процедура (Procedure)*).

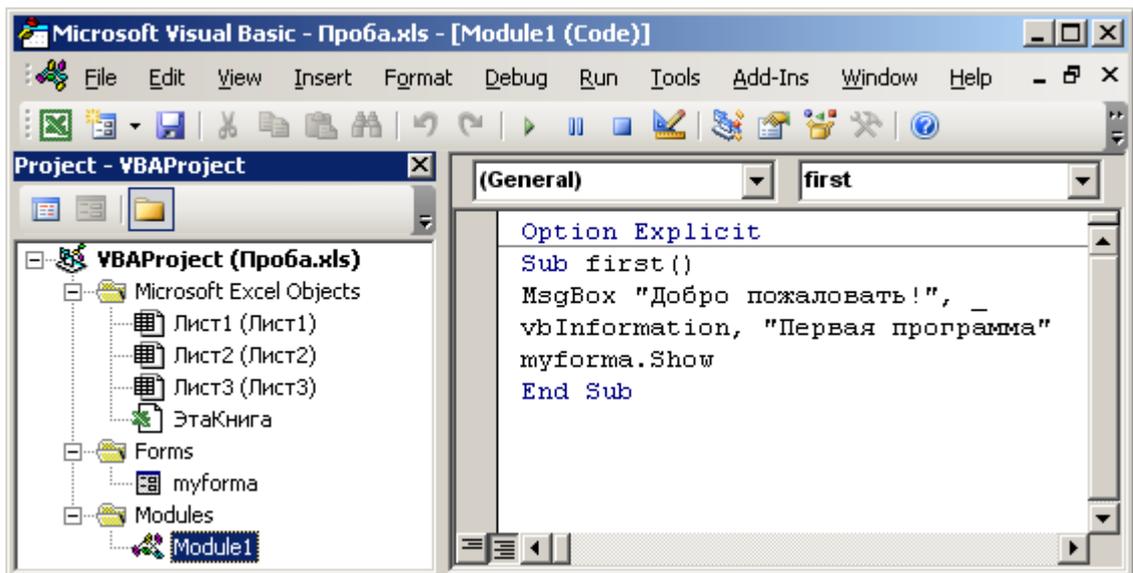


21. В окне программы для модуля (**Module1**) перед строкой

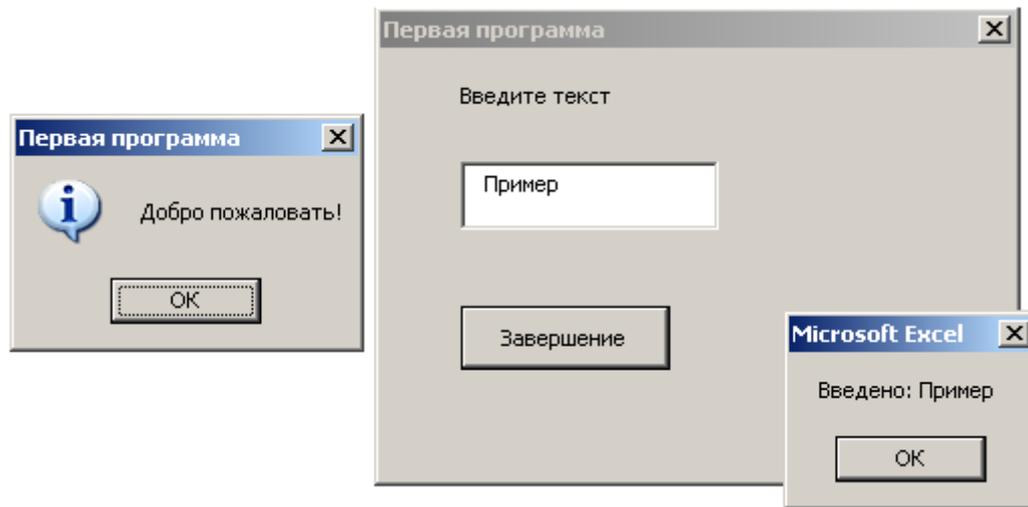
End Sub

добавить строку для отображения формы **myforma**

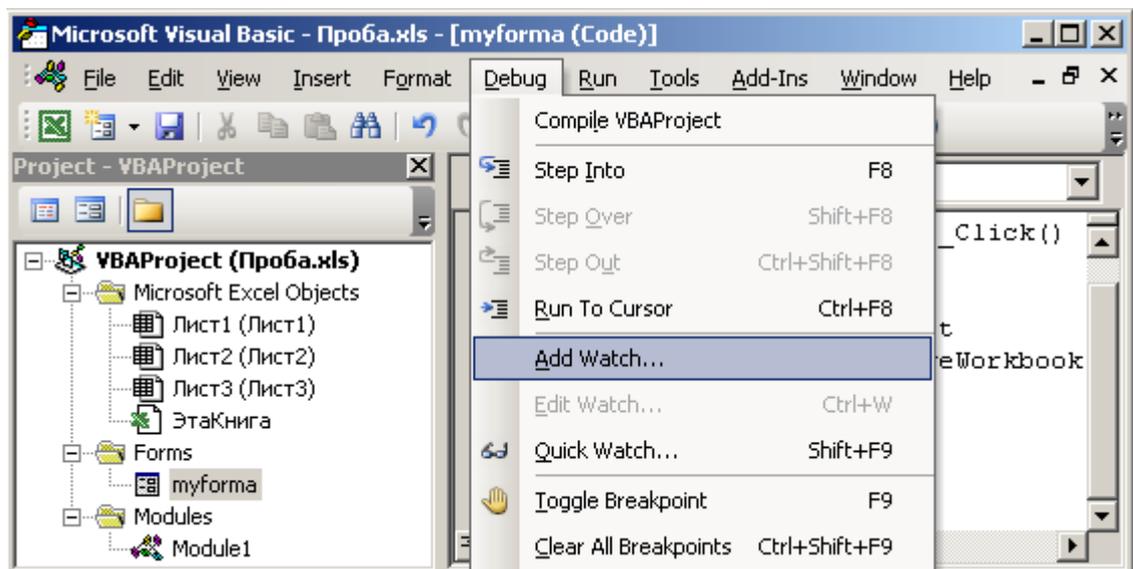
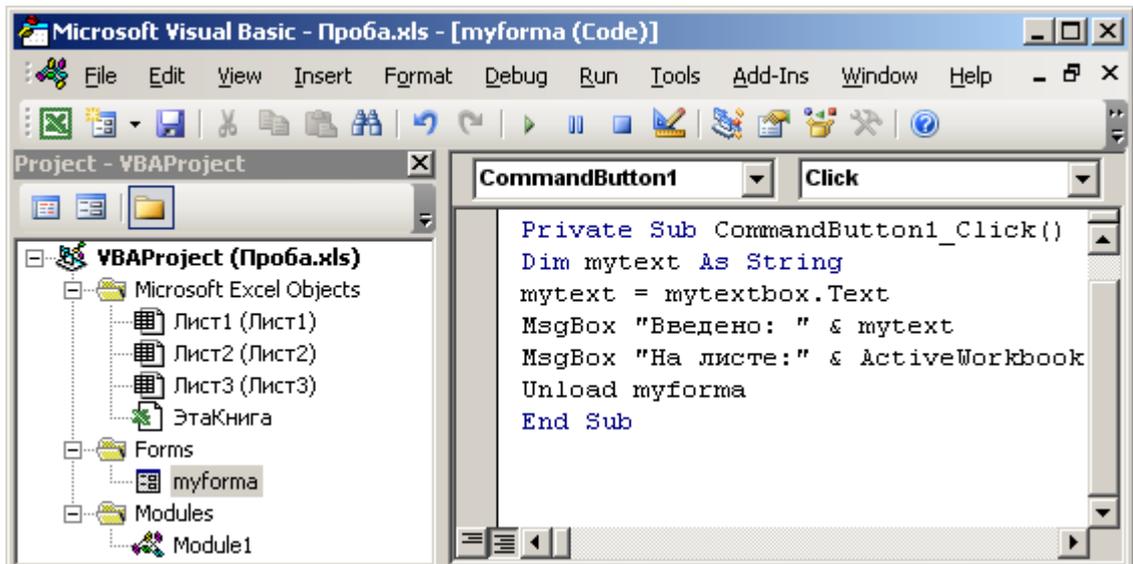
myforma.Show



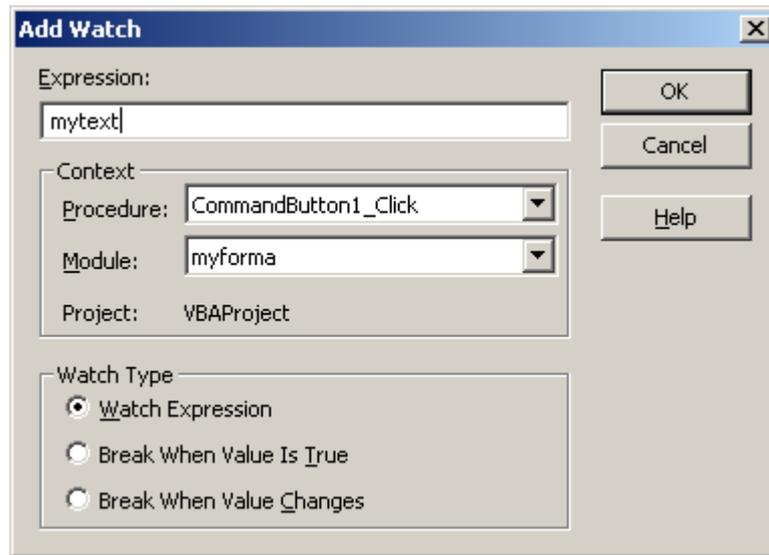
22. Сохранить файл, установить курсор в процедуру **first** и запустить модуль с процедурой **first** на выполнение (**F5**).



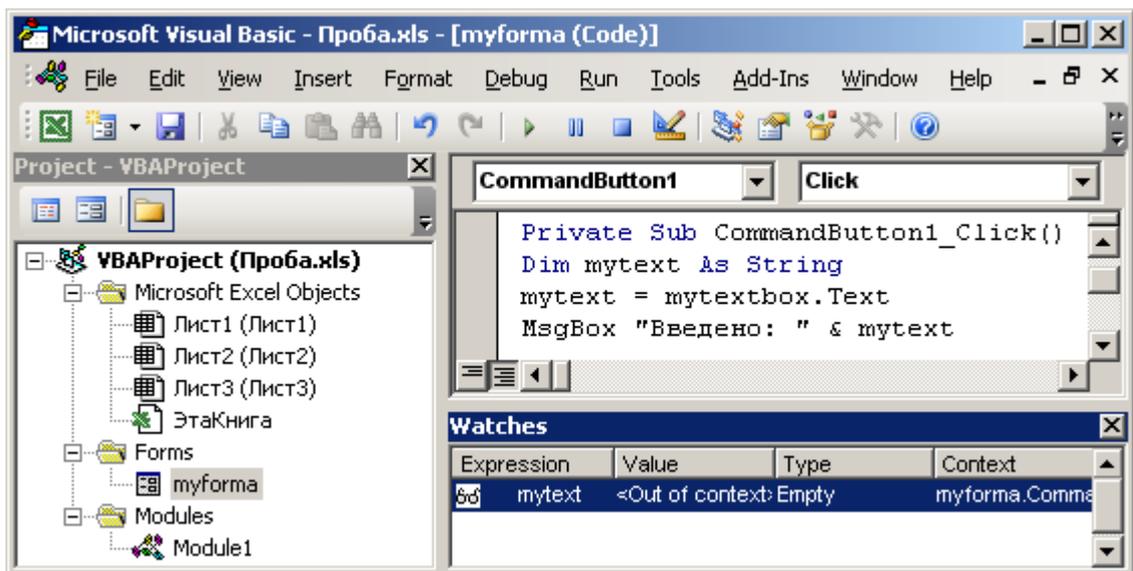
23. Перейти в окно программы для кнопки (**CommandButton1**) и выполнить команду: *Отладка (Debug) > Добавить контрольное значение (Add watch)*.



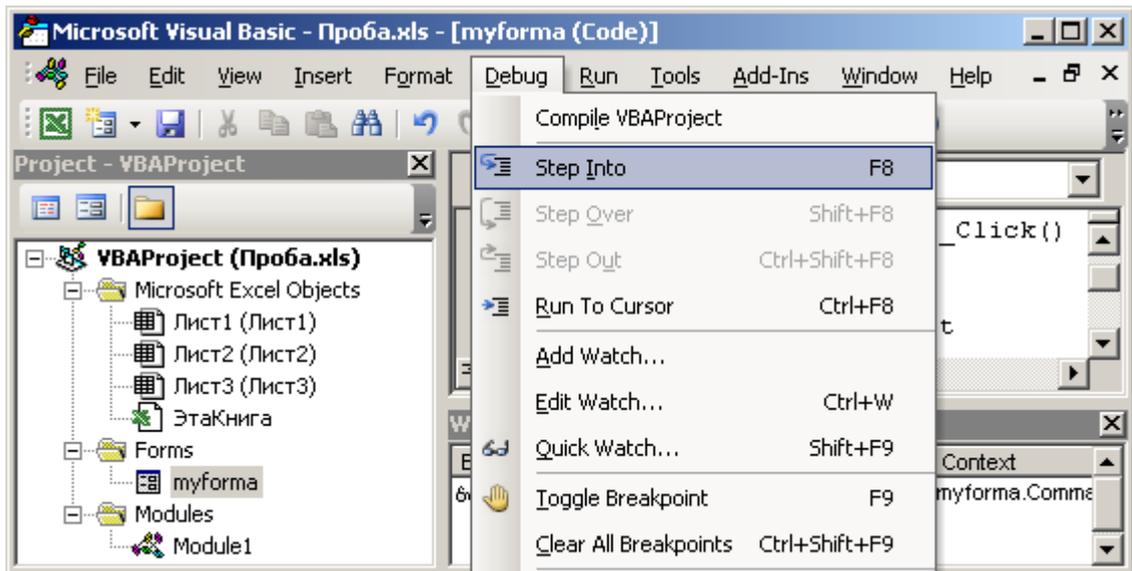
В открывшемся окне в поле **Выражение (Expression)** набрать имя переменной: **mytext**.



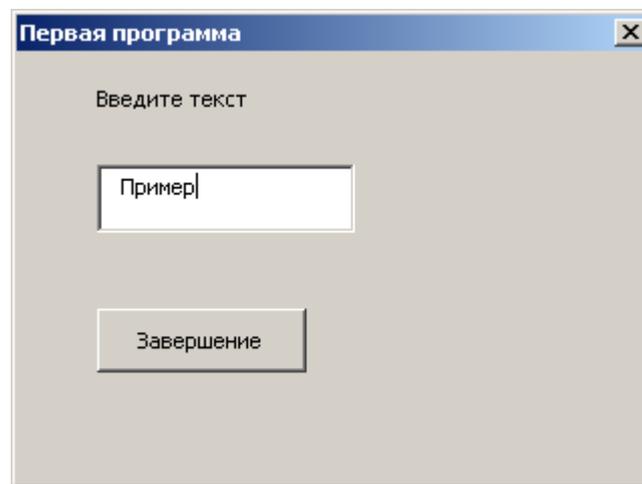
После нажатия **ОК** отобразится окно **Контрольное значение (Watch) (View) > Окно контрольного значения (Watch Window)**.



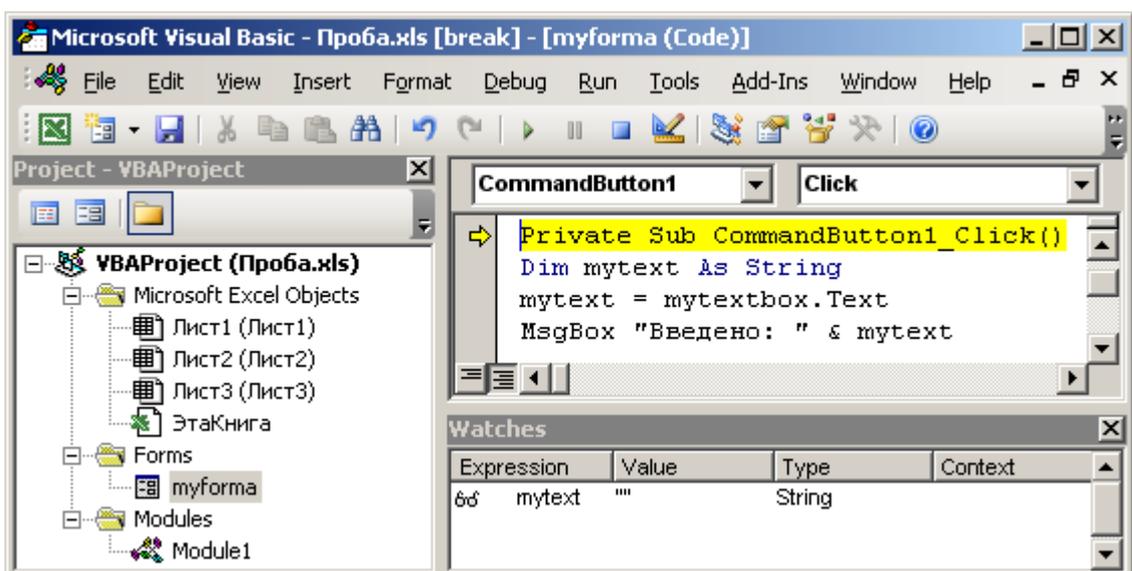
24. Запустить программу в режиме отладки (**Отладка (Debug) > Шаг с заходом (Step Into)** или **F8**).



В запущенном окне диалога ввести текст и нажать кнопку **Завершение**.

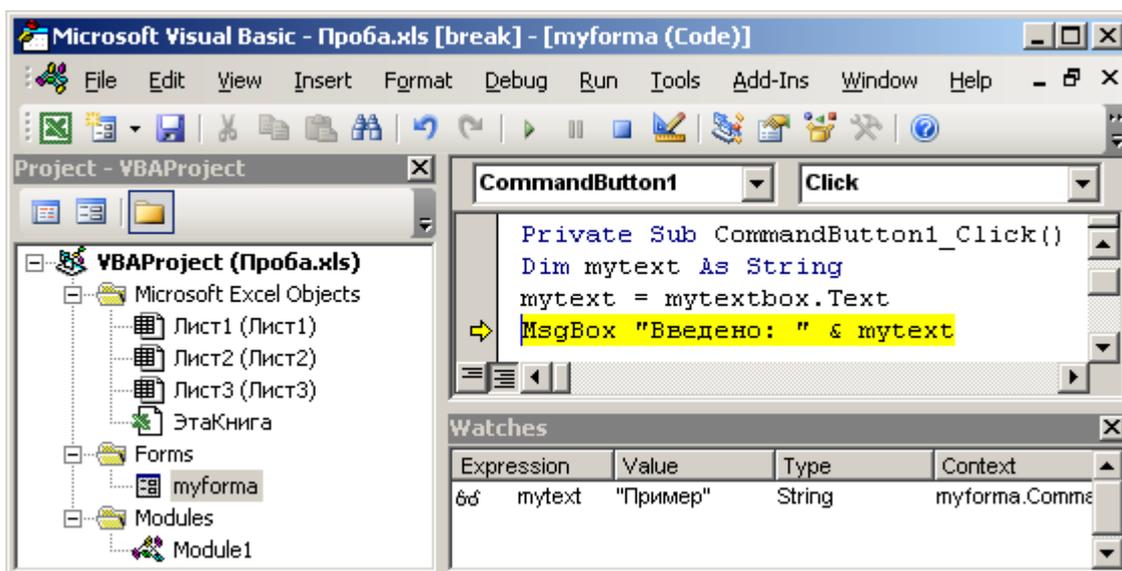


В открывшемся окне VB желтым цветом отмечается текущая операция.



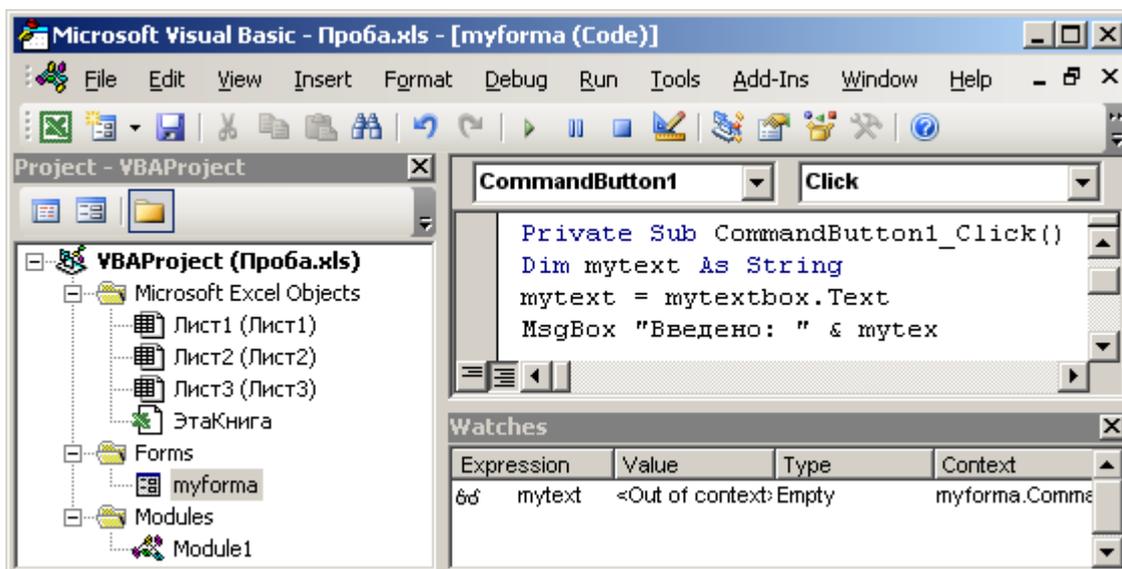
Для выполнения следующей операции нужно нажать **F8**. Выполняя программу по шагам, можно установить с помощью окна контрольного

значения при выполнении какой строки кода переменной **mytext** будет присвоено значение.



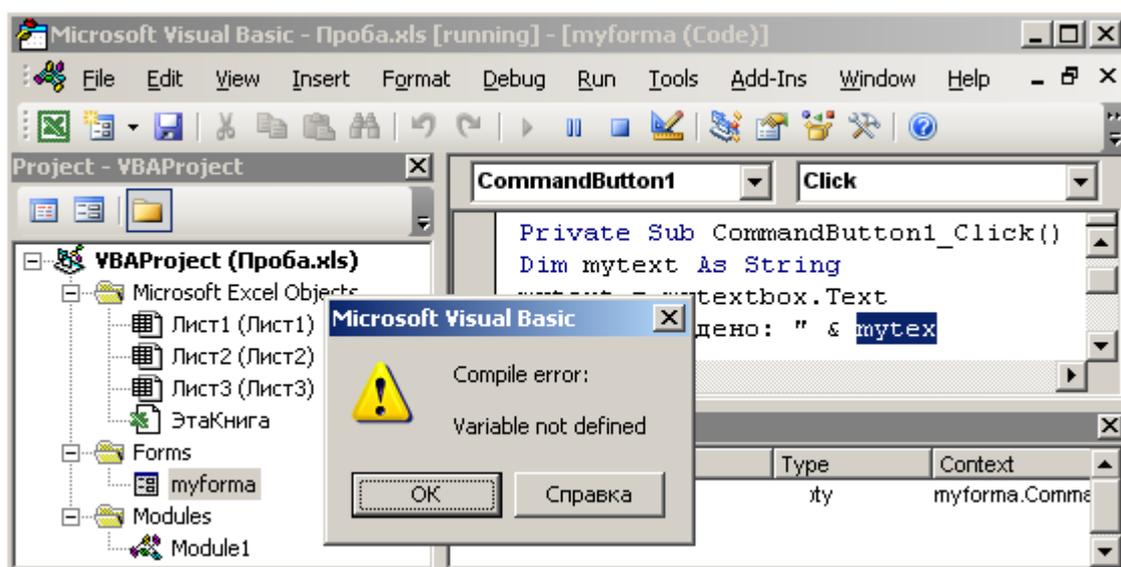
Продолжать выполнять программу по шагам (*F8*) до ее завершения.

25. После завершения выполнения программы в ее тексте заменить **mytext** на **mytex** в строке **MsgBox "Введено: " & mytex**.

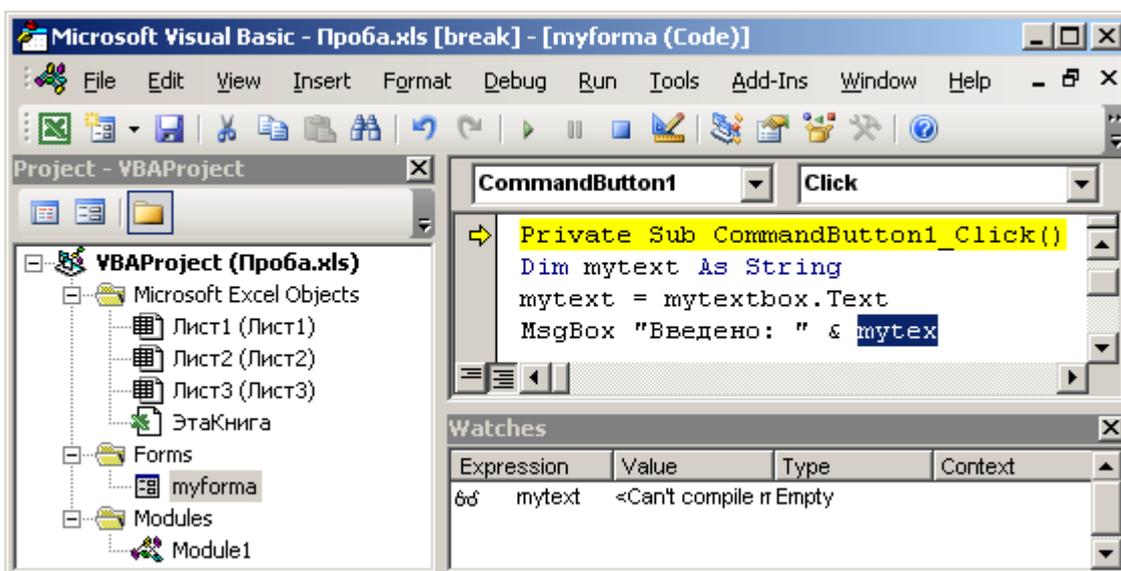


Запустить программу на выполнение (*F5*).

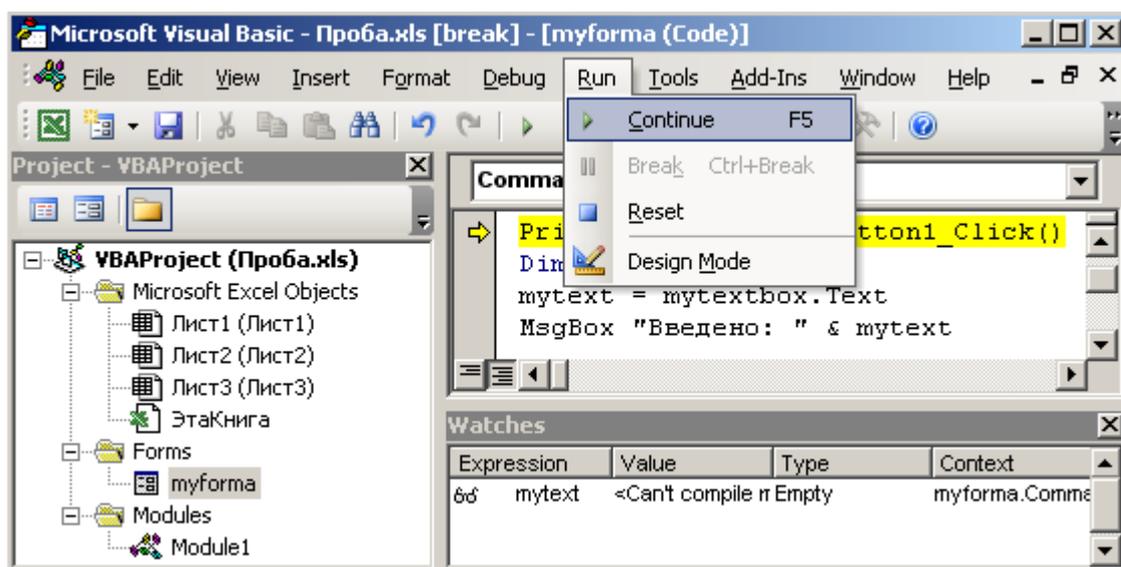
26. Ввести текст в поле ввода и нажать кнопку **Завершение**. Прочитать сообщение об ошибке и нажать в его окне **OK**.



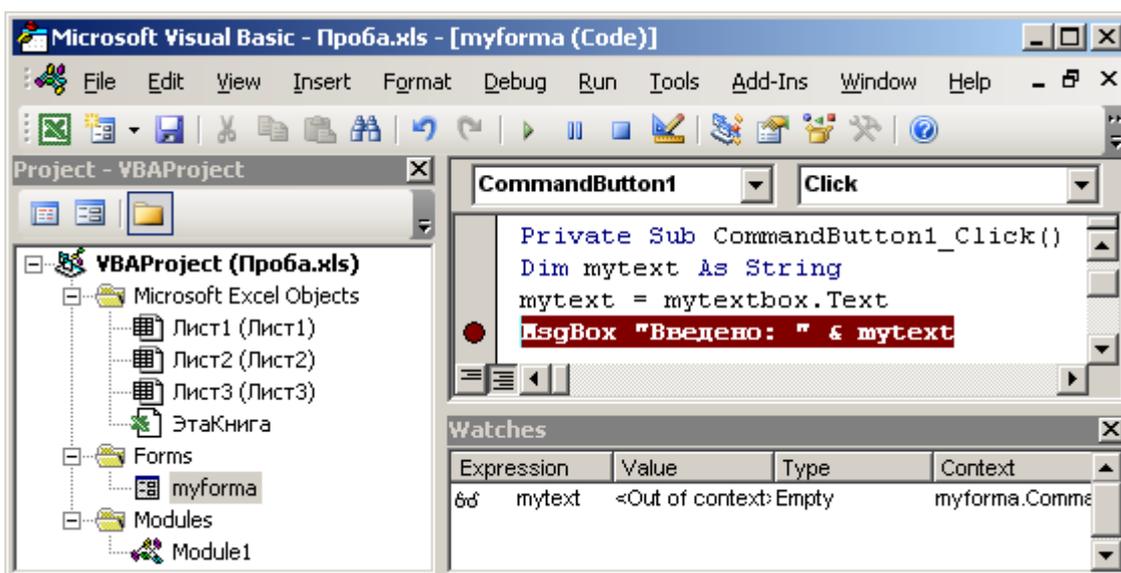
В открывшемся окне VB желтым цветом будет отмечена процедура, содержащая ошибку, а синим – место ошибки.



Ошибку можно устранить (заменить **mytex** на **mytext**), не завершая текущий запуск программы (исправить ошибку и нажать кнопку **F5** (*Продолжить (Continue)*)) или завершив его (*Запуск (Run)* > *Сброс (Reset)*) для дальнейшего редактирования.

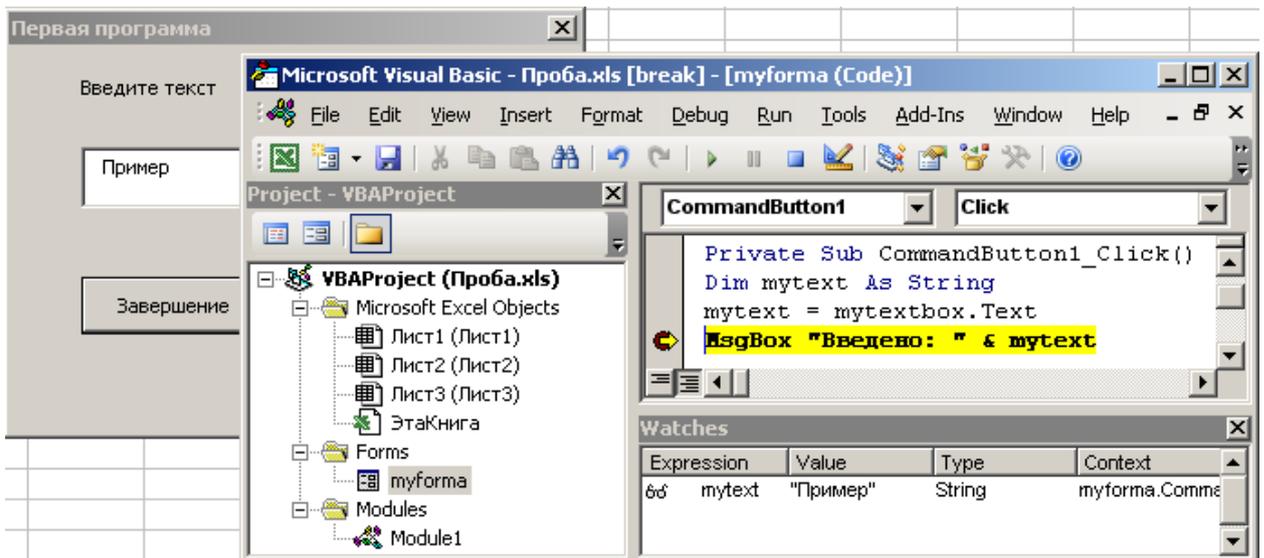


27. После завершения выполнения программы и устранения ошибки в процедуре **CommandButton1_Click()** установить курсор перед ключевым словом **MsgBox** и выполнить команду *Отладка (Debug) > Точка останова (Toggle Breakpoint)* или щелкнуть на поле слева от строки.



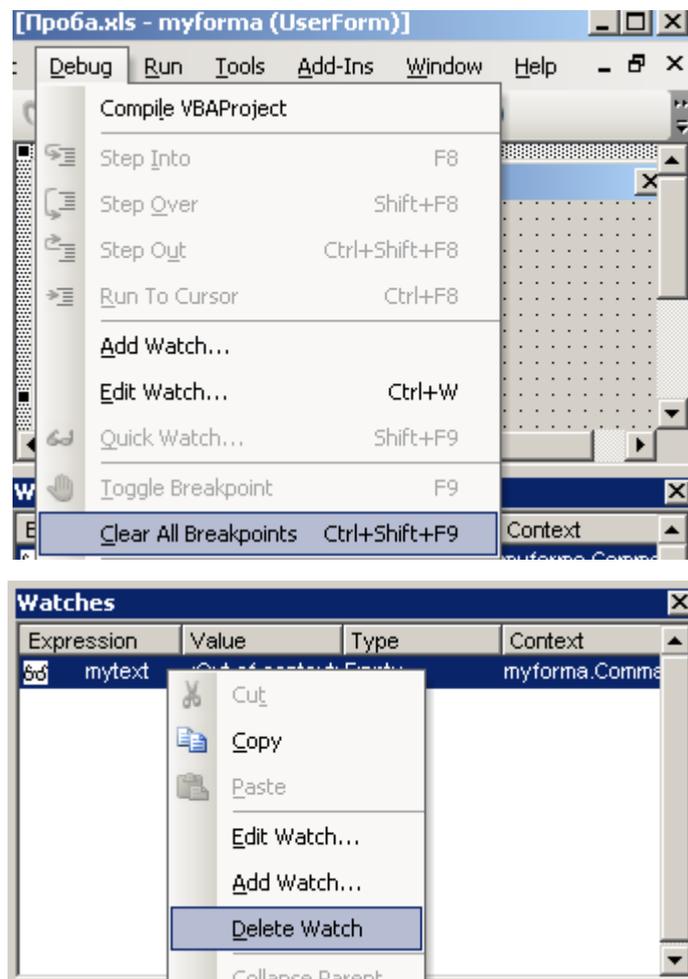
Запустить программу на выполнение (**F5**).

28. Ввести текст в поле ввода и нажать кнопку **Завершение**. Процедура будет приостановлена перед строкой отмеченной желтым цветом (точка останова). Убедиться в наличии значения в переменной **mytext** (окно контрольного значения).



Завершить выполнение программы (**F5** или **F8 (по шагам)**).

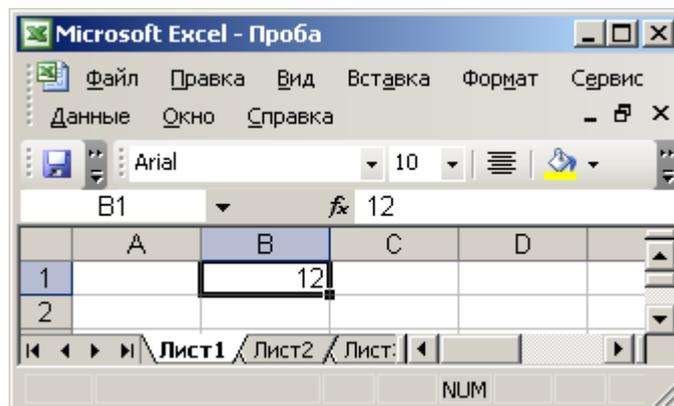
29. Очистить все точки останова (щелчок слева от строки с остановом или **Отладка (Debug) > Снять все точки останова (Clear All BreakPoints)**) и контрольные значения (в окне контрольного значения команда контекстного меню **Удалить контрольное значение (Delete Watch)**).



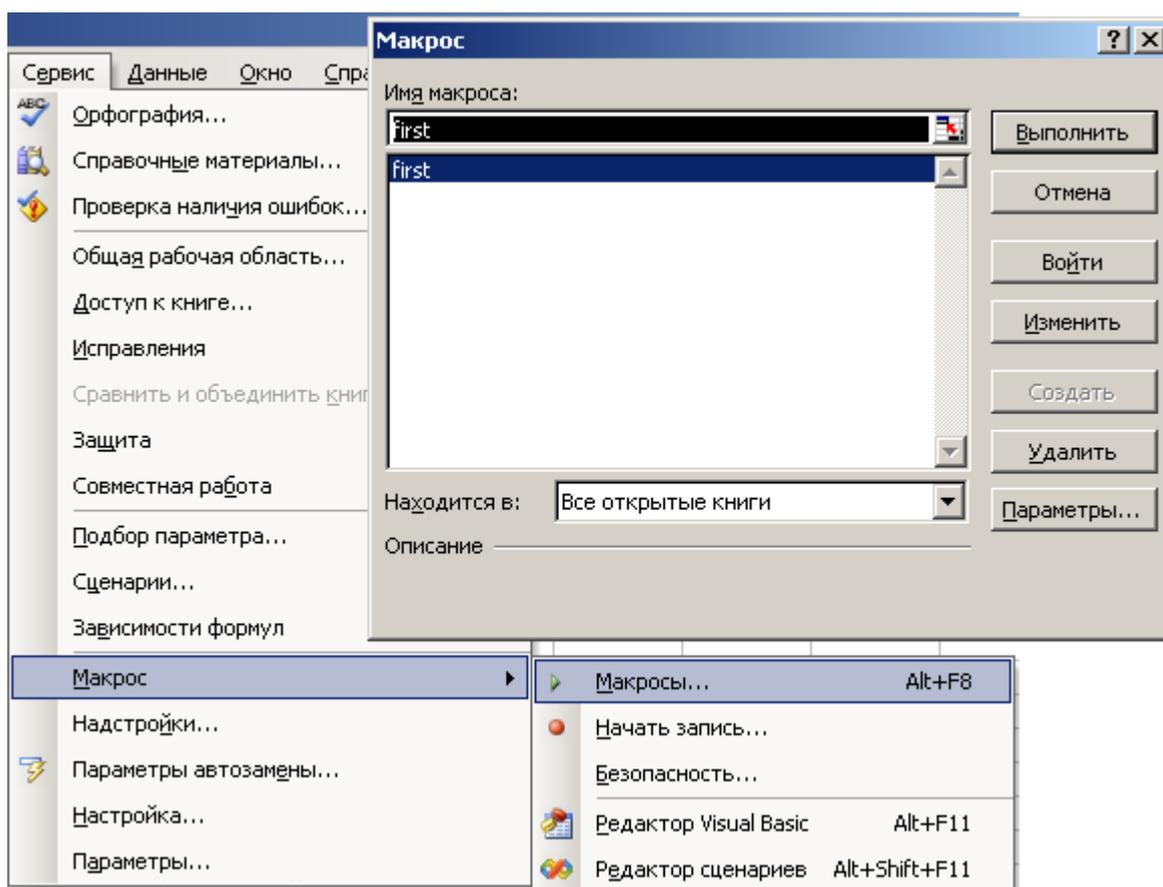
Сохранить файл.

30. Закрывать проект и вернуться в MS Excel (**Файл (File) > Закрывать и вернуться в MS Excel (Close and Return to Microsoft Excel)**).

31. На первом листе книги MS Excel в ячейке **B1** ввести любое число.

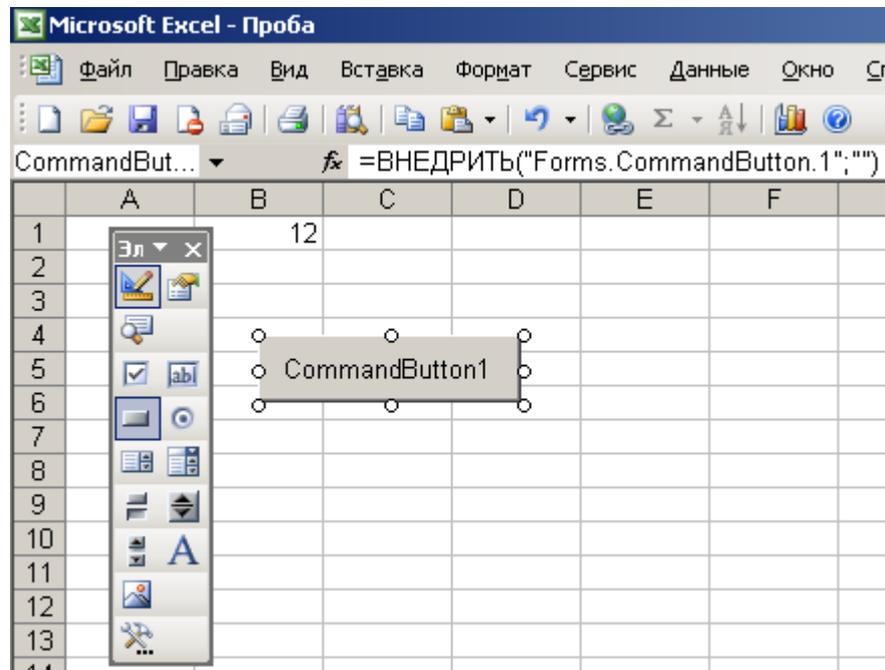


32. Запустить процедуру **first** командой оболочки MS Excel: **Сервис > Макрос > Макросы > first (в списке Имя макроса) > Выполнить**.

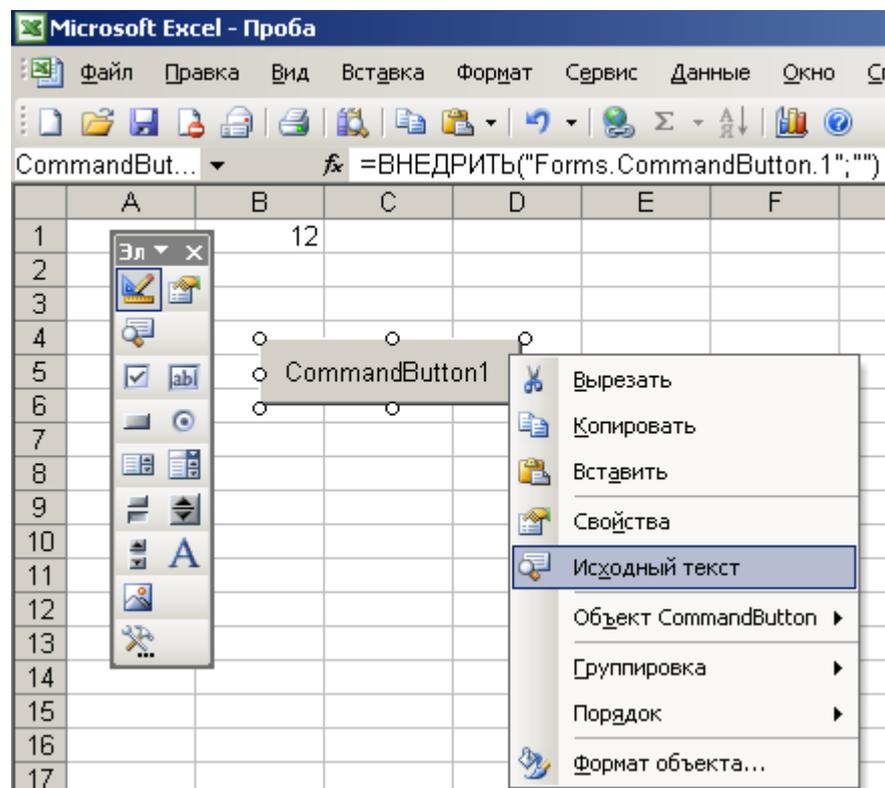


Обратить внимание на появлении в окне результатов работы программы значения из ячейки таблицы MS Excel.

33. В оболочке MS Excel выполнить команду: **Вид > Панели инструментов > Элементы управления**. Выбрать на панели элемент управления **Кнопка** и нарисовать его контур на листе MS Excel.



34. В контекстном меню кнопки выполнить команду **Исходный текст**.

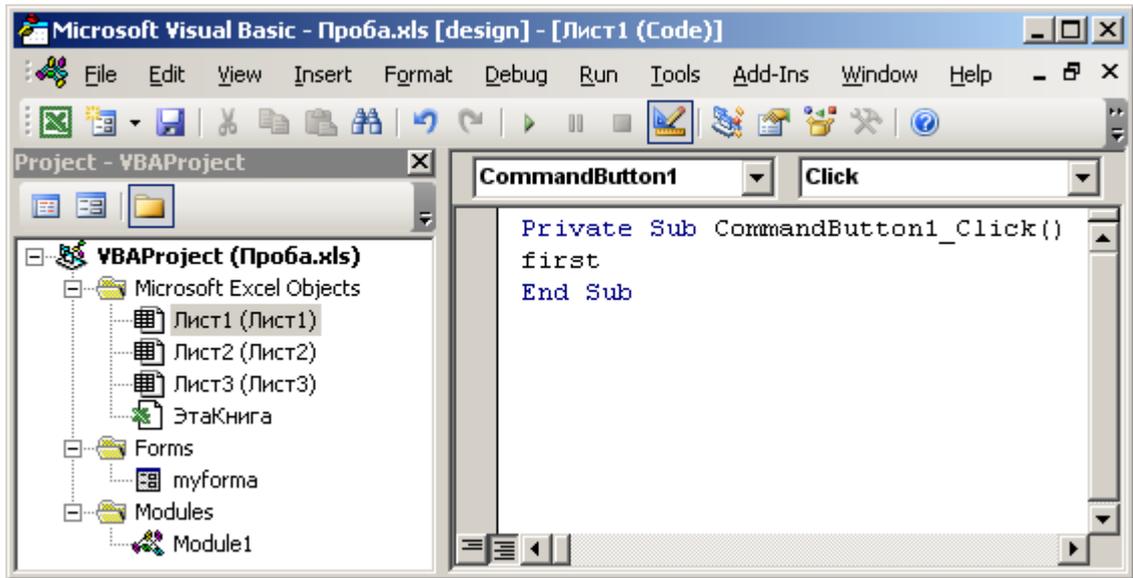


35. В открывшемся окне VB внутри шаблона процедуры новой кнопки вызвать процедуру **first**.

Private Sub CommandButton1_Click()

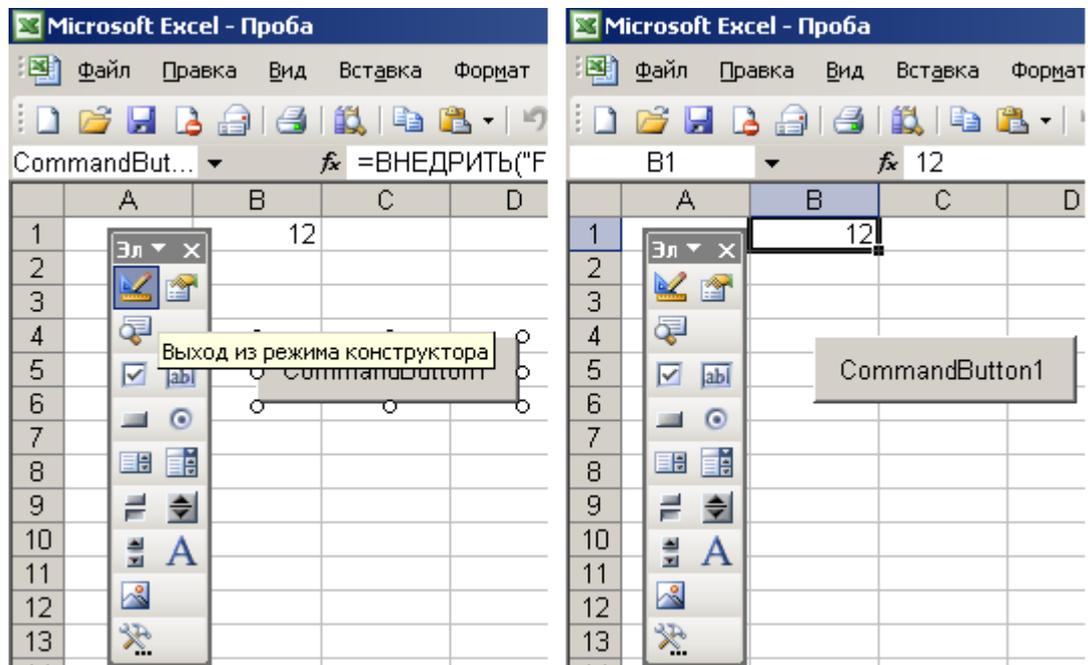
first

End Sub



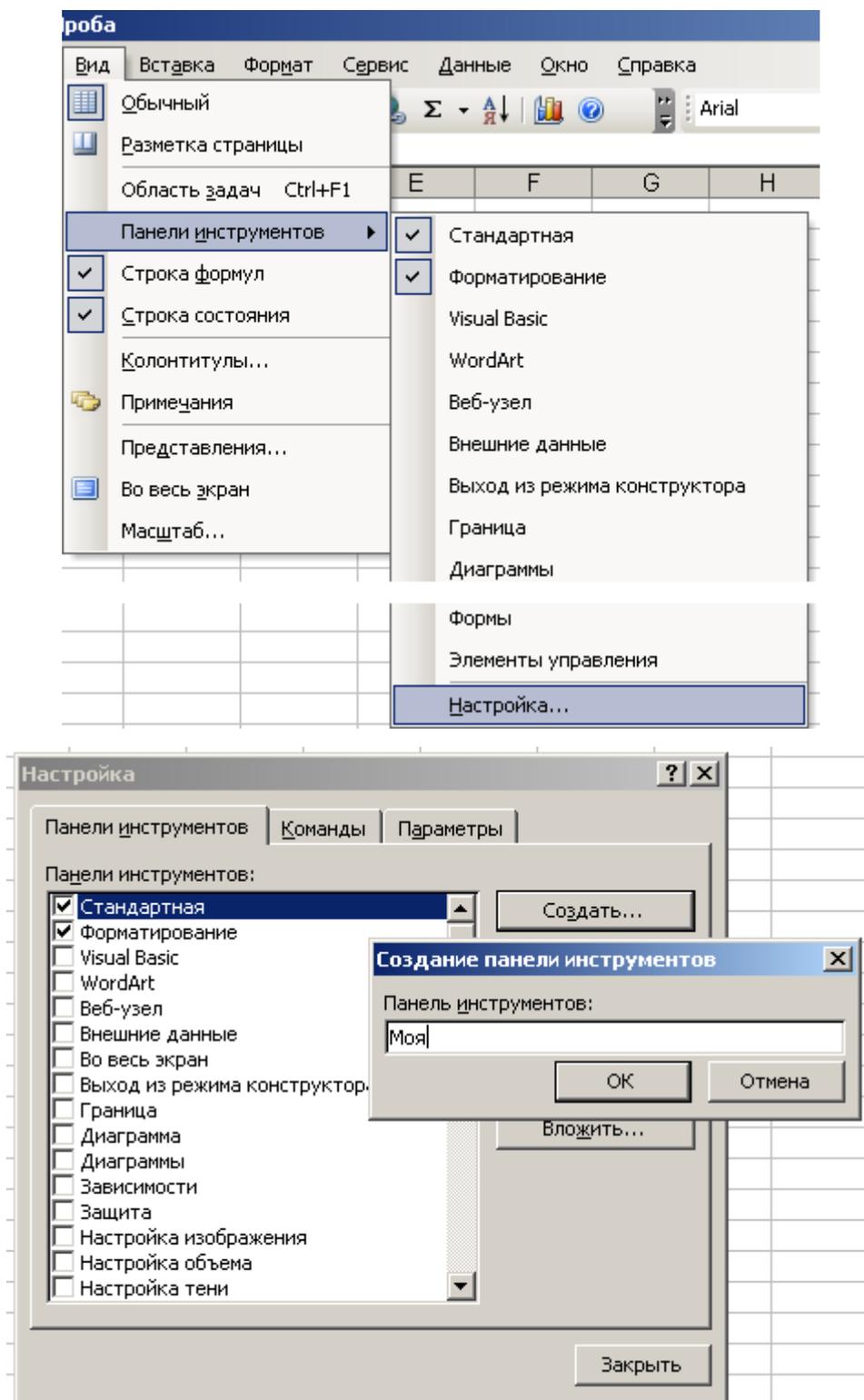
36. Сохранить файл и вернуться в MS Excel.

37. Выполнить команду: панель *Элементы управления* > *Режим конструктора или Вид* > *Панели инструментов* > *Visual Basic* и далее панель *Visual Basic* > *Выход из режима конструктора*.

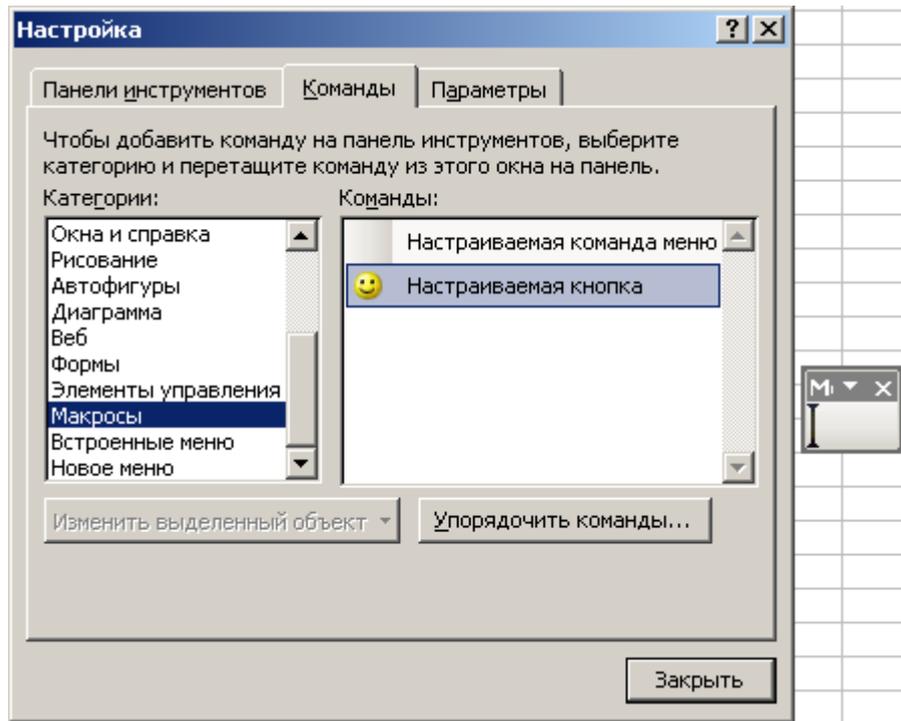


38. Нажать на добавленную кнопку на листе MS Excel. Ввести текст в окно созданной программы и завершить ее.

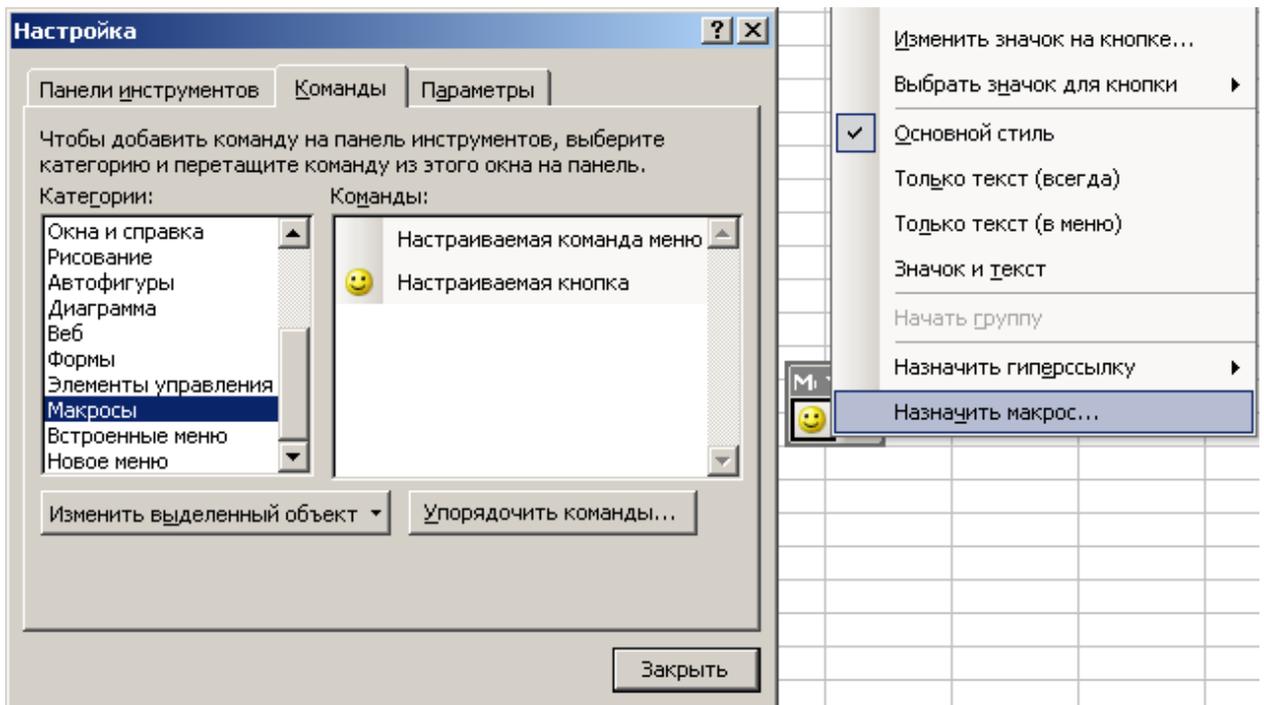
39. Выполнить команду оболочки MS Excel: **Вид > Панели инструментов > Настройка > Панели инструментов > Создать**. Назвать новую панель и нажмите **ОК**.



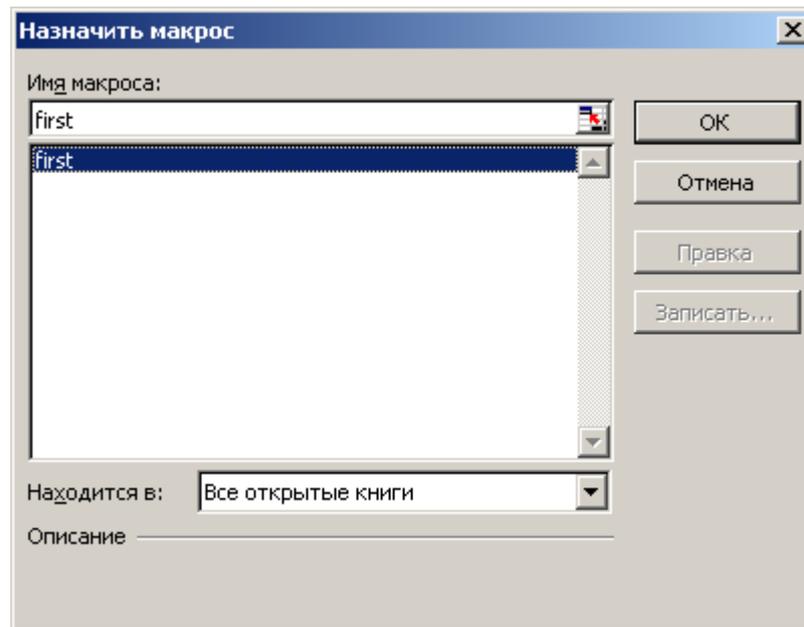
40. В закладке **Команда** окна **Настройка** выбрать категорию **Макросы**. В списке **Команды** выбрать **Настраиваемая кнопка** и перетащить ее на созданную панель.



41. С помощью контекстного меню на кнопке новой панели инструментов выполнить команду *Назначить макрос*.

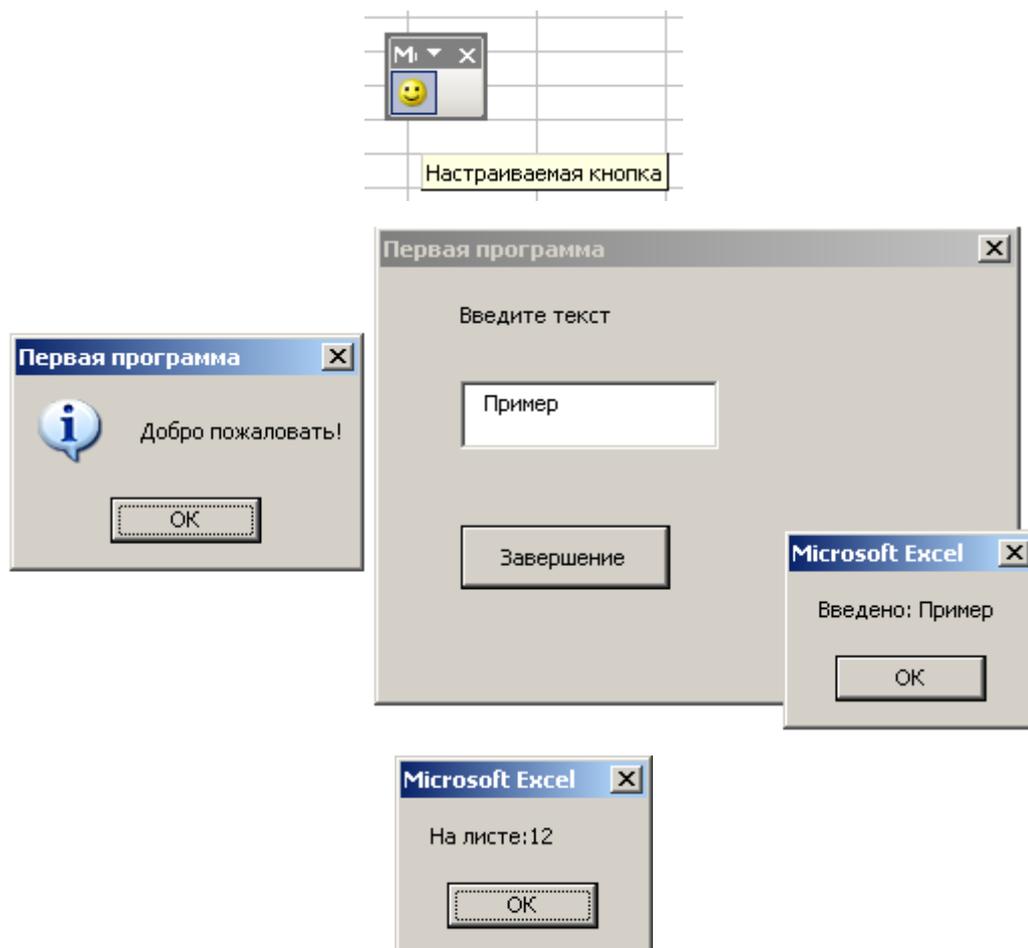


В предложенном списке выбрать процедуру **first** и нажать **OK**.



Закрывать окно *Настройка*.

42. Нажать на кнопку созданной панели инструментов.



Состав отчета

1. Номер, название и цель работы.

2. Ответы на контрольные вопросы:

- а) Каким образом открыть редактор VB в MS Excel?
- б) Каким образом добавить программный модуль и сохранить проект VB MS Excel?
- в) Как получить информацию о синтаксисе текущей процедуры или функции в VB?
- г) Как вызывается и для чего используется Окно свойств VB (пример)?
- д) Как добавить форму в проект VB? Как добавить элемент управления в форму?
- е) Как вызвать окно программного кода для элемента управления?
- ж) Как добавить контрольное значение в VB?
- з) Как запустить программу с остановкой на каждом шаге?
- и) Как добавить точки останова в программу VB?
- к) Перечислите и поясните способы запуска программ VB в MS Excel.

ЛАБОРАТОРНАЯ РАБОТА № 1. ЗНАКОМСТВО СО СРЕДОЙ ПРОГРАММИРОВАНИЯ VISUAL BASIC (ДЛЯ MS OFFICE 2007-2010)

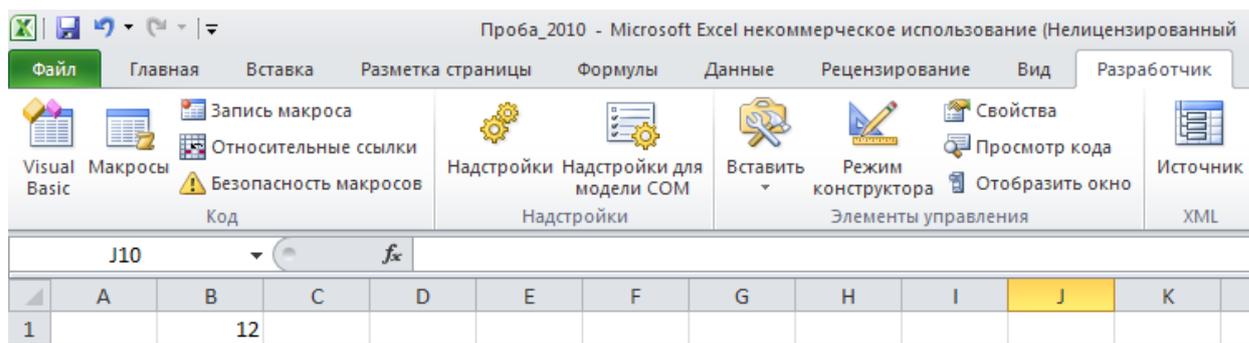
Цель работы

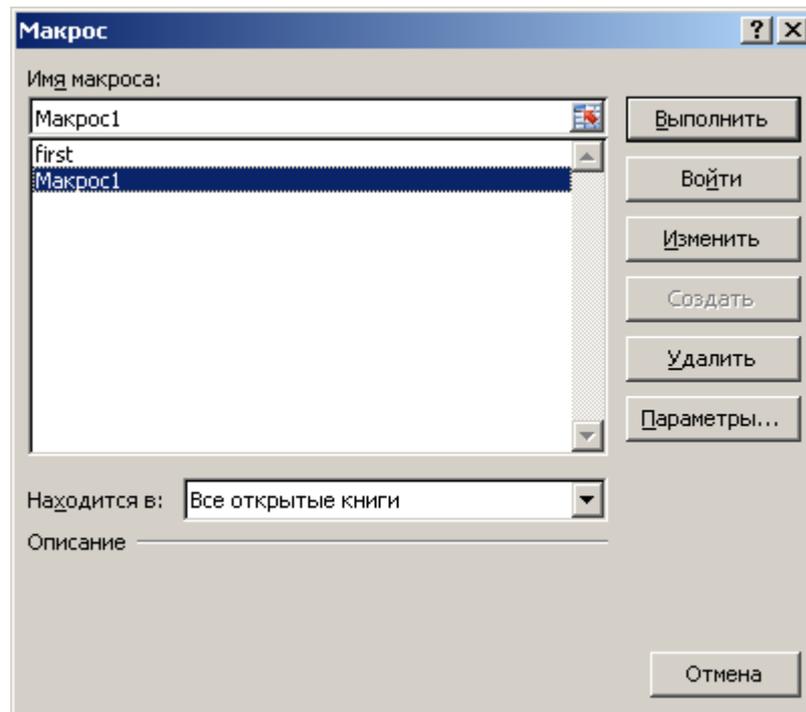
Получить представление о принципах работы в среде программирования Visual Basic (VB).

Теоретические сведения

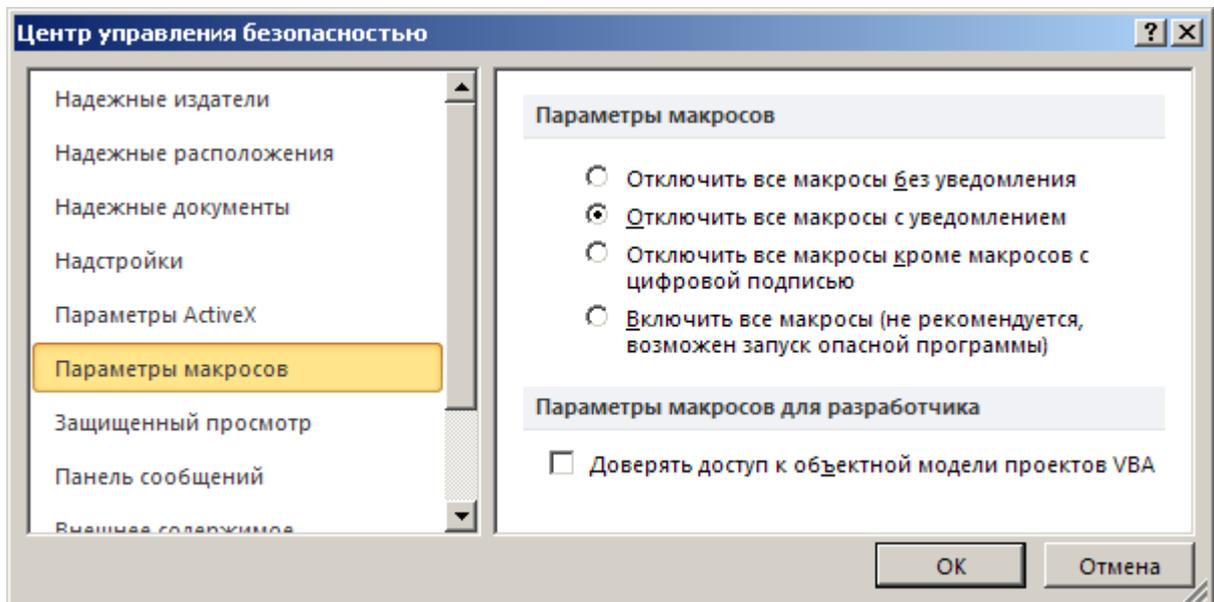
Разработка специалистом конкретной предметной области программных продуктов может быть оправдана при использовании языков программирования, имеющих не только широкие функциональные возможности, но и синтаксис, близкий к естественному математическому и логическому представлению объектов рассматриваемой области. К таким языкам относится VB. В виде интегрированного средства (Visual Basic for Application (VBA)) он входит в пакет MS Office, а многие программы позволяют использовать собственные объекты с помощью библиотек VB.

Для работы в VBA в любом приложении MS Office предназначена вкладка ленты *Разработчик*. Вкладку можно отобразить, вызвав на ленте контекстную команду *Настройка ленты* и выбрав в списке основных вкладок вкладку *Разработчик*. (Для Office 2007 отображение вкладки Разработчик производится командой кнопка *Office > Параметры... > Основные > Показывать вкладку "Разработчик" на ленте*).

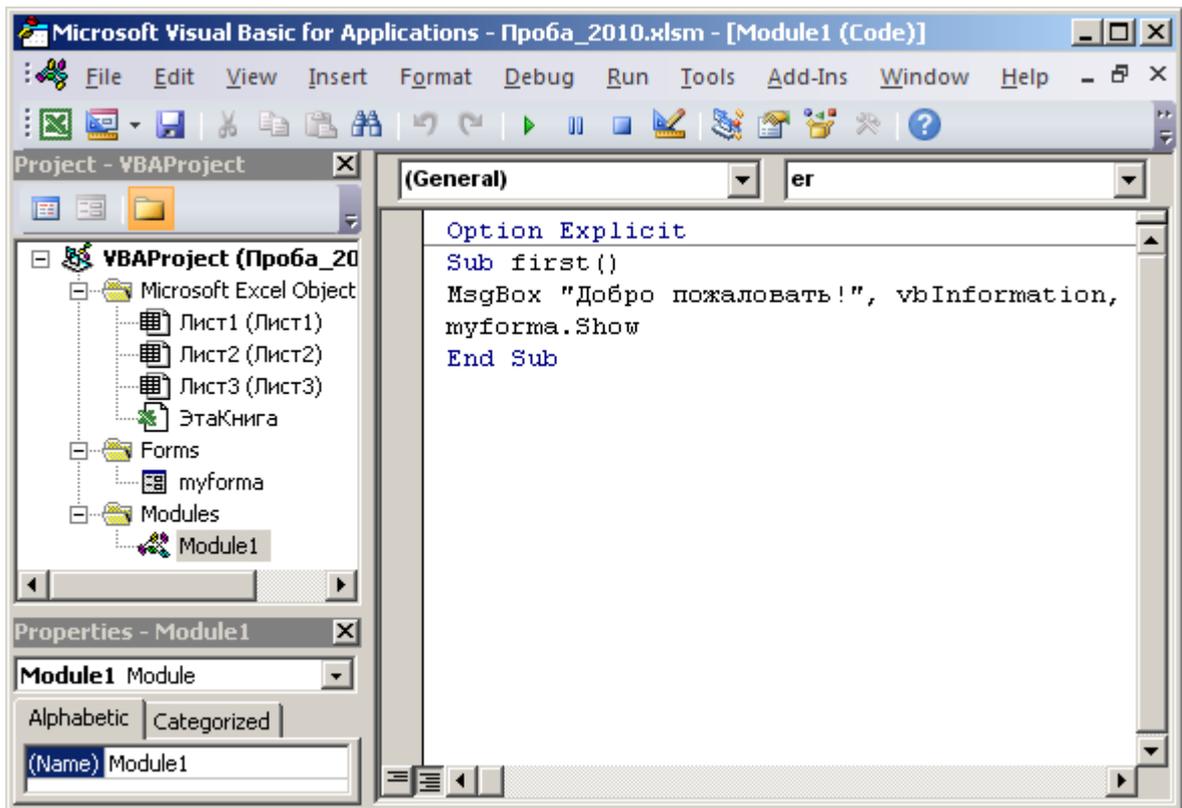




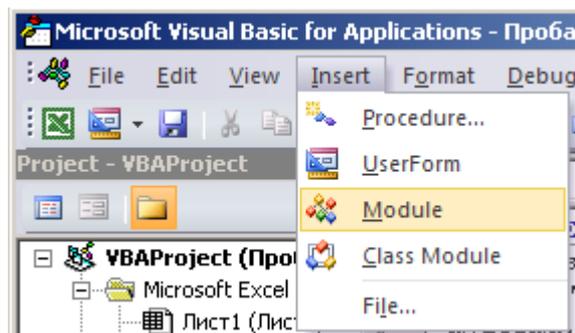
Команда *Security (Безопасность макросов)* позволяет установить уровень защиты от запуска макросов (н.п., источник содержащего их файла неизвестен), так как некоторые из них могут выполнять опасные (нежелательные) действия.



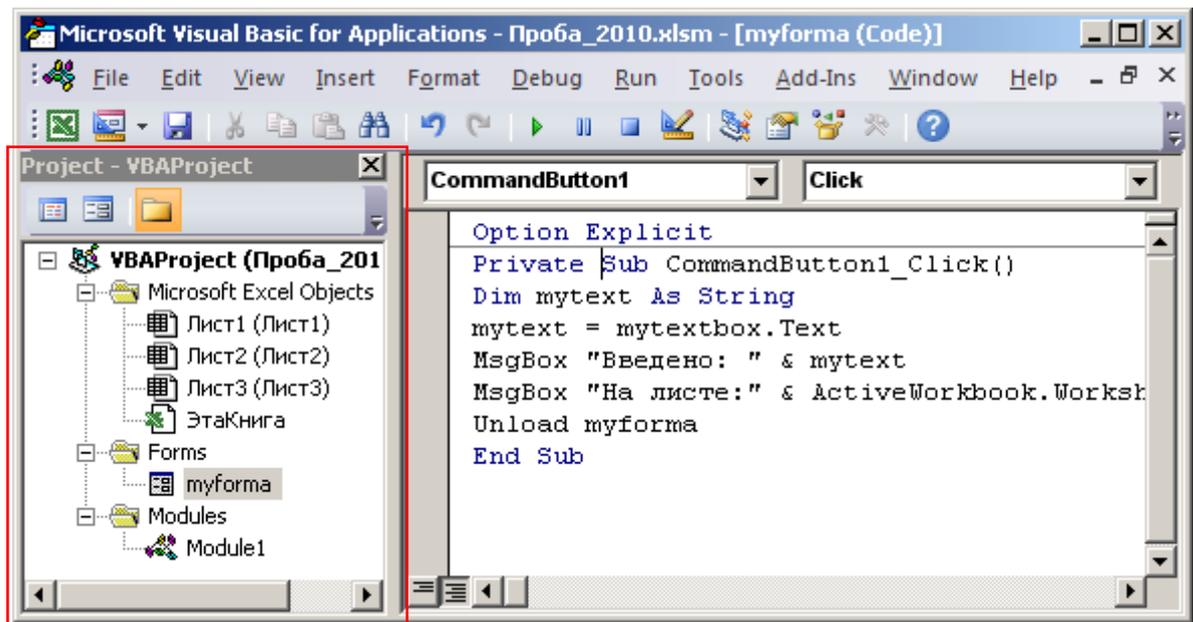
Команда *Visual Basic (Редактор Visual Basic)* открывает оболочку для создания и редактирования программ VB (в т.ч. макросов).



Оболочка VB позволяет создавать программные проекты на базе модулей (*Insert (Вставка) > Module (Модуль)*), электронных форм (*Вставка (Вставка) > User Form (Пользовательская форма)*) и модулей пользовательских объектов (*Insert (Вставка) > Class Module (Модуль класса)*).



Контролировать состав проекта VB и осуществлять навигацию между всеми его компонентами можно в окне проекта (*View (Вид) > Project Explorer (Окно проекта)*), снабженного кнопками отображения объектов  (*View Object*) для форм и программного кода  (*View Code*) для форм и модулей.



Окно проекта (Project Explorer)

Модуль – это лист с текстом программы, вставленный в документ MS Office (записанные макросы добавляются именно в модули). Программный проект VB может состоять из нескольких модулей. Модули могут находиться в разных документах MS Office.

Структура программы в модуле VB следующая:

- 1) ключевое слово – тип программного фрагмента: функция, процедура, объявление переменных, объявление типа данных, объявление свойств;
- 2) имя программного фрагмента;
- 3) опции программного фрагмента: параметры, переменные;
- 4) объявления и инициализация переменных для функций и процедур;
- 5) программный код, реализующий необходимый пользовательский алгоритм;
- 6) завершение программы: выходные результирующие значения, ключевое слово.

Пример программы VB в форме процедуры, выводящей на экран текстовое сообщение (' – символ-метка комментария):

Sub qwerty()

'Sub – ключевое слово для программы-процедуры

' по имени "qwerty", () – без входных параметров

Dim usertitle As String

'Dim – объявление переменной "usertitle"

' типа String (строка)

usertitle = "qwerty1234"

'инициализация переменной

' "usertitle" значением "qwerty1234"

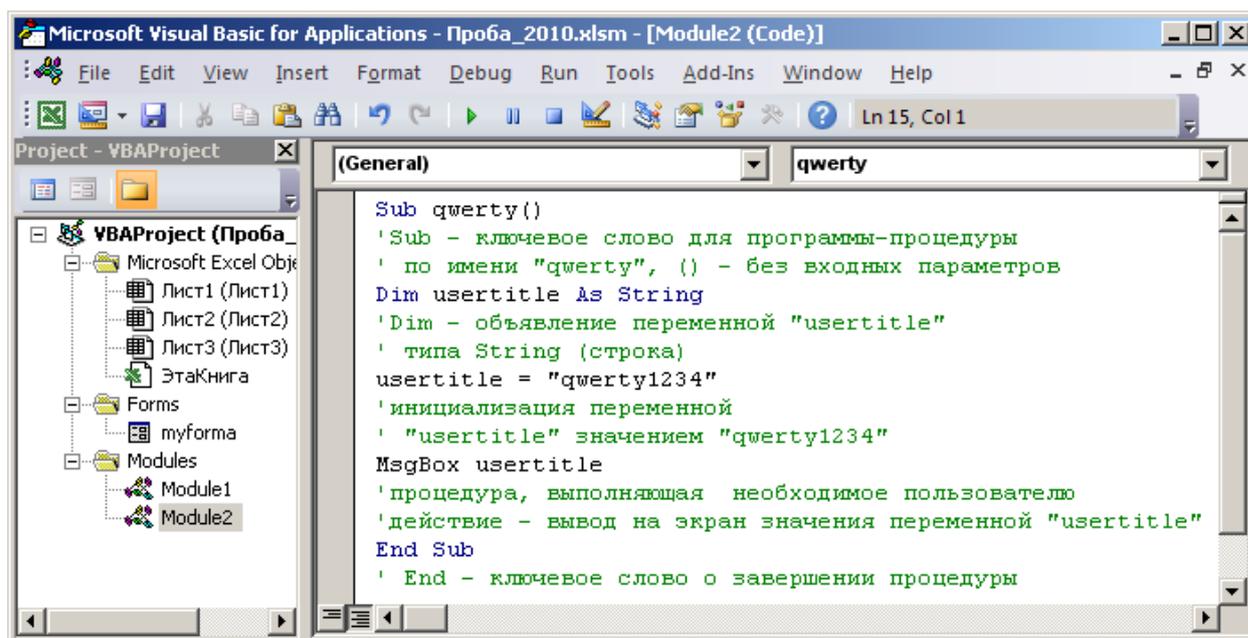
msgBox usertitle

'процедура, выполняющая необходимое пользователю

'действие – вывод на экран значения переменной "usertitle"

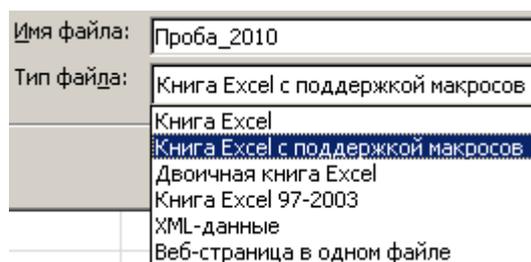
End Sub

' End – ключевое слово о завершении процедуры



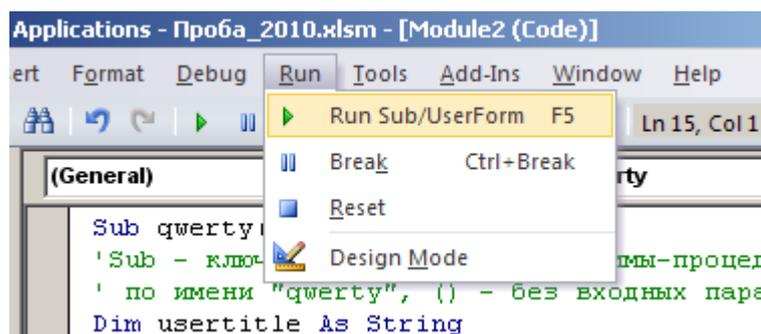
В оболочке VB MS Office имеется команда для сохранения программы в составе документа соответствующего приложения (**File (Файл) > Save... (Сохранить...)**) или отдельного компонента – модуля VB для последующего использования (**File (Файл) > Export File (Экспорт)**).

При сохранении документа MS Office, содержащего программные модули на языке VB, необходимо выбирать вариант типа файла *с поддержкой макросов*.



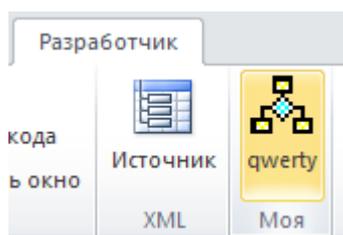
Запуск программы производится несколькими методами:

1) в оболочке VB команда **Run (Запуск) > Run Sub/UserForm (Запуск Процедуры/Пользовательской формы)**;



2) в оболочке соответствующего приложения MS Office: **Разработчик > Код > Macros (Макросы) > Run (Выполнить)**;

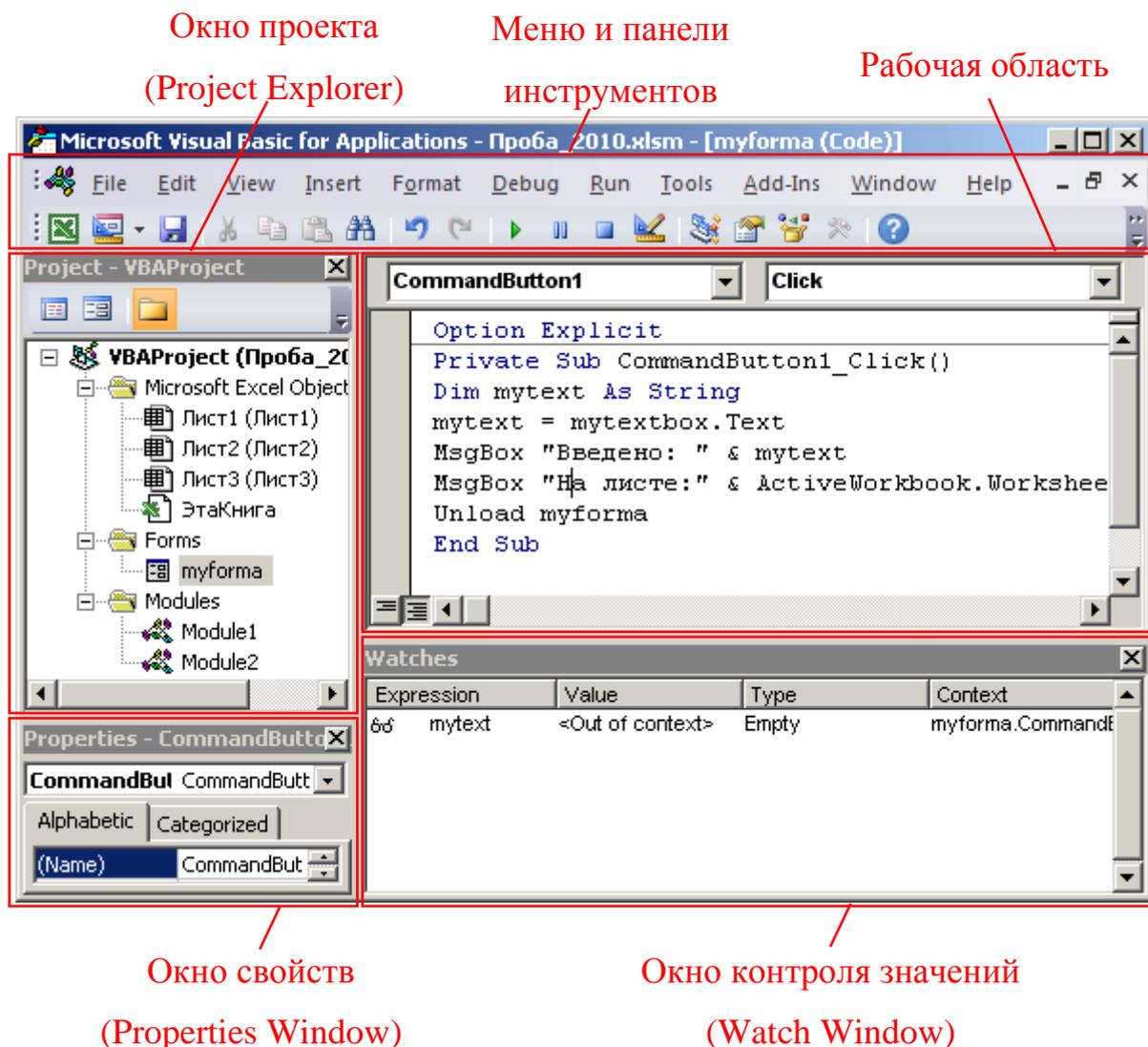
3) в оболочке соответствующего приложения MS Office по нажатию созданной кнопки на ленте (контекстная команда ленты **Настройка ленты...**).



Оболочка VB состоит из следующих основных частей:

- 1) панель меню (содержит все команды среды программирования VB);
настраиваемые панели инструментов (тематические наборы часто используемых команд);
- 2) рабочая область (ввод и редактирование текста программы);
- 3) вспомогательные окна: свойств объектов (**Properties Window**), состава программного проекта (**Project Explorer**), доступных программных

компонентов (*Object Browser*), отладка (*Immediate Window*), контроля значений выражений при выполнении программы (*Watch Window*).



Панель меню содержит подменю:

- 1) **File (Файл)**: сохранение, экспорт, импорт программных компонентов, печать форм и текстов программных модулей, выход в приложение-контейнер (компонент MS Office);
- 2) **Edit (Редактирование)**: команды редактирования текста программ (операции с буфером обмена, возврат действий, табулирование текста, закладки, контекстные справки и шаблоны синтаксиса);
- 3) **View (Вид)**: переключение между окнами кода и объектов (форм), переход в окно приложения-контейнера, отображение структуры проекта VB (модулей, форм) и доступных программных библиотек и их компонентов,

настройка панелей инструментов оболочки VB, отображение окон отладки (переменных, операторов);

4) **Insert (Вставка)**: добавление в программу процедур, функций, свойств, модулей, форм, файлов других проектов;

5) **Format (Формат)**: редактирование форм (расположение и размеры объектов);

6) **Debug (Отладка)**: проверка синтаксиса программ, выполнение программ по шагам (строкам кода), по точкам прерывания (остановки), просмотр текущих значений выражений при выполнении (контрольные значения);

7) **Run (Запуск)**: запуск выполнения программных модулей или диалогов, прерывание и сброс выполнения, переход в режим разработки;

8) **Tools (Инструменты)**: работа с макросами, загрузка дополнительных программных библиотек, настройка параметров оболочки и проекта VB;

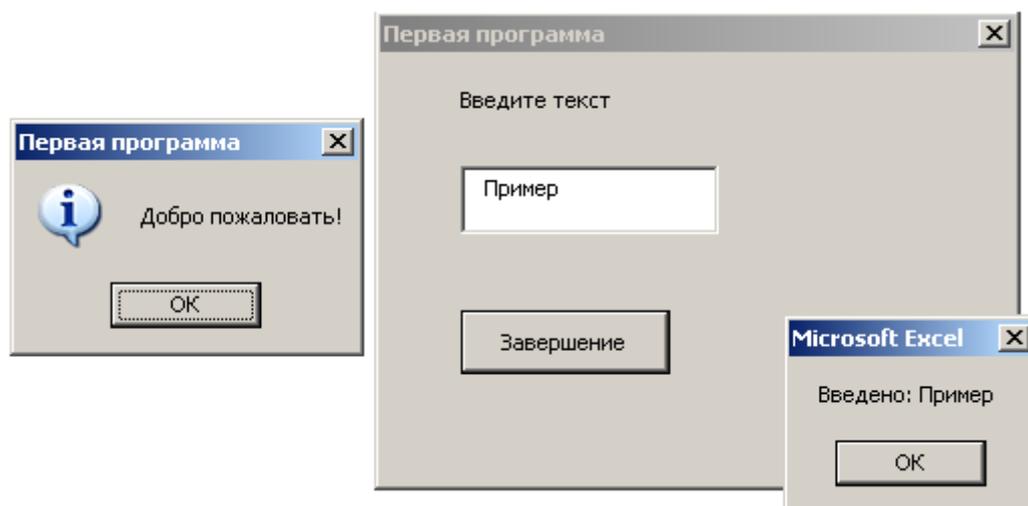
9) **Add-Ins (Надстройки)**: добавление системных объектов в проект;

10) **Window (Окно)**: управление расположением окон VB;

11) **Help (Помощь)**: справка по синтаксису VB.

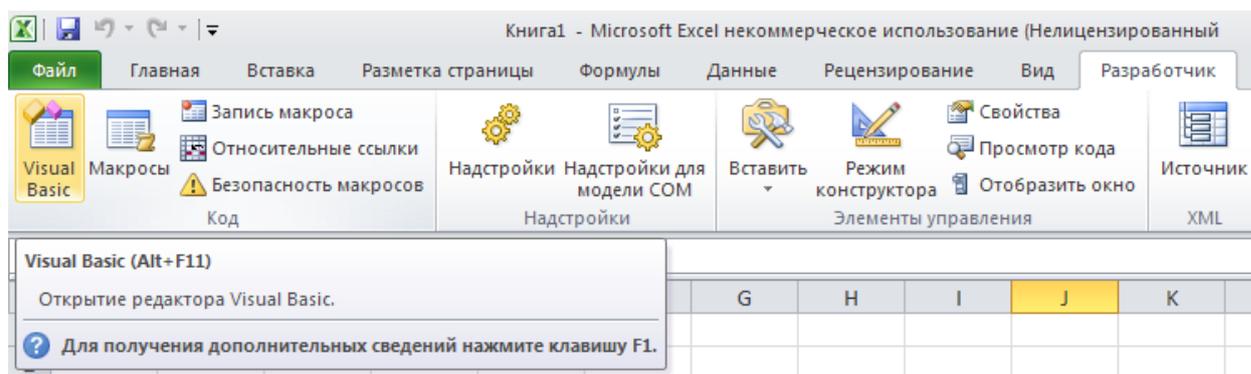
Задание

Итогом выполнения этой работы должна быть программа на VB, запускаемая в созданном пользователем файле электронных таблиц MS Excel. Программа будет выводить на экран сообщение о начале своей работы, а затем выводить диалоговое окно с полем ввода и кнопкой.

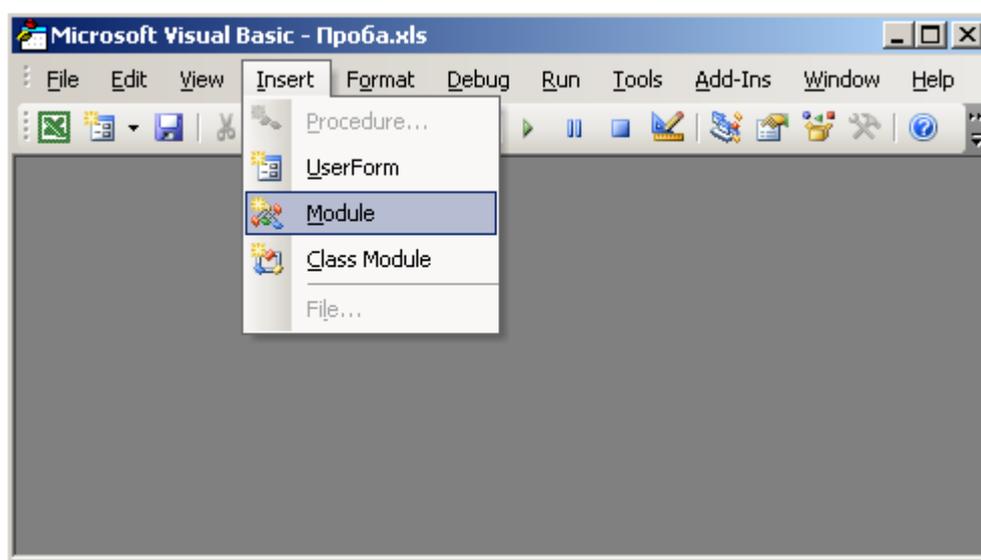


При нажатии кнопки программа отобразит значение, введенное пользователем в поле ввода, и закроет диалоговое окно.

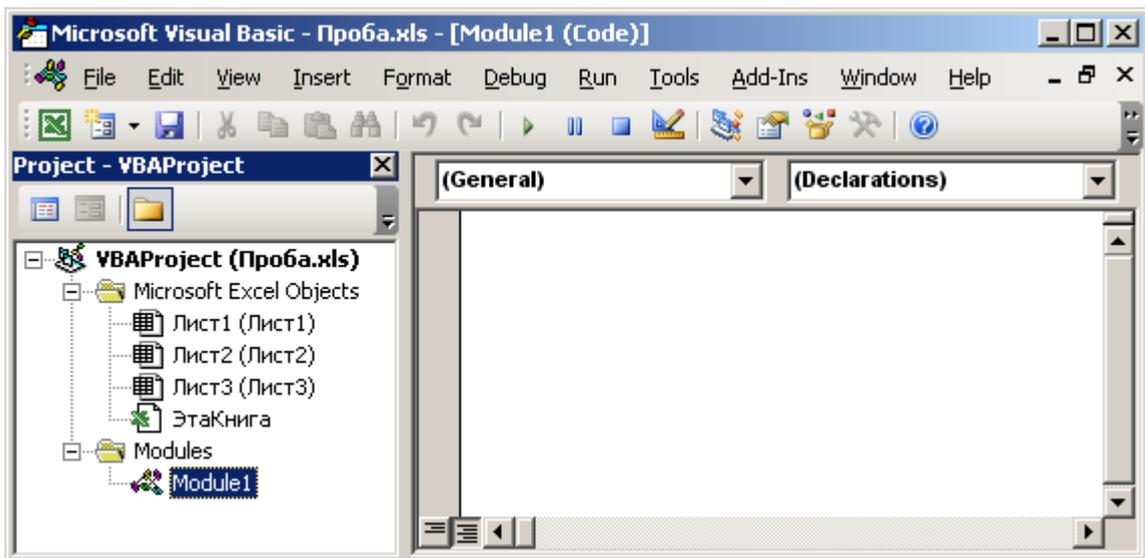
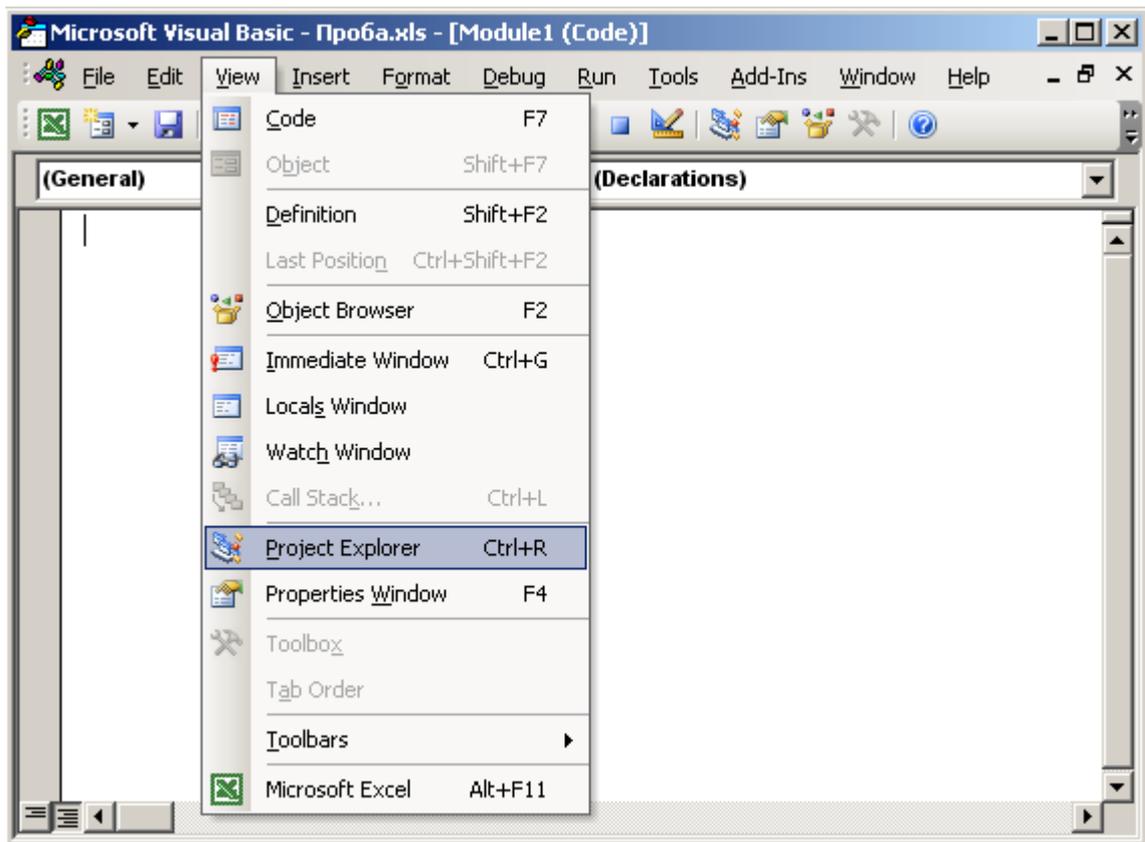
1. Открыть MS Excel, создать и сохранить электронную таблицу.
2. Отобразить вкладку **Разработчик** (контекстное меню ленты > **Настройка ленты...** > **Основные вкладки** > **Разработчик** > **ОК**) и запустить среду VBA: **Разработчик** > **Код** > **Visual Basic**.



3. Добавить программный модуль VB: **Вставка (Insert)** > **Module**.



4. Открыть (если оно скрыто) окно проекта (**Project-VBA Project**): **Вид (View)** > **Окно проекта (Project Explorer)** и найти добавленный модуль в структуре текущего файла **Excel (VBAProject (...xls))** > **Модули (Modules)** > **Модуль1 (Module1)**.



5. Дважды щелкнуть на имени модуля в окне проекта и в открывшемся окне (окне программы) набрать текст процедуры:

Option Explicit

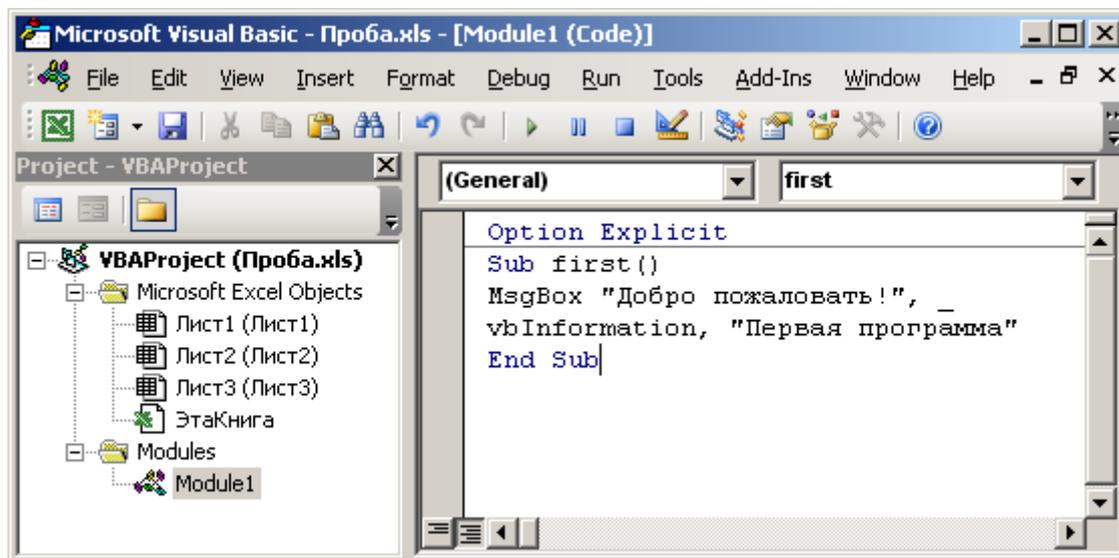
' Инструкция для обязательного объявления переменных

Sub first()

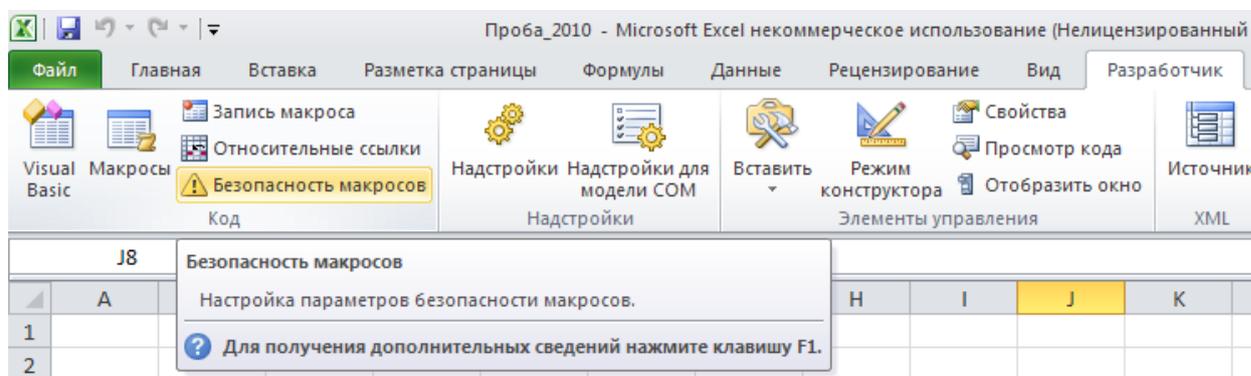
' Начало процедуры-программы по имени first

MsgBox "Добро пожаловать!", vbInformation, "Первая программа"

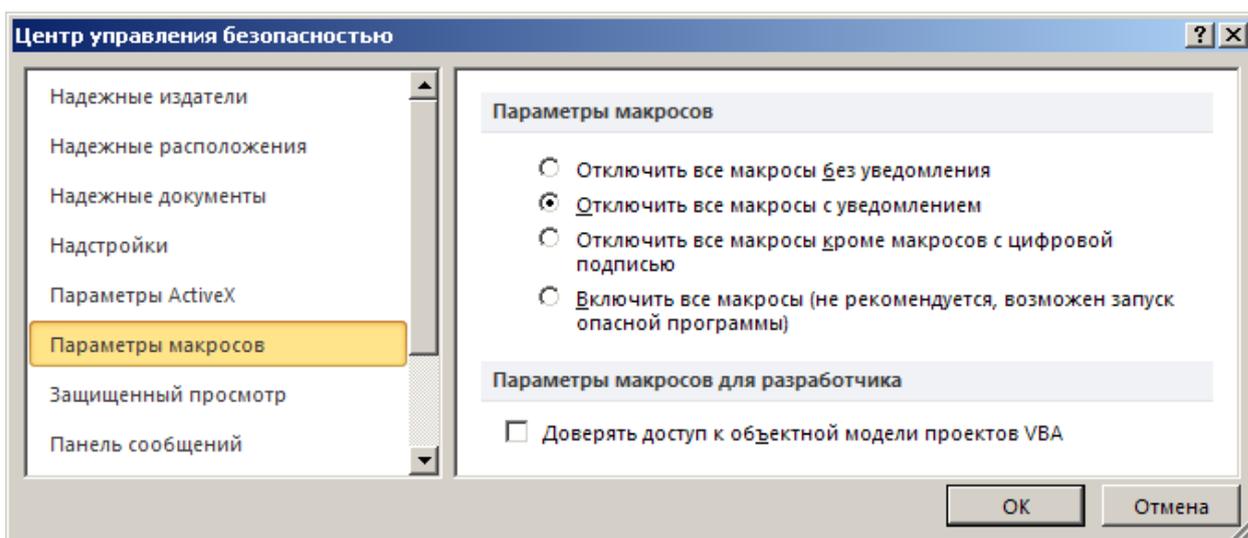
- ' Вывод окна-сообщения с текстом "Добро пожаловать"
- ' и заголовком "Первая программа"
- End Sub
- ' Завершение процедуры-программы по имени first



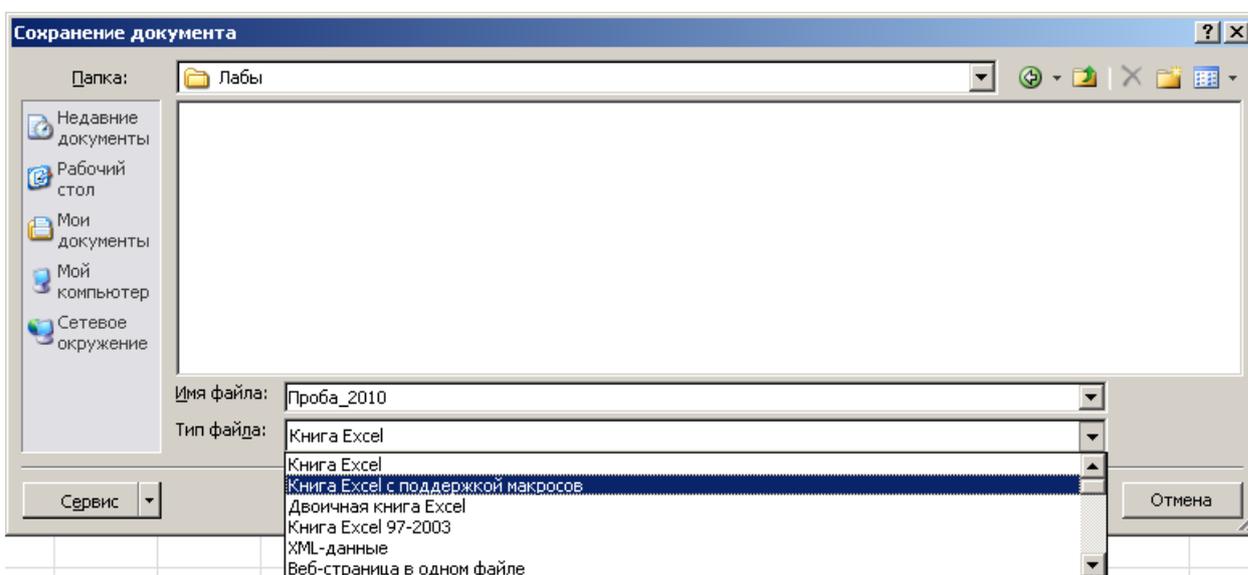
6. Перейти в MS Excel командой *Вид (View) > Microsoft Excel* и проверить уровень безопасности при запуске программ-макросов командой *Разработчик > Код > Безопасность макросов*.



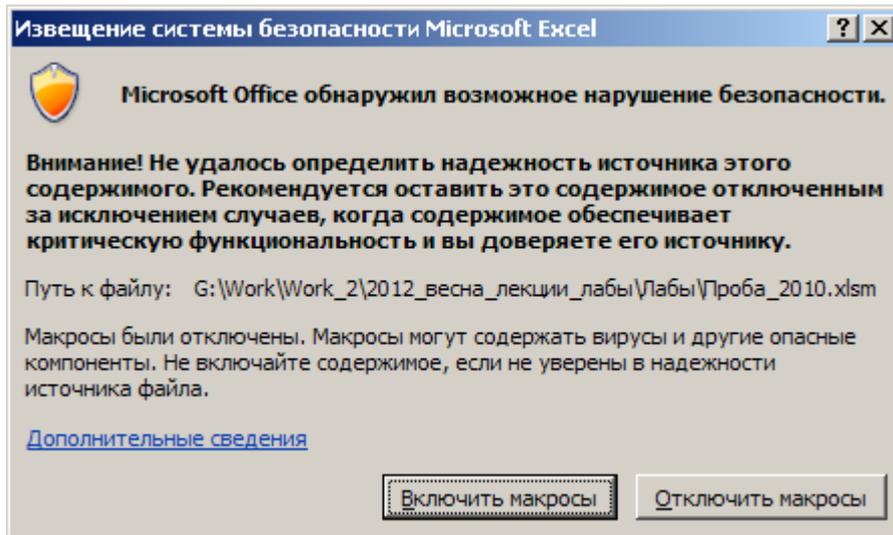
Установить (если он не установлен) режим *Отключить все макросы с уведомлением (Решение о запуске потенциально-опасных макросов принимается пользователем)*, нажать *ОК*.



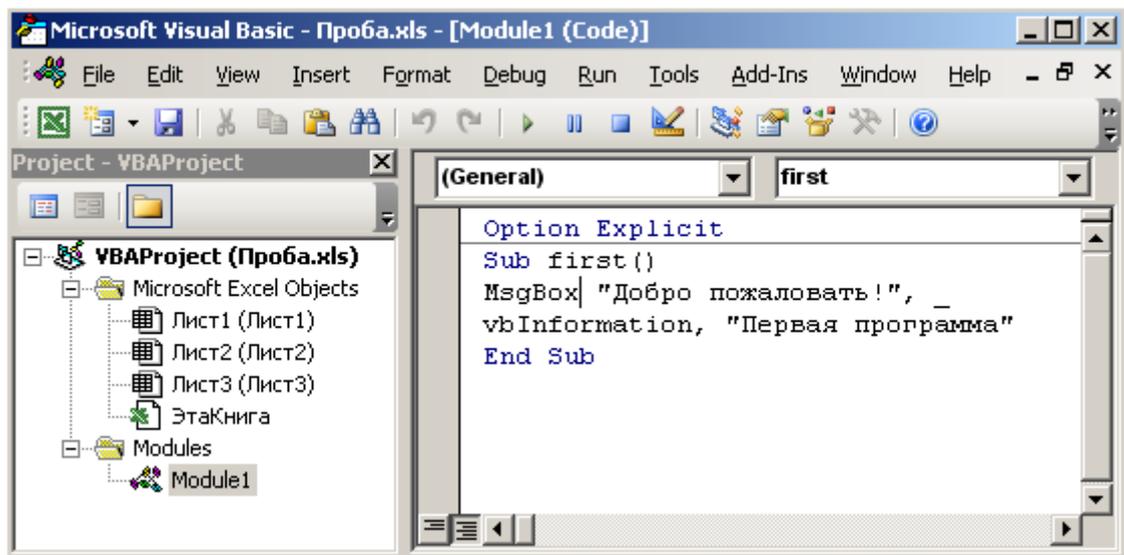
Сохранить файл (*Файл > Сохранить*), выбрав тип файла *...с поддержкой макросов*.



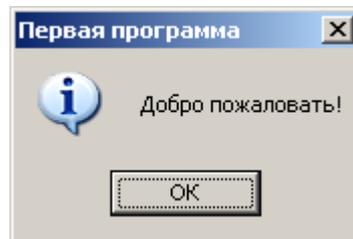
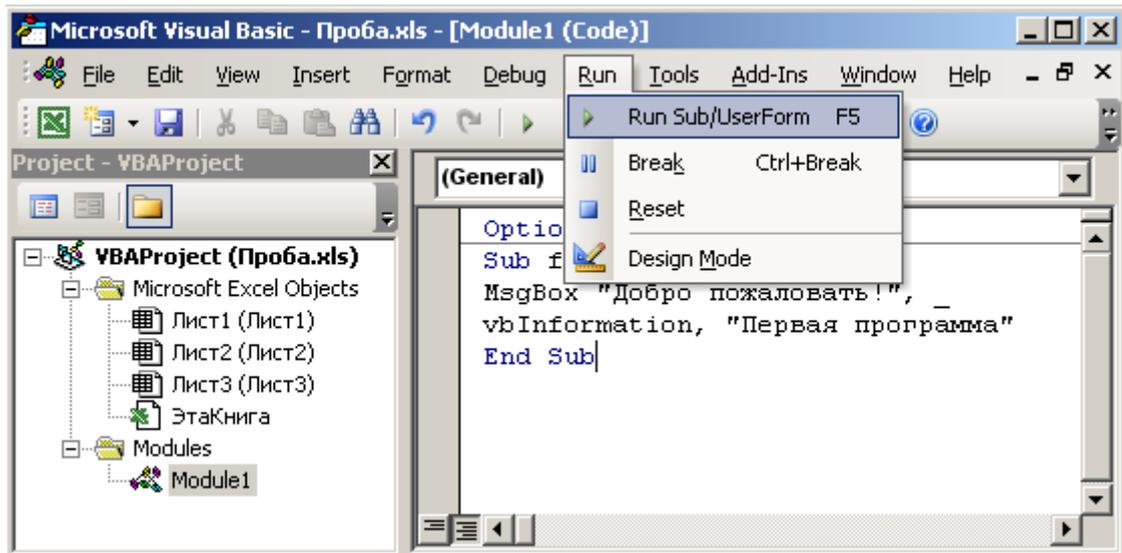
Закрывать файл (*Файл > Закрывать*) и повторно открывать (*Файл > Открыть* или *Файл > Последние > первый файл в списке последних открытых*). На *Извещение системы безопасности* при открытии файла отвечать *Включить макросы*.



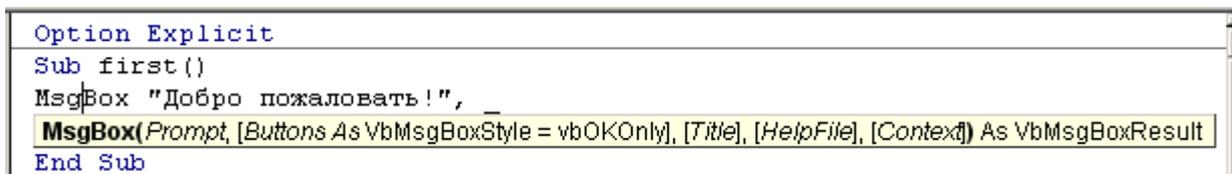
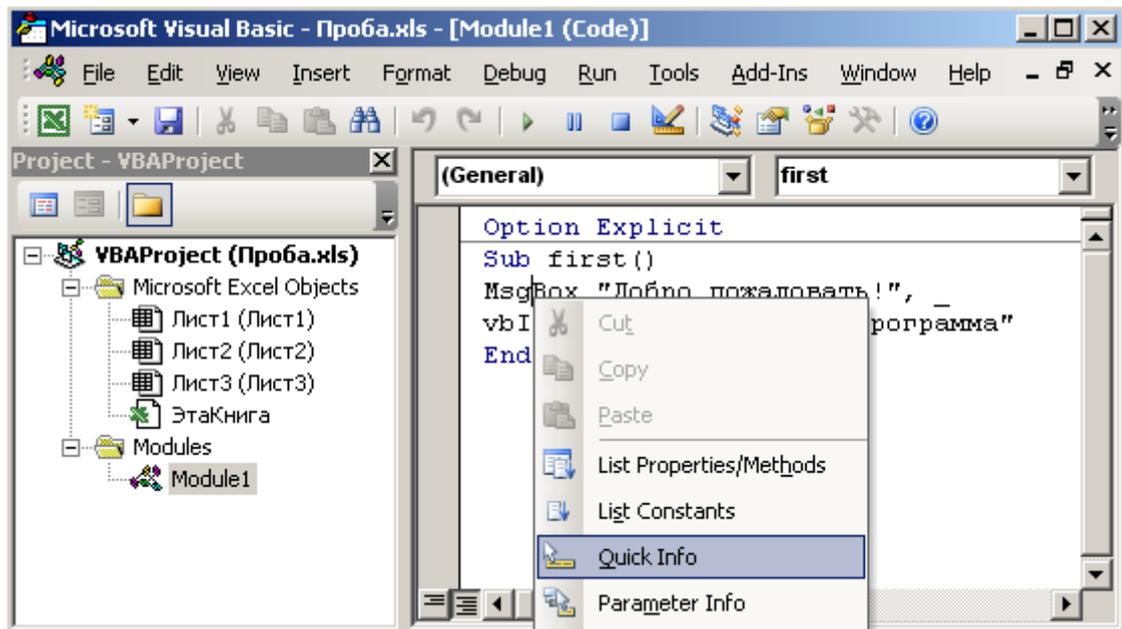
7. Открыть среду VBA (*Сервис > Макрос > Редактор Visual Basic*) и перейти в модуль **Module1**, щелкнув по нему в *Окне проекта (Project Explorer)*. Установить курсор в любом месте процедуры **first (Sub first()...End Sub)**.



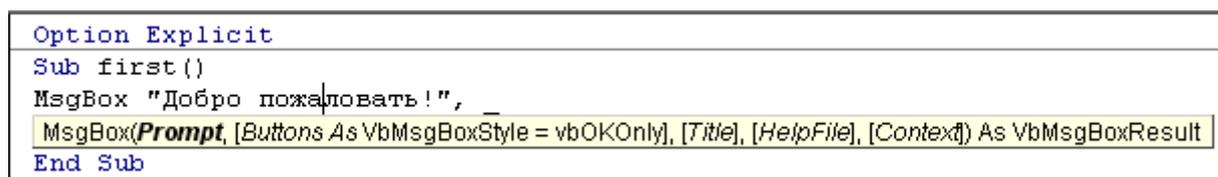
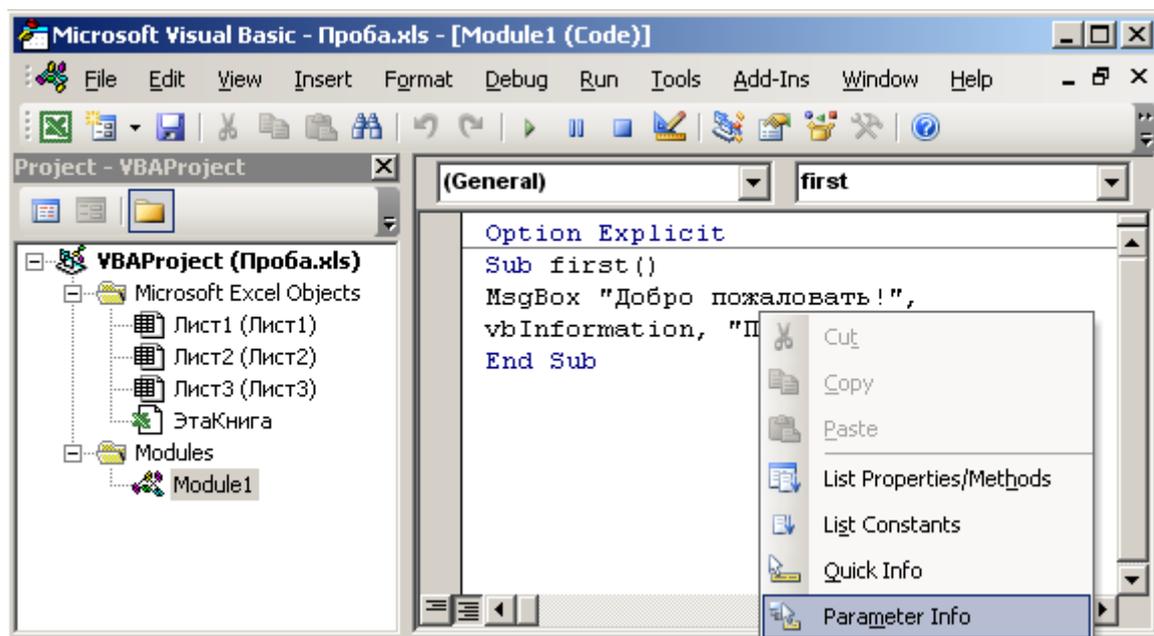
8. Сохранить файл (*Файл (File) > Сохранить... (Save...)*) и запустить модуль с процедурой **first** на выполнение командой: *Запуск (Run) > Запуск подпрограммы (Run Sub или F5)*.



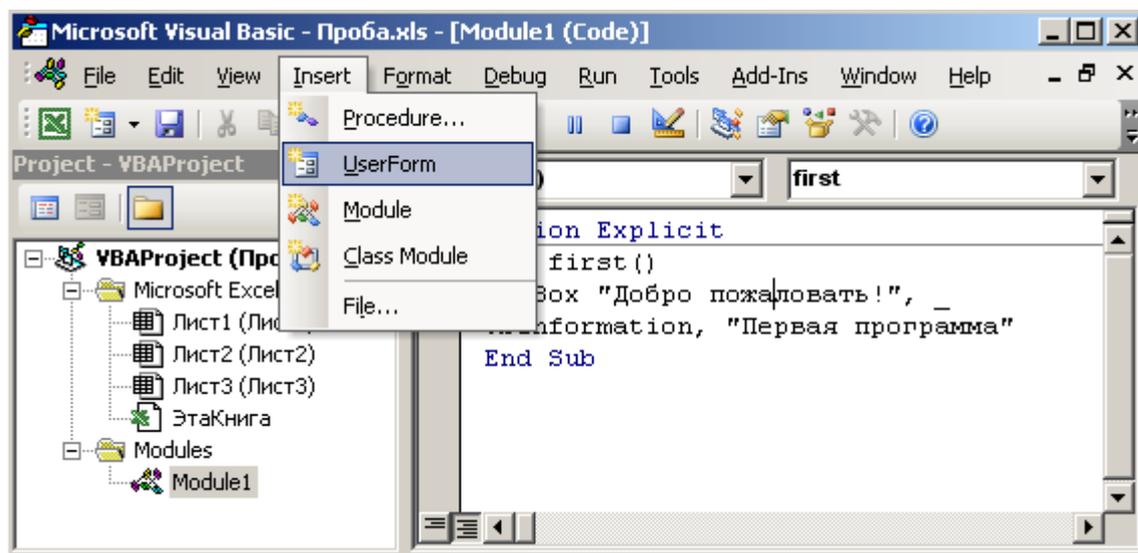
9. После завершения программы (нажатие *OK* в окне *Первая программа*) в окне модуля установить курсор на ключевом слове **MsgBox** и выполнить команду: *Правка (Edit) > Сведения (Quick Info)* – информация о синтаксисе функции или процедуры.



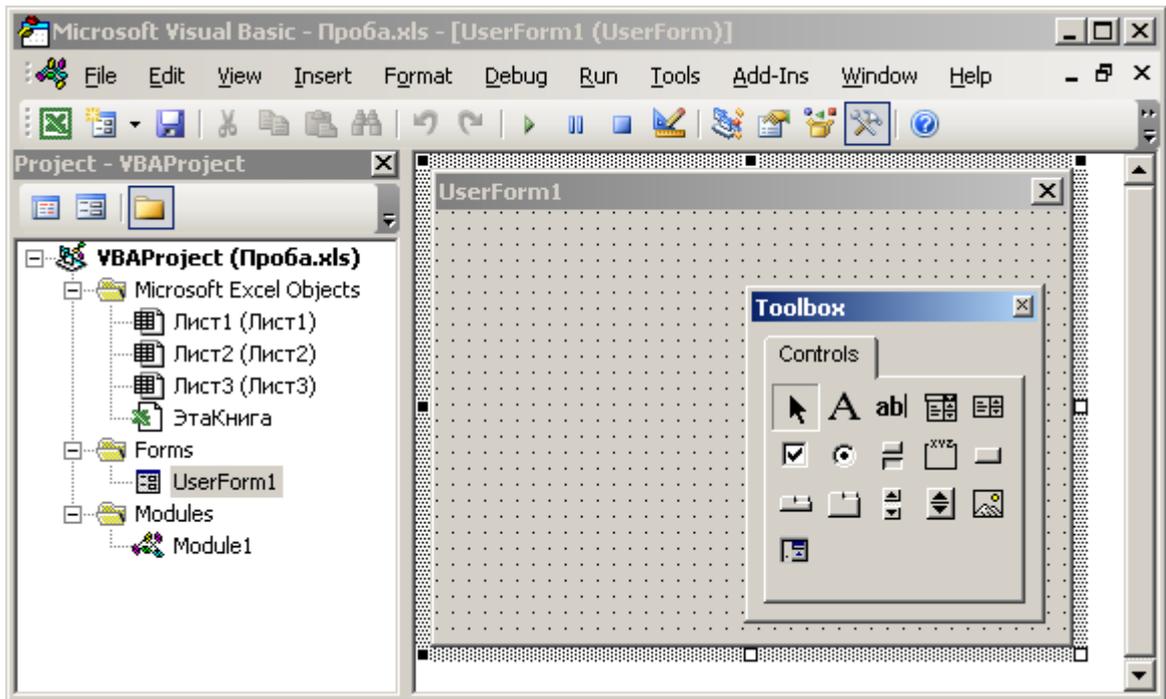
10. Установить курсор на тексте **"Добро пожаловать"** и выполнить команду: *Правка (Edit) > Параметры (Parameter Info)* – информация о текущем параметре функции или процедуры.



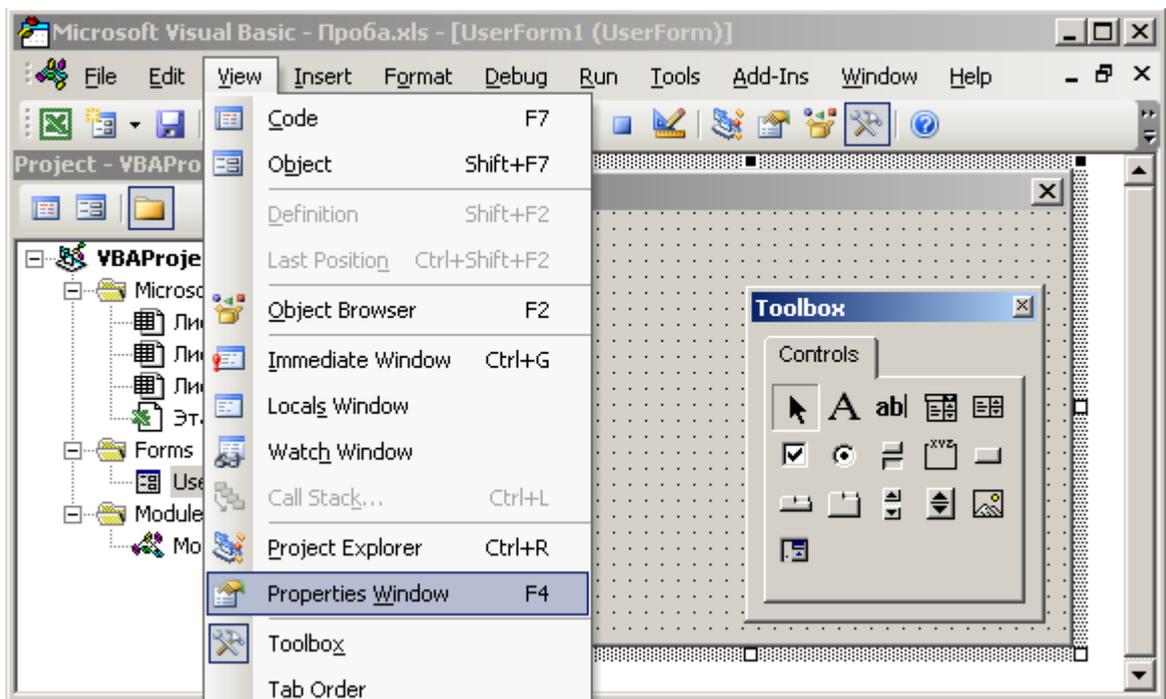
11. Добавить к проекту окно формы командой: *Вставка (Insert) > UserForm*.



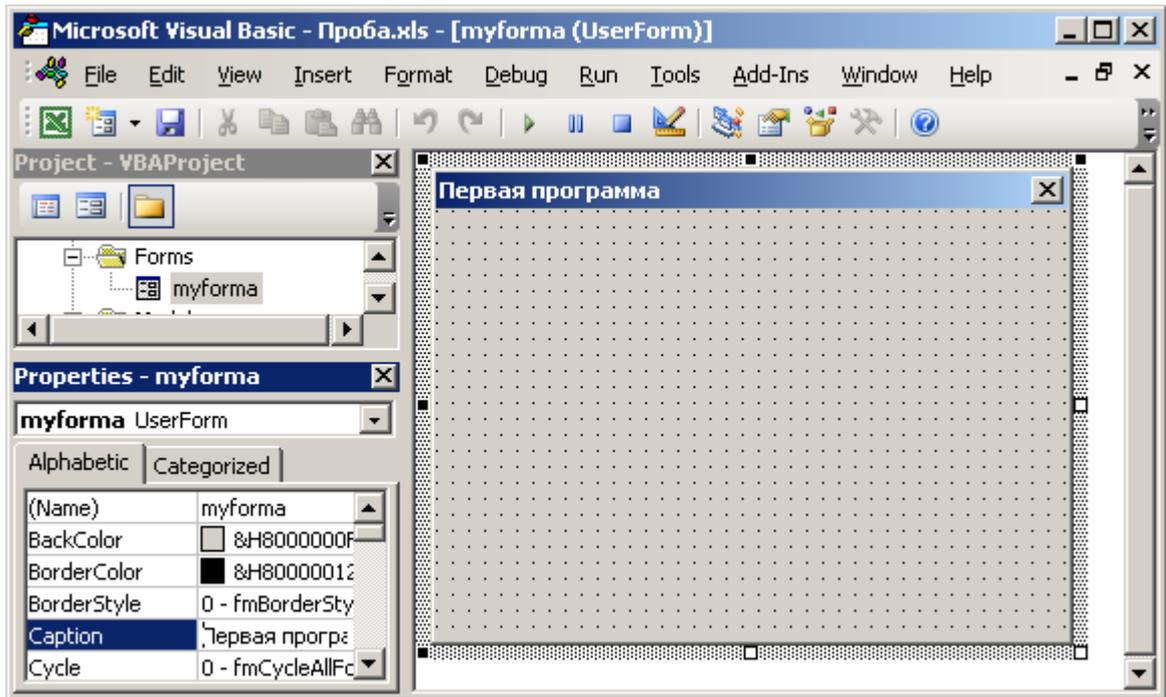
В проект будет добавлена новая форма, окно редактирования которой выводится на экран. В окне проекта будет добавлена группа **Формы (Forms)** с новой формой **UserForm1**.



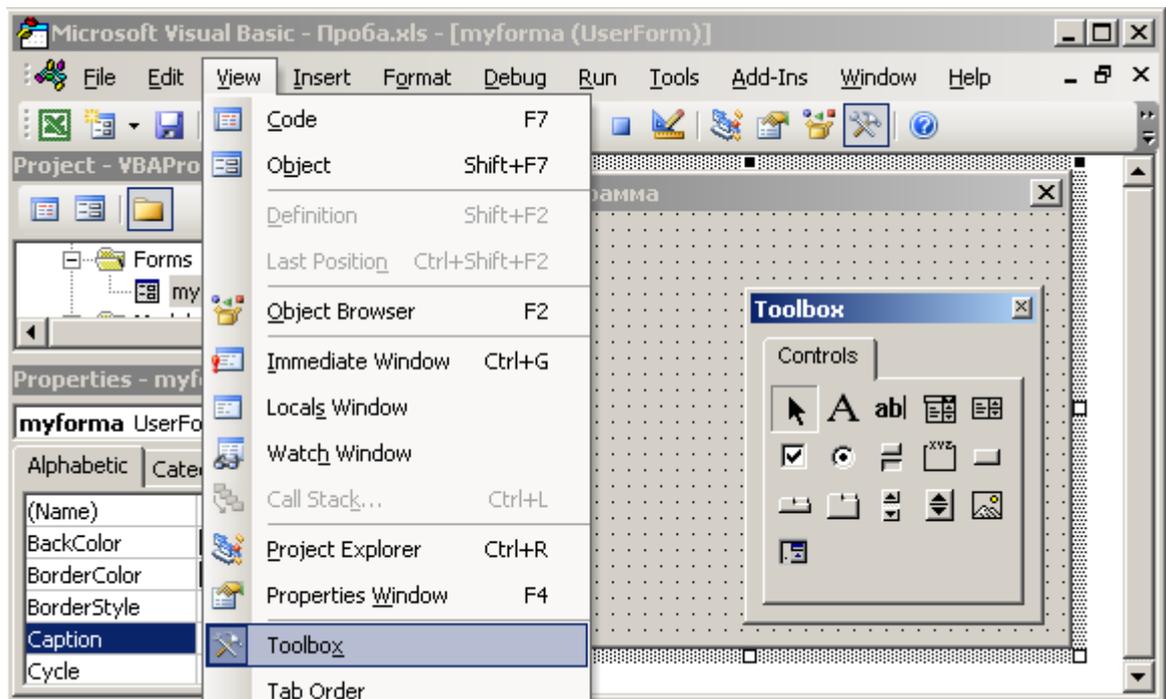
12. Открыть (если оно скрыто) окно свойств (*Properties*): *Вид (View) > Окно свойств (Properties Window)* для добавленной формы.



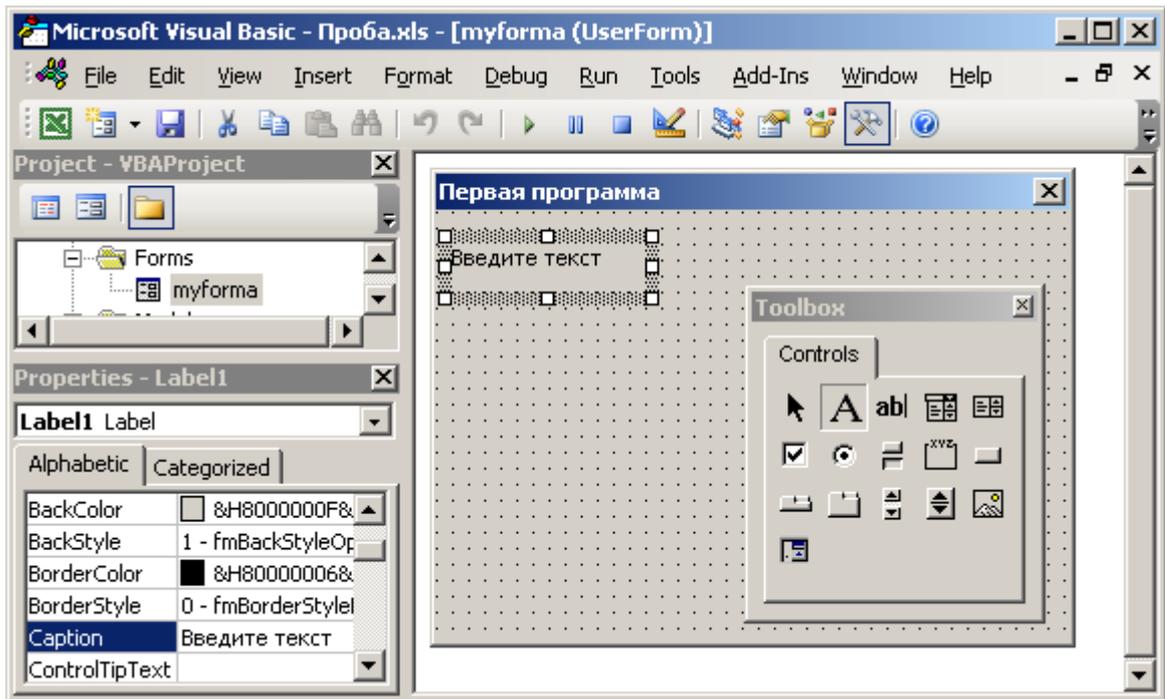
13. В окне свойств в поле *Name* ввести: **myforma** (в окне проекта изменится имя формы), а в поле *Caption* ввести: **Первая программа** (в окне формы изменится ее заголовок).



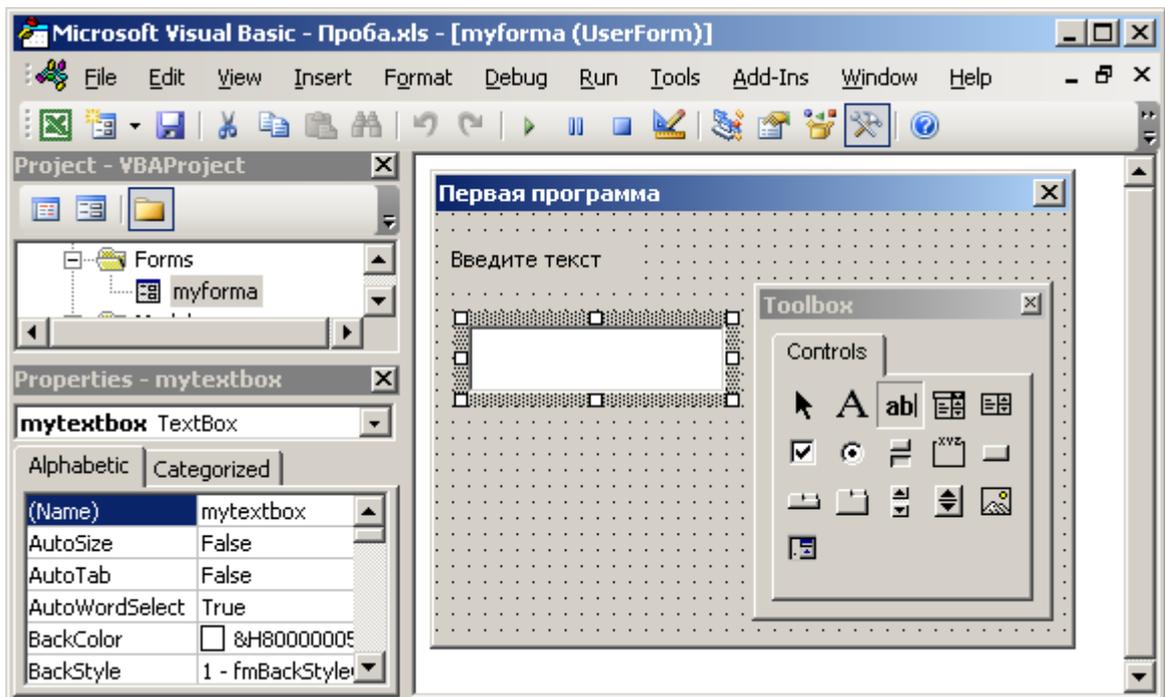
14. Щелчком мыши перейти в окно формы, активировав панель элементов (*ToolBox*). Если *ToolBox* был закрыт, его можно вызвать командой: *Вид (View) > Панель элементов (ToolBox)*.



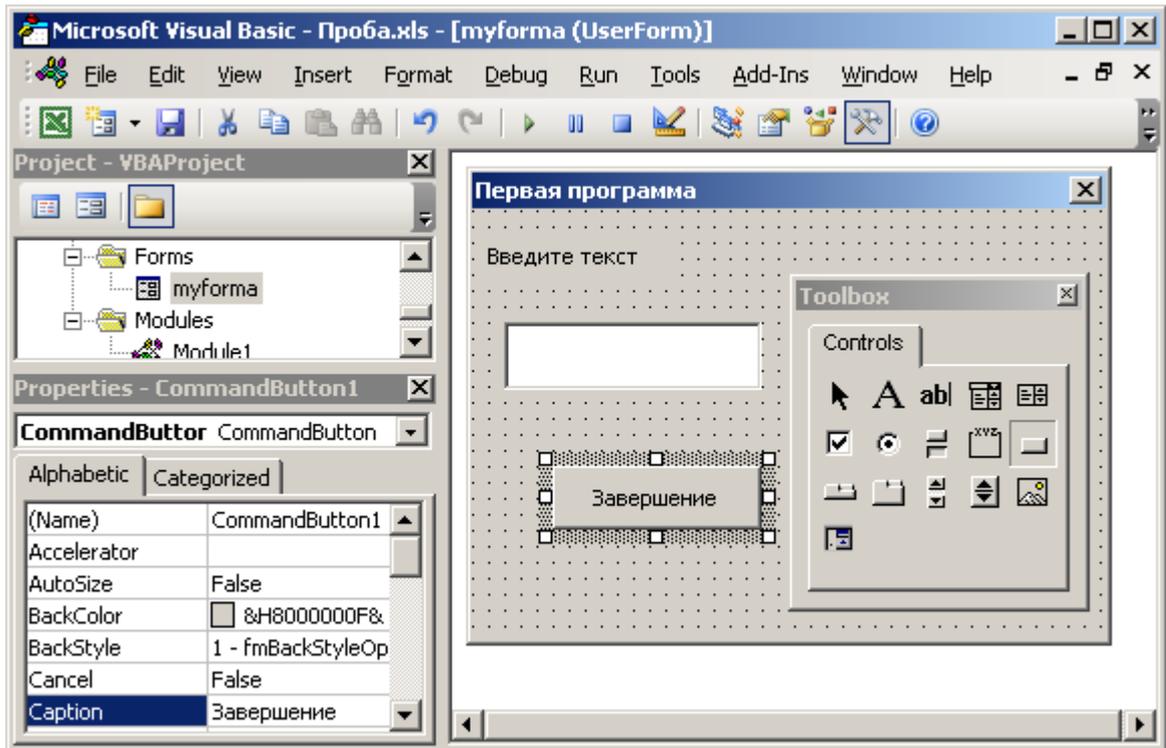
15. На панели элементов щелкнуть на кнопке *Надпись (Label)* и мышью "нарисовать" в форме контур элемента управления. В окне свойств для созданной надписи в поле *Caption* набрать: **Введите текст.**



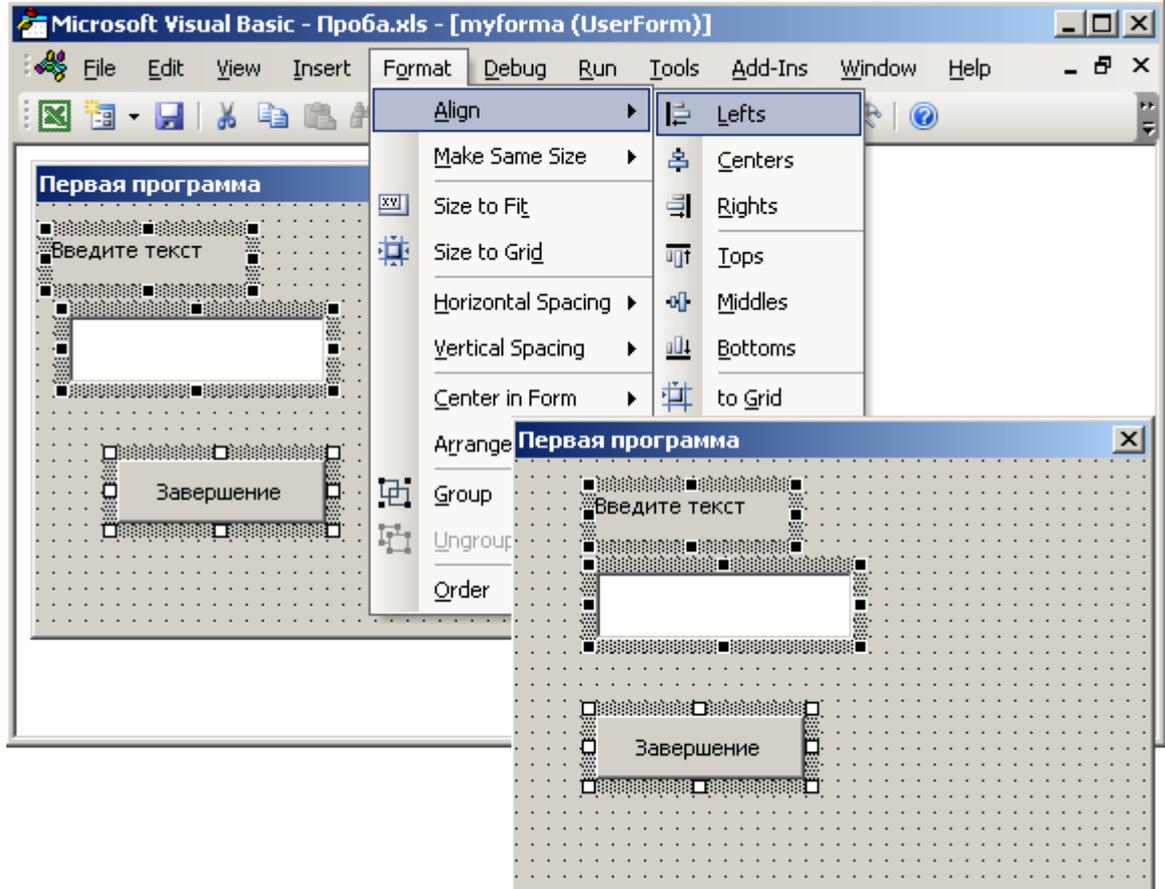
16. На панели элементов щелкнуть на кнопке *Поле (TextBox)* и добавить поле для ввода текста в форму (под надписью). В окне свойств для созданного поля в свойстве *Name* набрать: **mytextbox**.



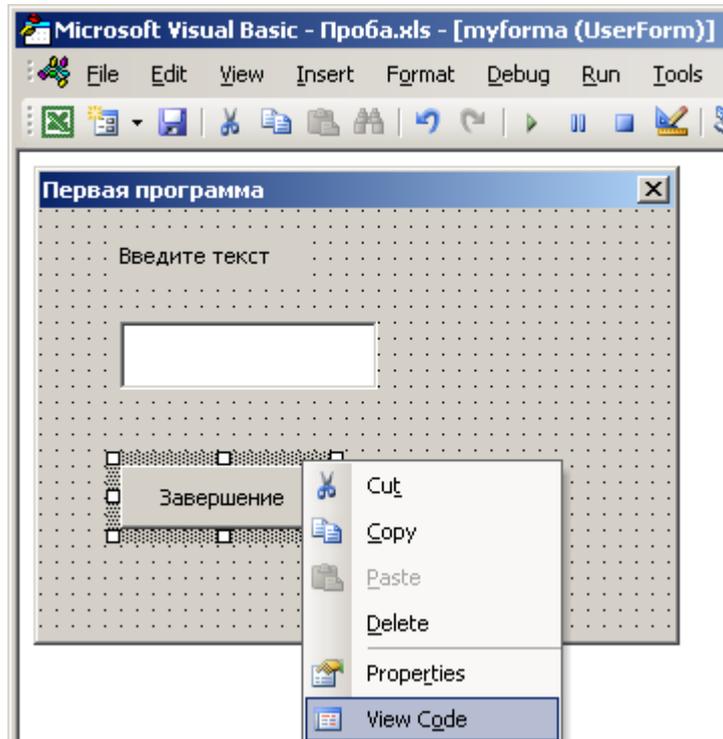
17. На панели элементов щелкнуть на кнопке *Кнопка (CommandButton)* и добавить управляющую кнопку в форму (под текстовым полем). В окне свойств для созданной кнопки в свойстве *Caption* набрать: **Завершение**.



18. Выделить все добавленные элементы в форме (надпись, текстовое поле и кнопку) и применить к ним команду: **Формат (Format) > Выровнять (Align) > По левому краю (Left)**.



19. Выделить в форме добавленную кнопку и вызвать программу для обработки связанных с ней действий командой: **Вид (View) > Программа (Code)** (или **окно проекта > кнопка Программа (View Code)**).



20. В окне программы в открывшемся шаблоне набрать:

Option Explicit

' Инструкция для обязательного объявления переменных

Private Sub CommandButton1_Click()

' Начало процедуры-программы, выполняемой при щелчке

' (клике) по кнопке по имени **CommandButton1**

Dim mytext As String

' Объявление (создание) переменной по имени **mytext**

' для хранения данных типа "строка" (**String**)

mytext = mytextbox.Text

' Запись текста из поля по имени **mytextbox** в переменную

' **mytext**

MsgBox "Введено: " & mytext

' Вывод окна-сообщения со строкой, начинающейся с

' "Введено: " и заканчивающейся текстом из переменной

```
' mytext
```

```
MsgBox "На листе:" & ActiveWorkbook.Worksheets(1).Cells(1, 2)
```

```
' Вывод окна-сообщения со строкой, начинающейся с
```

```
' "На листе: " и заканчивающейся текстом-значением
```

```
' из ячейки B1 первого листа текущего файла MS Excel
```

```
Unload myforma
```

```
' Заккрытие формы по имени myforma
```

```
End Sub
```

```
' Завершение программы-процедуры
```

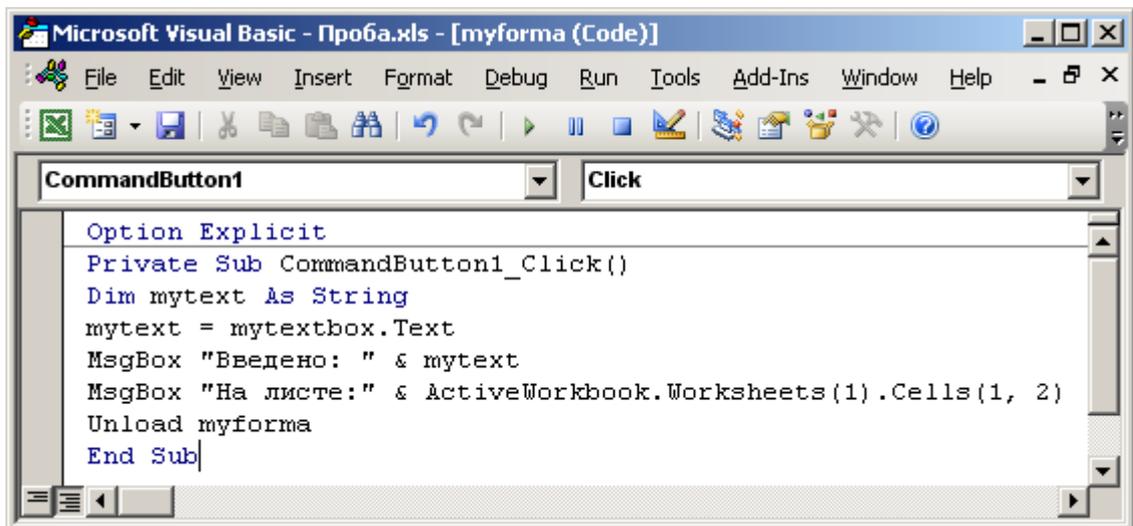
Шаблон процедуры для кнопки (**Private Sub CommandButton1_Click() ...**

End Sub) был добавлен автоматически. Процедура срабатывает при щелчке

(**Click**) по объекту – кнопке (**CommandButton1**). В окне программы

процедуры для различных объектов выбираются из двух списков (*Объект*

(*Object*) и *Процедура (Procedure)*).

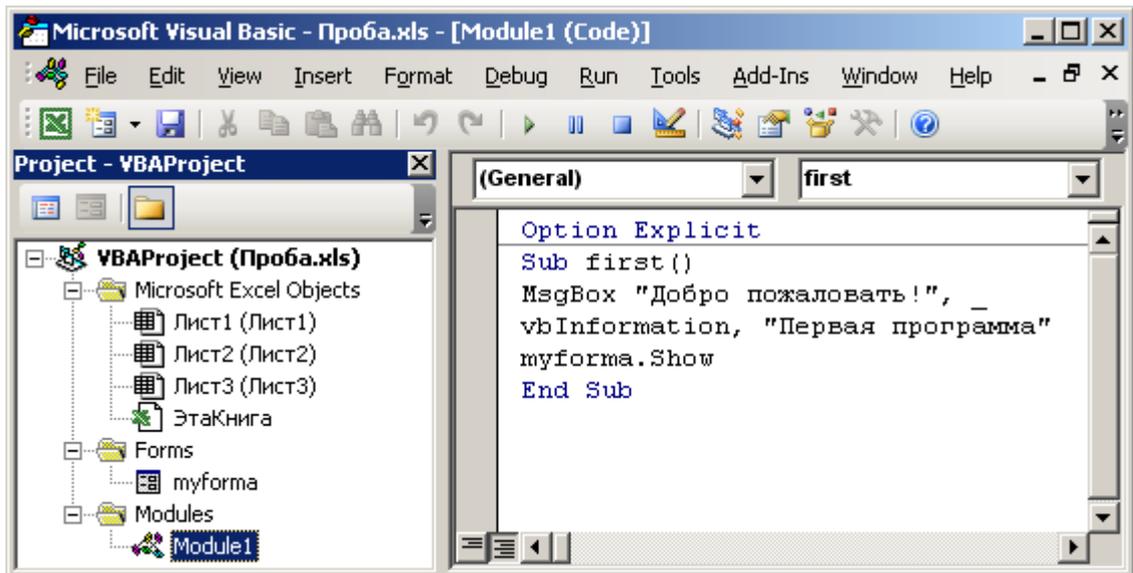


21. В окне программы для модуля (**Module1**) перед строкой

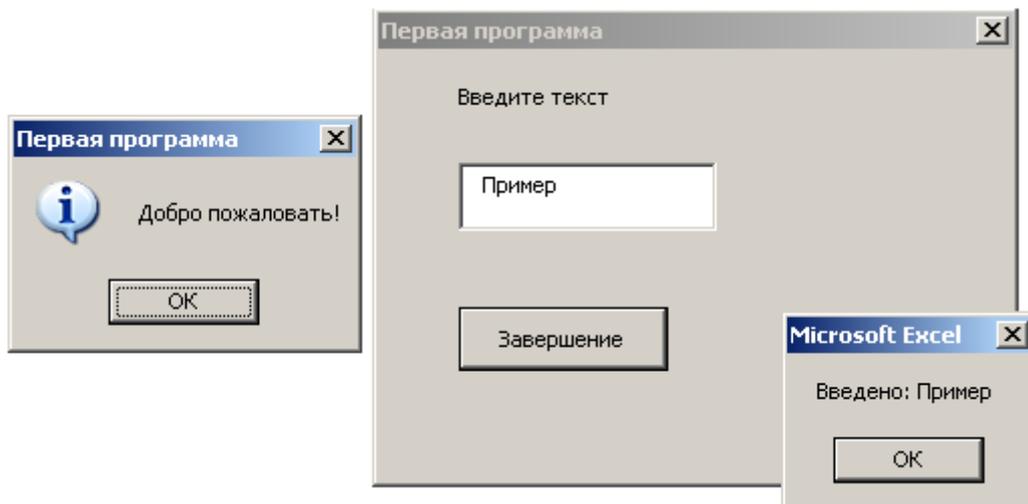
```
End Sub
```

добавить строку для отображения формы **myforma**

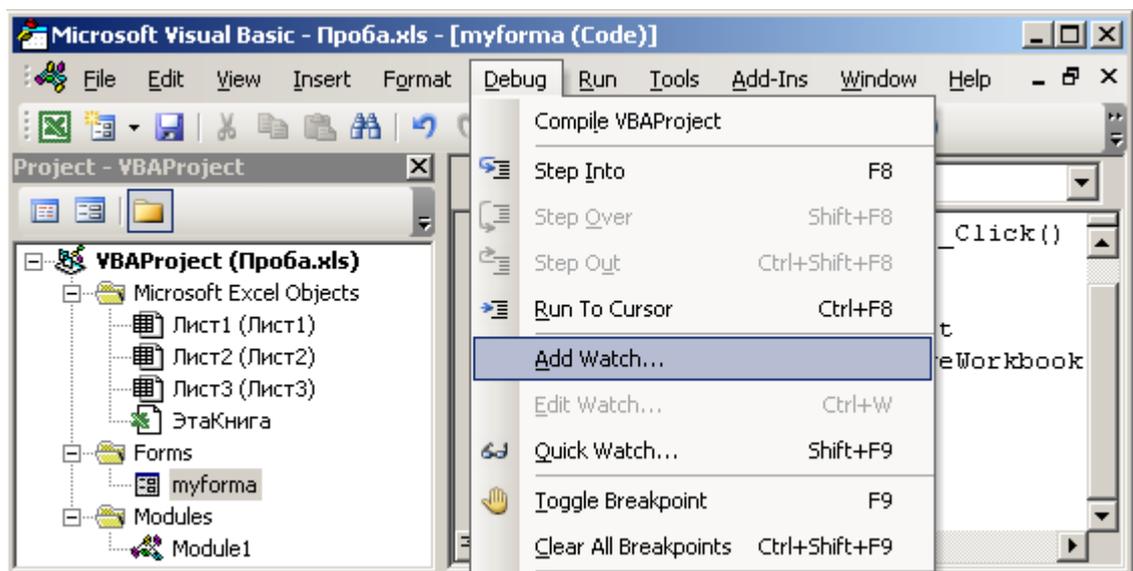
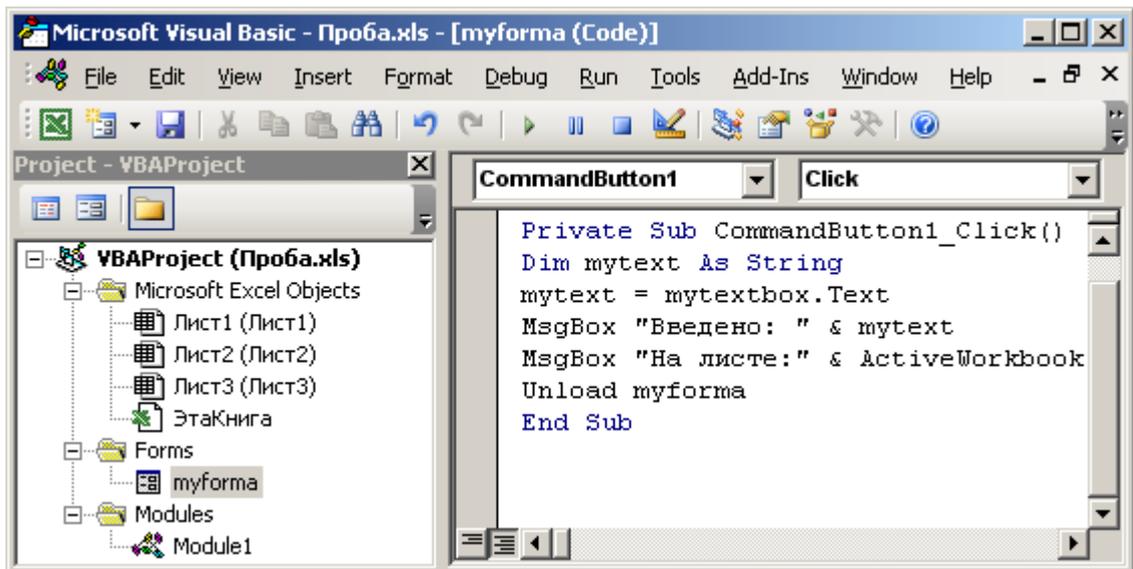
```
myforma.Show
```



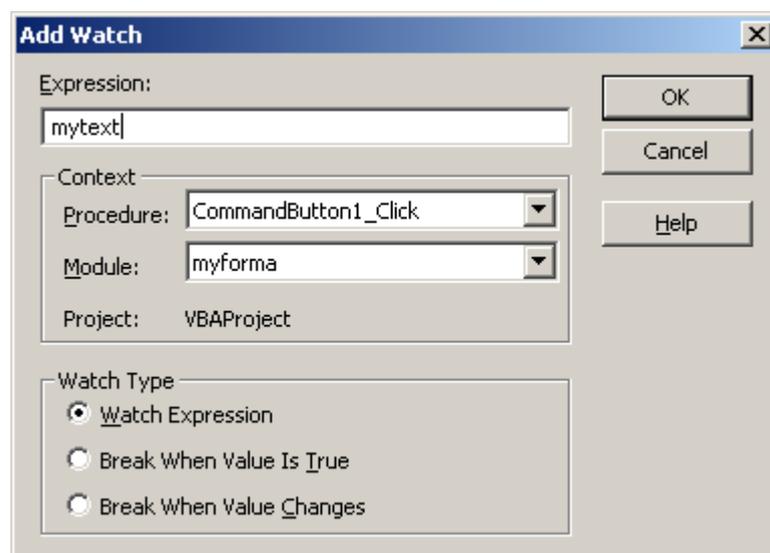
22. Сохранить файл, установить курсор в процедуру **first** и запустить модуль с процедурой **first** на выполнение (**F5**).



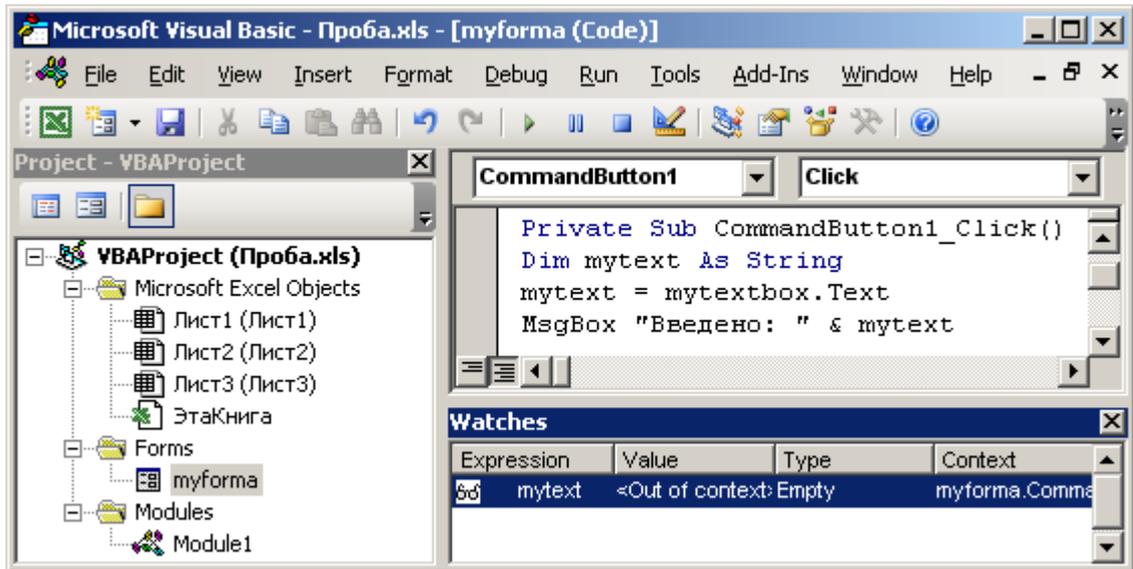
23. Перейти в окно программы для кнопки (**CommandButton1**) и выполнить команду: *Отладка (Debug) > Добавить контрольное значение (Add watch)*.



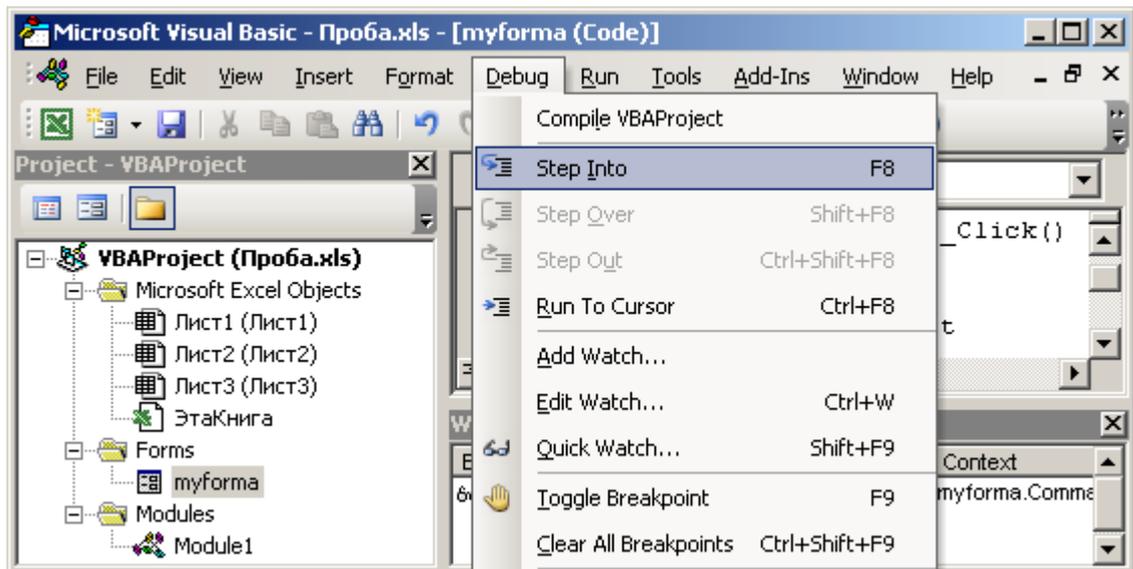
В открывшемся окне в поле **Выражение (Expression)** набрать имя переменной: **mytext**.



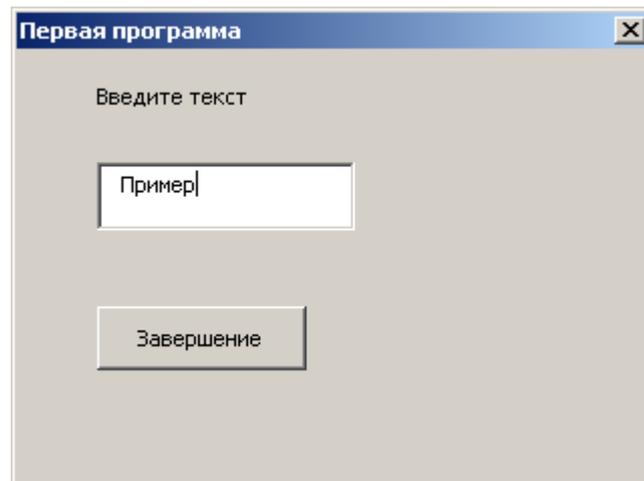
После нажатия **ОК** отобразится окно *Контрольное значение (Watch) (View) > Окно контрольного значения (Watch Window)*.



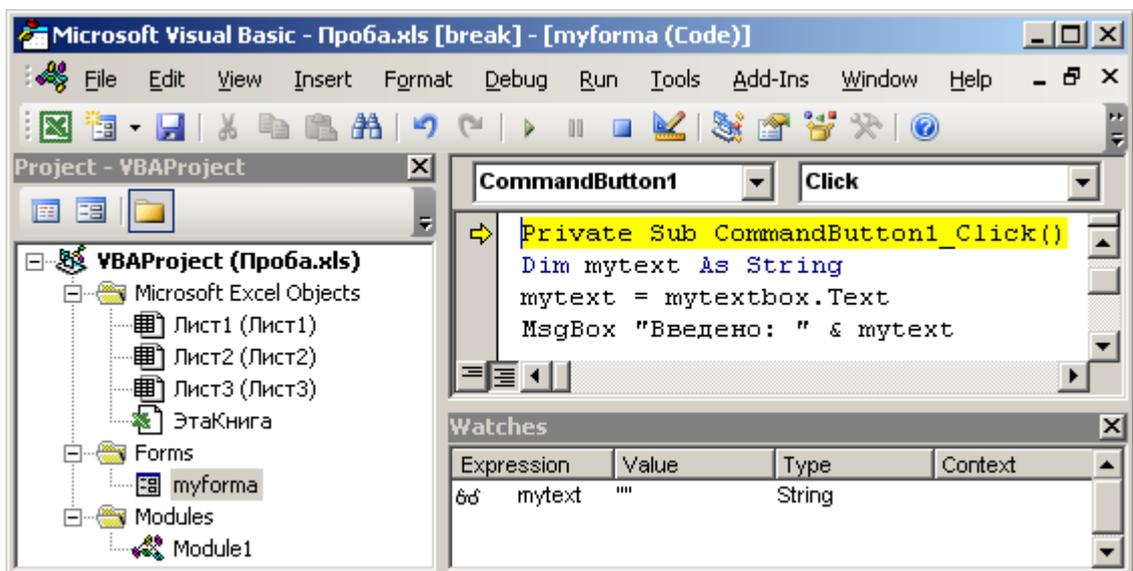
24. Запустить программу в режиме отладки (*Отладка (Debug) > Шаг с заходом (Step Into)* или **F8**).



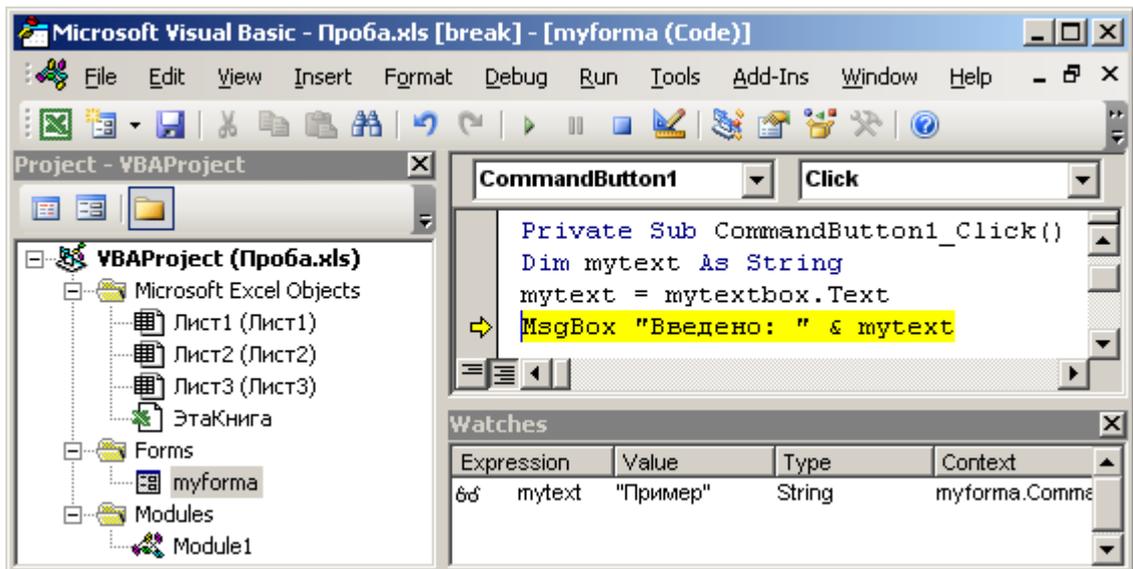
В запущенном окне диалога ввести текст и нажать кнопку **Завершение**.



В открывшемся окне VB желтым цветом отмечается текущая операция.

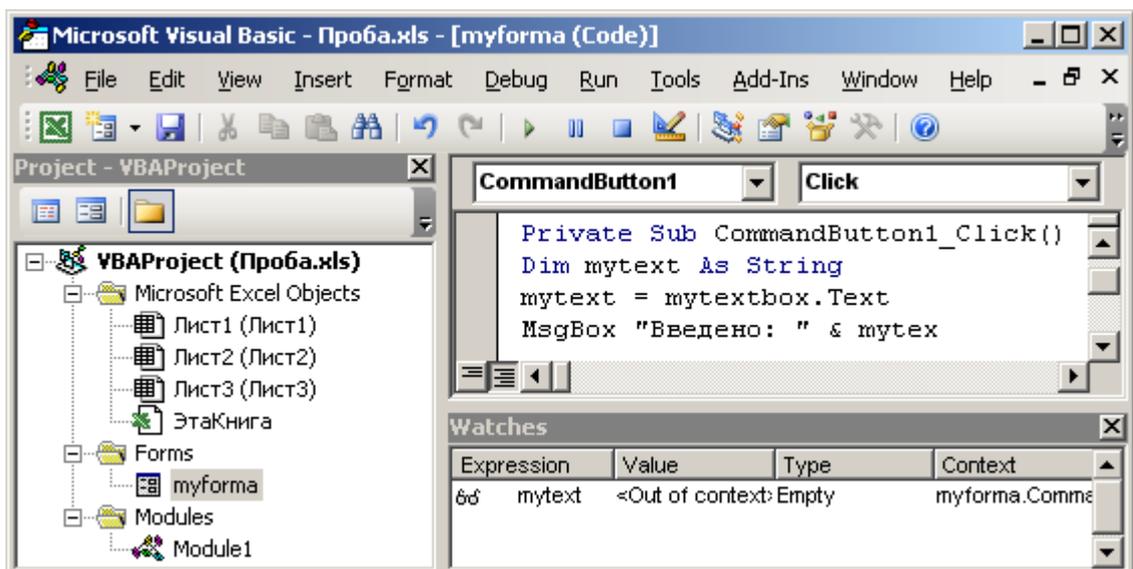


Для выполнения следующей операции нужно нажать *F8*. Выполняя программу по шагам, можно установить с помощью окна контрольного значения при выполнении какой строки кода переменной **mytext** будет присвоено значение.



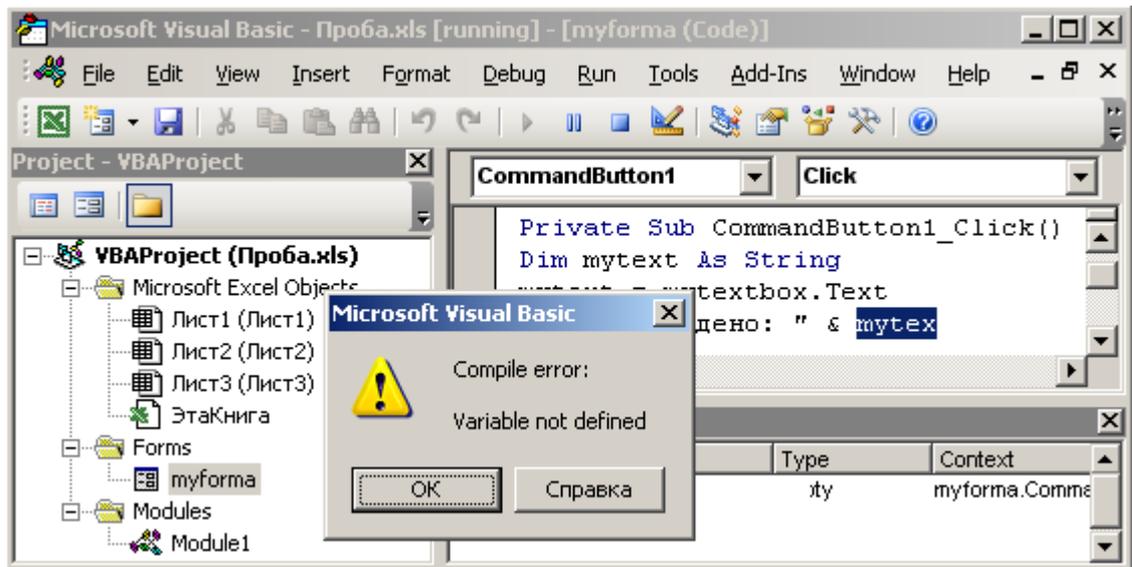
Продолжать выполнять программу по шагам (*F8*) до ее завершения (прекратятся выделяться строки в тексте программного кода).

25. После завершения выполнения программы в ее тексте заменить **mytext** на **mytex** в строке **MsgBox "Введено: " & mytext**.

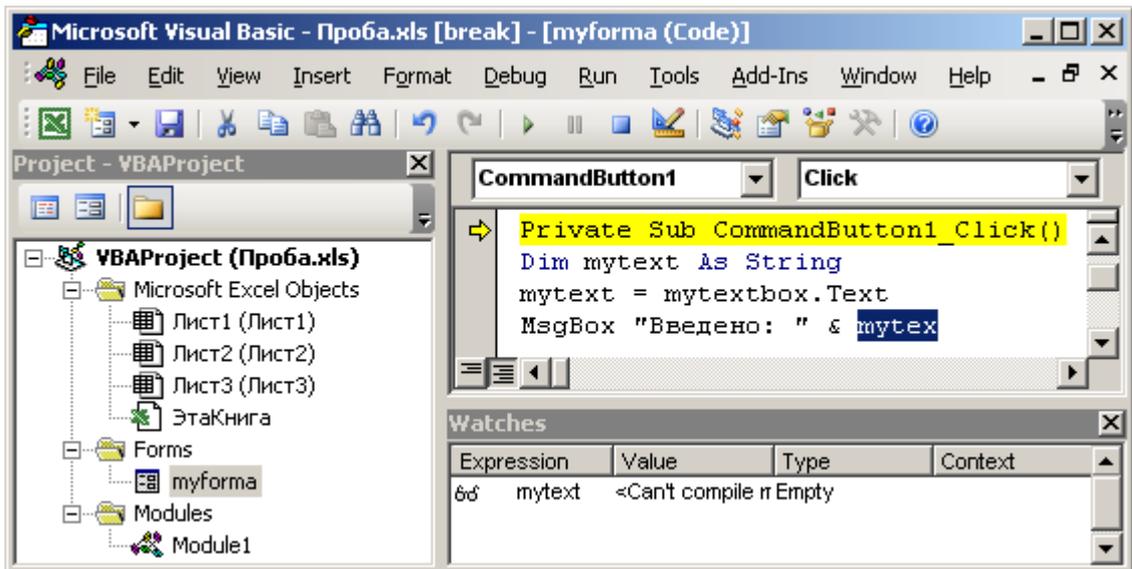


Запустить программу на выполнение (*F5*).

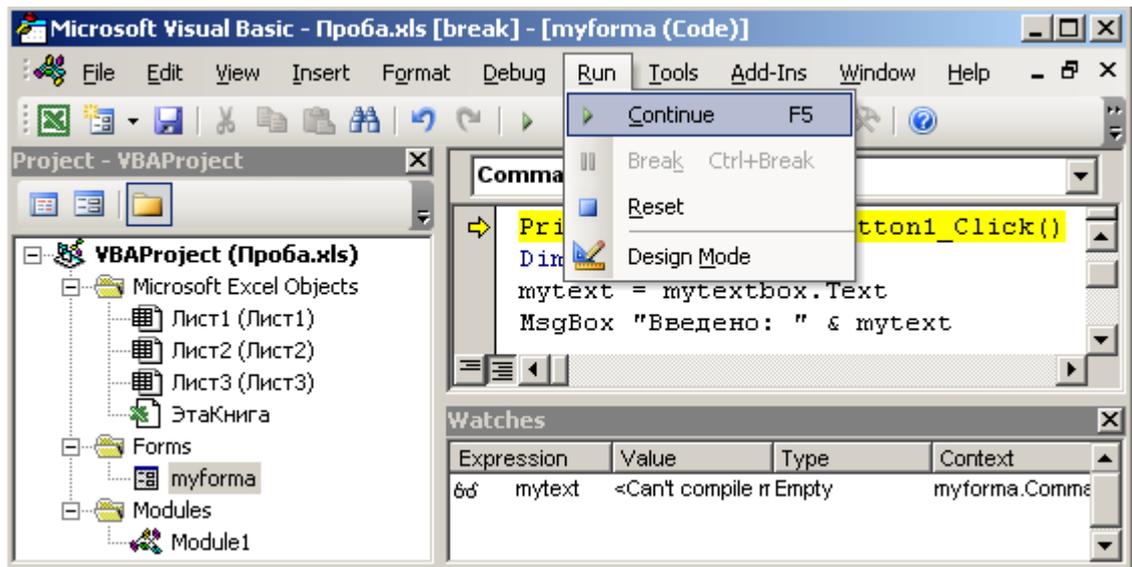
26. Ввести текст в поле ввода и нажать кнопку **Завершение**. Прочитать сообщение об ошибке и нажать в его окне **OK**.



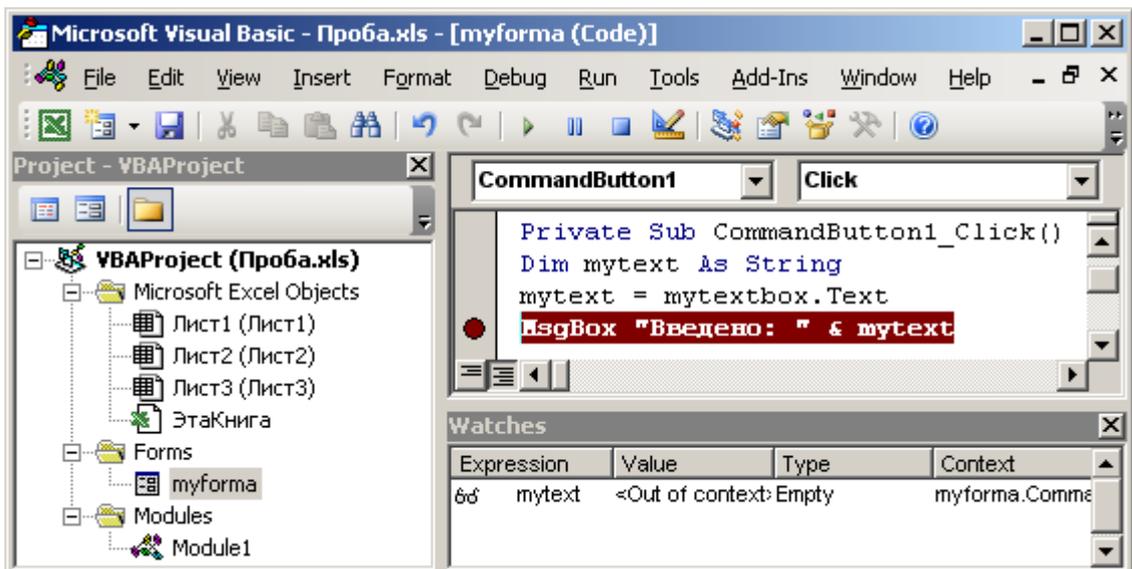
В открывшемся окне VB желтым цветом будет отмечена процедура, содержащая ошибку, а синим – место ошибки.



Ошибку можно устранить (заменить **mytex** на **mytext**), не завершая текущий запуск программы (исправить ошибку и нажать кнопку **F5** (*Продолжить (Continue)*)) или завершив его (*Запуск (Run)* > *Сброс (Reset)*) для дальнейшего редактирования.

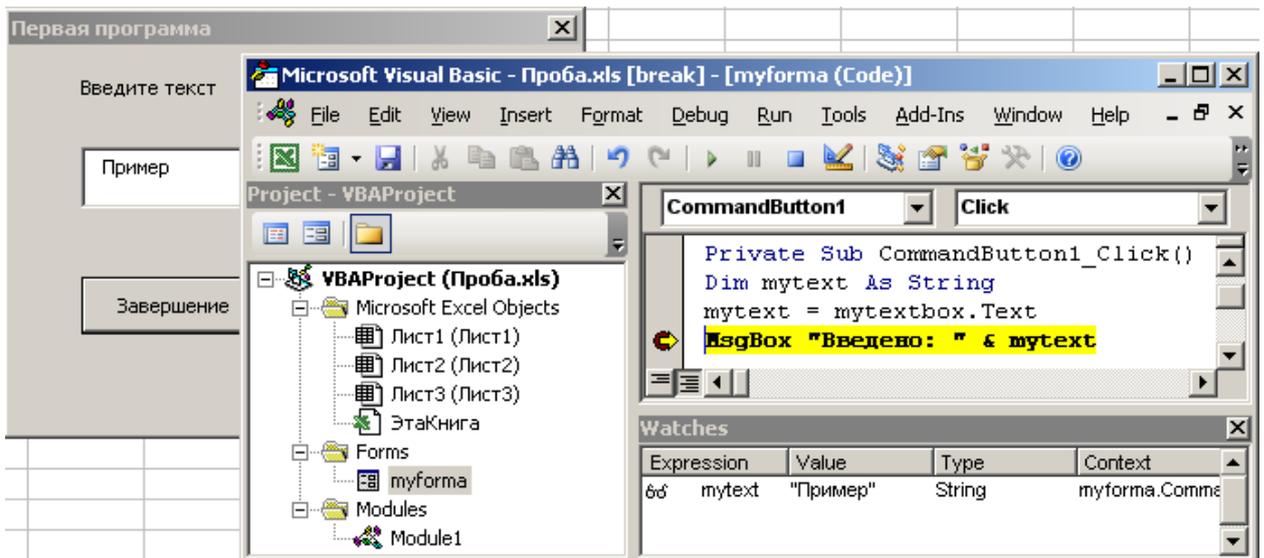


27. После завершения выполнения программы и устранения ошибки в процедуре **CommandButton1_Click()** установить курсор перед ключевым словом **MsgBox** и выполнить команду *Отладка (Debug) > Точка останова (Toggle Breakpoint)* или щелкнуть на поле слева от строки.



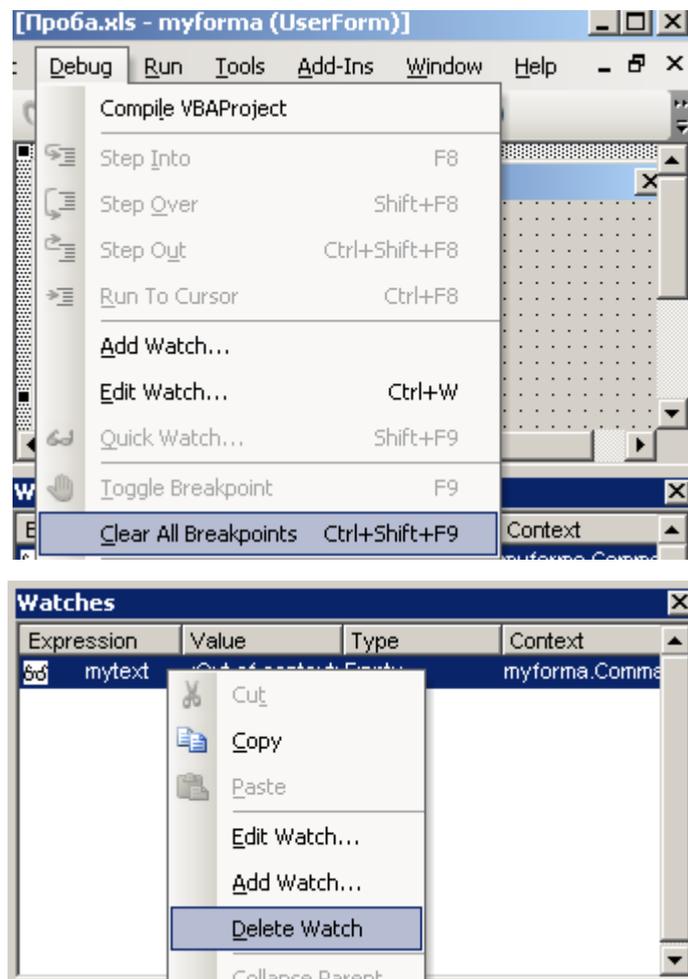
Запустить программу на выполнение (**F5**).

28. Ввести текст в поле ввода и нажать кнопку **Завершение**. Процедура будет приостановлена перед строкой отмеченной желтым цветом (точка останова). Убедиться в наличии значения в переменной **mytext** (окно контрольного значения).



Завершить выполнение программы (**F5** или **F8 (по шагам)**).

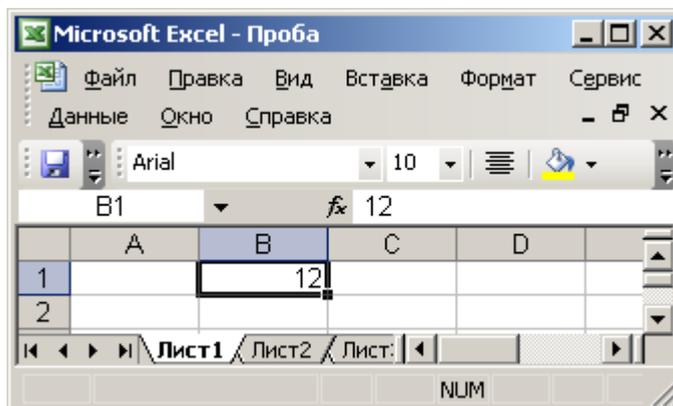
29. Очистить все точки останова (щелчок слева от строки с остановом или **Отладка (Debug) > Снять все точки останова (Clear All BreakPoints)**) и контрольные значения (в окне контрольного значения команда контекстного меню **Удалить контрольное значение (Delete Watch)**).



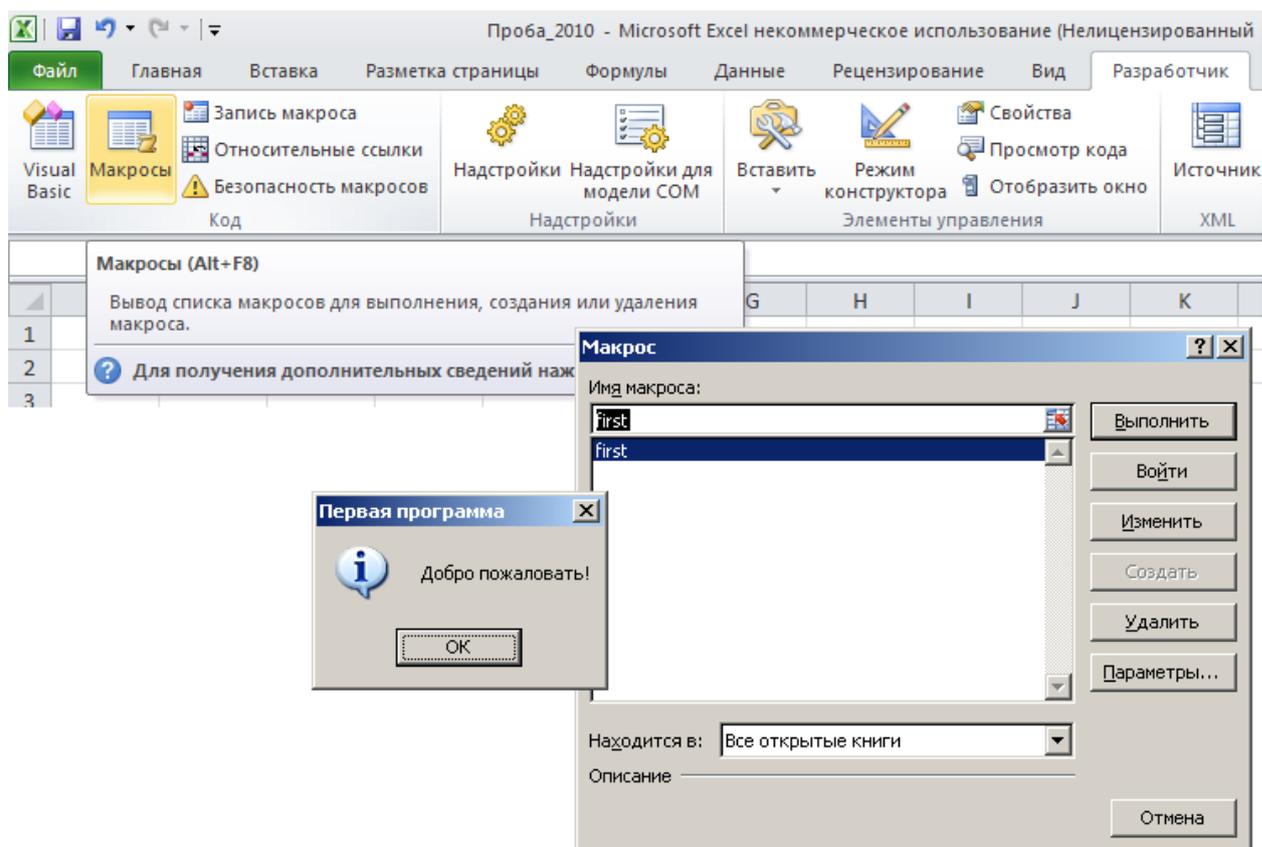
Сохранить файл.

30. Закрывать проект и вернуться в MS Excel (**Файл (File) > Закрывать и вернуться в MS Excel (Close and Return to Microsoft Excel)**).

31. На первом листе книги MS Excel в ячейке **B1** ввести любое число.

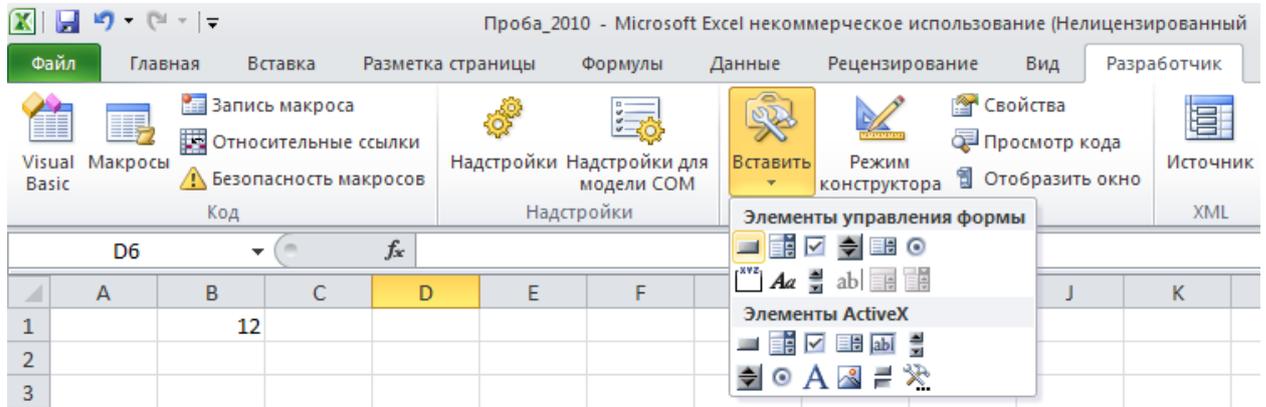


32. Запустить процедуру **first** командой оболочки MS Excel: **Разработчик > Код > Макросы > first (в списке Имя макроса) > Выполнить**.

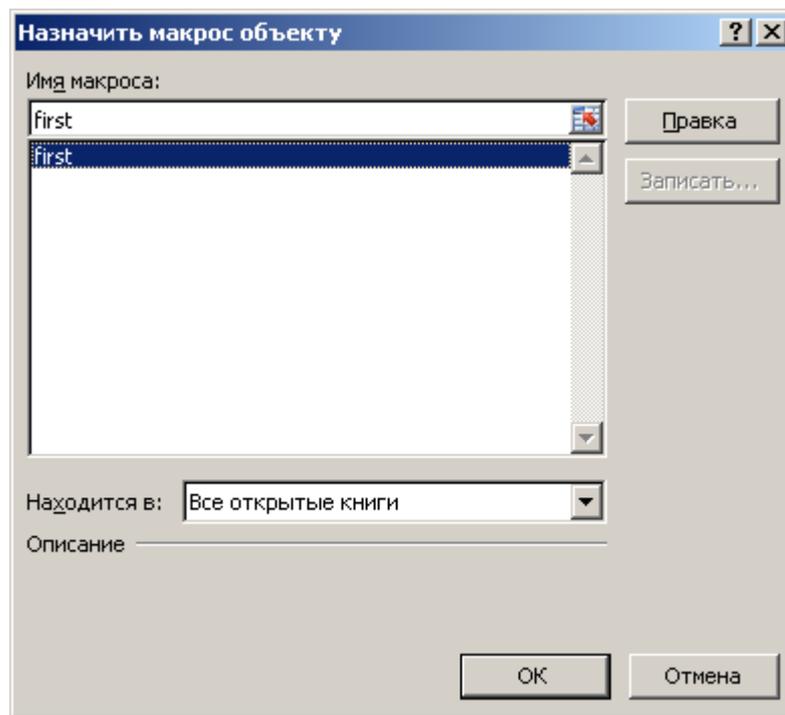


Обратить внимание на появлении в окне результатов работы программы значения из ячейки таблицы MS Excel.

33. В оболочке MS Excel выполнить команду: *Разработчик > Элементы управления > Вставить > Кнопка*. Указать позицию элемента управления типа "кнопка" на листе MS Excel.

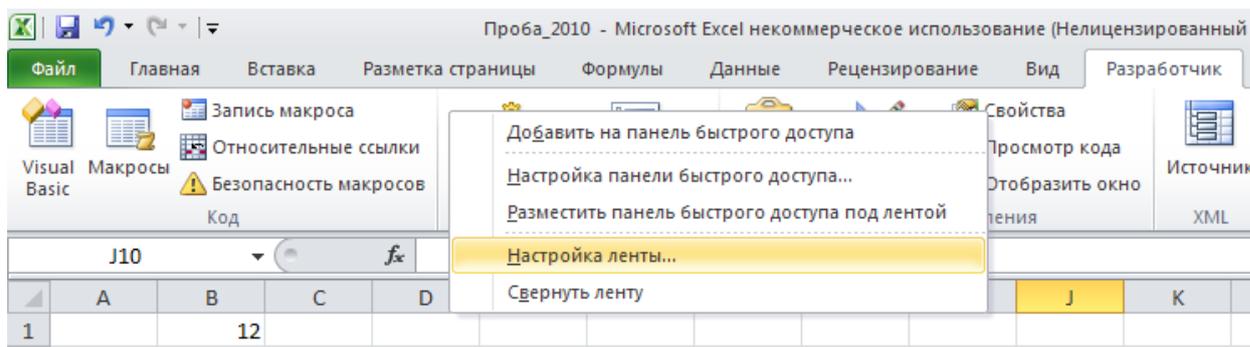


34. В открывшемся окне (*Назначить макрос объекту*) выбрать процедуру **first** и нажать **OK**.

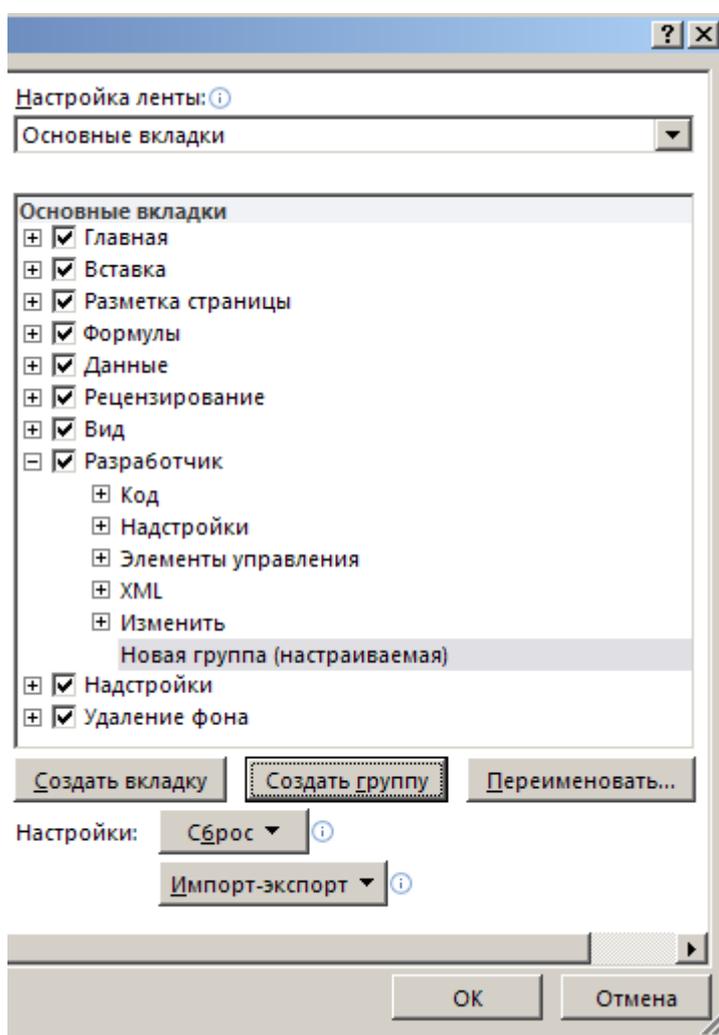


35. Нажать на добавленную кнопку на листе MS Excel. Ввести текст в окно созданной программы и завершите ее.

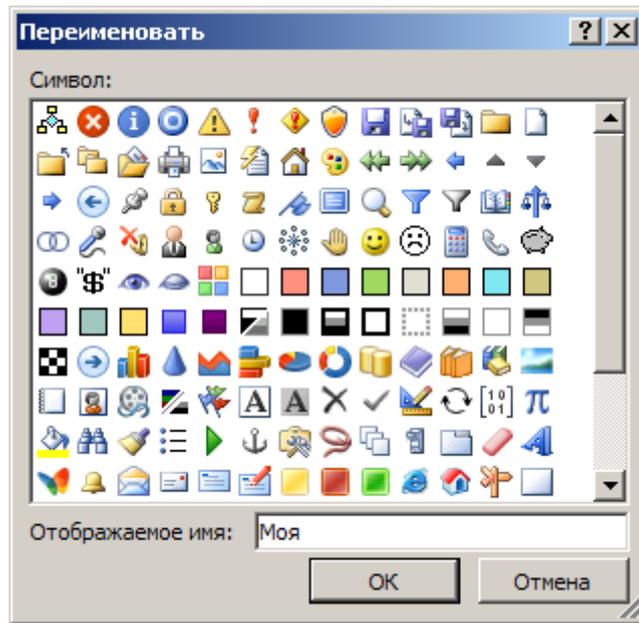
36. Выполнить контекстную команду ленты *Настройка ленты....*



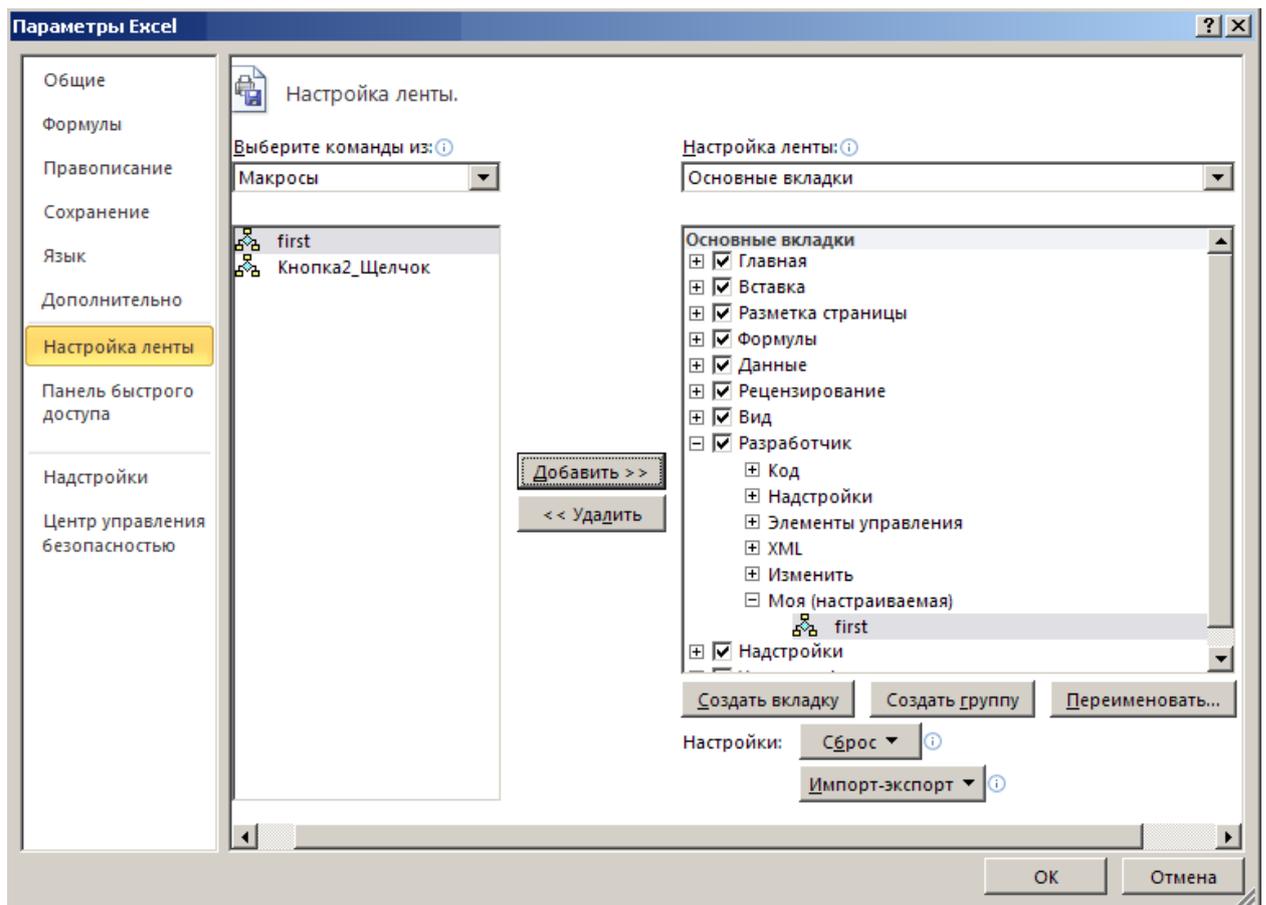
37. В окне *Параметры Excel > Настройка ленты* для основной вкладки *Разработчик* выполнить команду *Создать группу*.



Переименовать созданную группу, выделив ее и нажав кнопку *Переименовать* (новое имя – **Моя**).

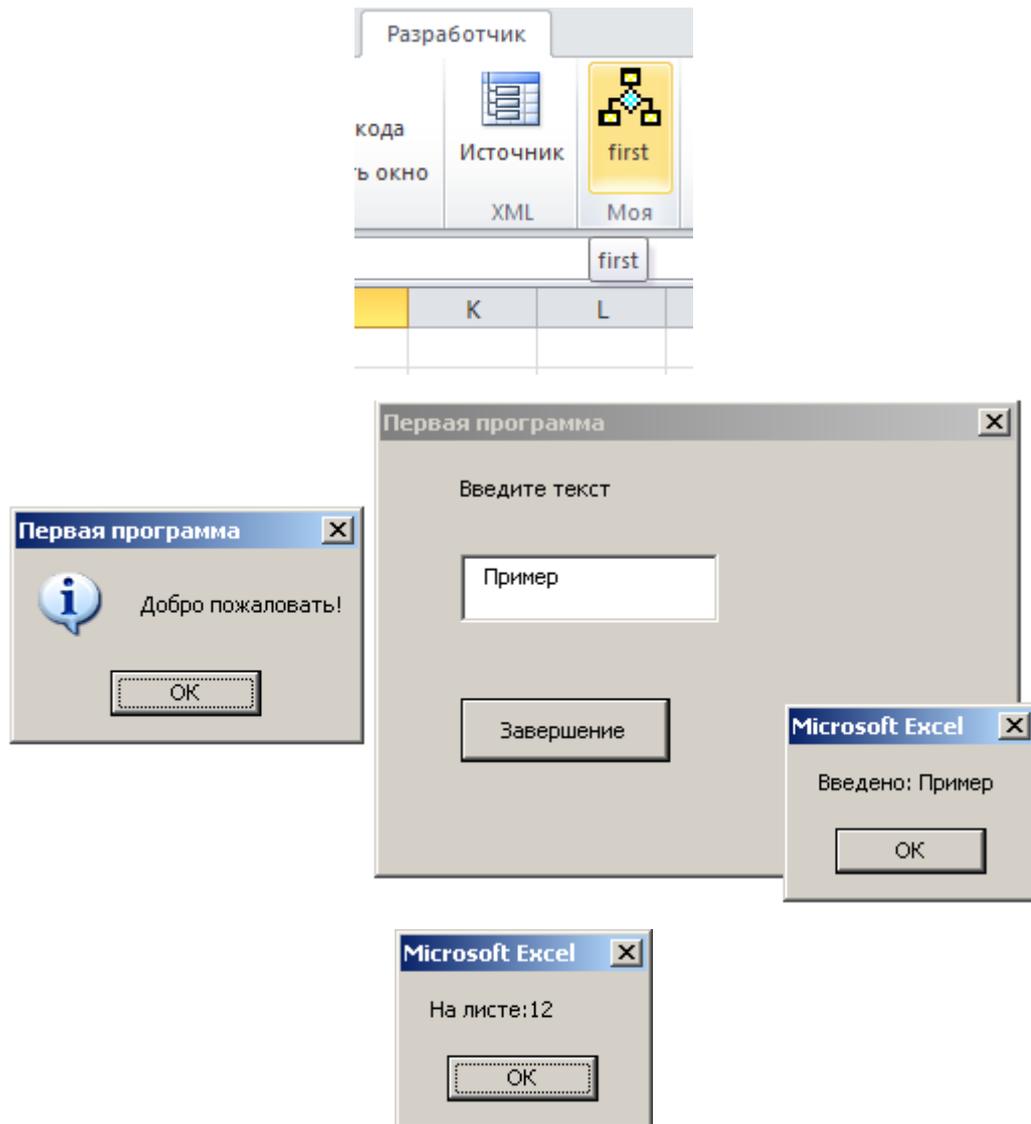


38. Выделить созданную группу (**Моя**) в списке *Основные вкладки*. Затем в списке *Выберите команды из* установить вариант *Макросы*, выбрать процедуру-макрос **first** и нажать кнопку *Добавить....*



Закрывать окно *Параметры Excel*, нажав *OK*.

39. Нажать кнопку **first** в созданной группе команд *Разработчик > Моя*.



Состав отчета

1. Номер, название и цель работы.
2. Ответы на контрольные вопросы:
 - а) Каким образом открыть редактор VB в MS Excel?
 - б) Каким образом добавить программный модуль и сохранить проект VB MS Excel?
 - в) Как получить информацию о синтаксисе текущей процедуры или функции в VB?
 - г) Как вызывается и для чего используется Окно свойств VB (пример)?
 - д) Как добавить форму в проект VB? Как добавить элемент управления в форму?

- е) Как вызвать окно программного кода для элемента управления?
- ж) Как добавить контрольное значение в VB?
- з) Как запустить программу с остановкой на каждом шаге?
- и) Как добавить точки останова в программу VB?
- к) Перечислите и поясните способы запуска программ VB в MS Excel.

ЛАБОРАТОРНАЯ РАБОТА № 2. ПЕРЕМЕННЫЕ. ОПЕРАТОРЫ. ВСТРОЕННЫЕ ФУНКЦИИ

Цель работы

Изучить типы данных VB и научиться использовать их в переменных и массивах. Получить навыки использования операторов и встроенных функций VB.

Теоретические сведения

Переменные и константы

Константы и переменные предназначены соответственно для хранения фиксированных и изменяемых значений в вычислениях.

Все переменные VB имеют определенный тип (табл. 1). Для использования переменных в программе VB их необходимо объявить, используя следующий синтаксис:

Dim <переменная> As <тип>

или

Dim <переменная 1> , ..., <переменная ...> As <тип>

или

Dim <переменная 1> As <тип 1>, <переменная ...> As <тип ...>

здесь

переменная ... – задаваемый разработчиком идентификатор,

тип ... – один из идентификаторов для типов данных: числовых (**Integer**, **Long**, **Single**, **Double**), текстовых (**String**), логических (**Boolean**) и др.,

<...> – позиция размещения обязательного элемента синтаксиса

[...] – позиция размещения необязательного элемента синтаксиса)

Примеры объявления переменных:

Dim peremen As Integer

' Объявление переменной по имени peremen для хранения

' целых чисел

Dim перемен1, перемен2 As Integer

' Объявление переменных перемен1 и перемен2

' для хранения целых чисел

Dim перемен3 As String, перемен4 As Double

' Объявление переменной перемен3 для хранения строк

' и переменной перемен4 для хранения вещественных чисел

Таблица 1

Типы данных в VB

Тип данных	Размер памяти	Диапазон хранимых значений
Byte	1 байт	От 0 до 255
Boolean	2 байта	True или False
Integer	2 байта	От -32768 до 32767
Long (длинное целое)	4 байта	От -2147483648 до 2147483647
Single (с плавающей точкой)	4 байта	От -3,402823E38 до -1,401298E-45 для отрицательных значений; От 1,401298E-45 до 3,402823E38 для положительных значений
Double (плавающее двойной точности)	8 байт	От -1,79769313486232E308 до -4,94065645841247E-324 для отрицательных значений; от 4,94065645841247E-324 до 1,79769313486232E308 для положительных значений
Currency (т.н. денежный формат)	8 байт	От -9223372036854775808 до 9223372036854775807
Decimal	14 байт	+/-79228162514264337593543950335 без дробной части;

		+/-7,9228162514264337593543950335 с 28 дробными разрядами; наименьшее ненулевое значение +/- 0,00000000000000000000000000000001
Date (Дата)	8 байт	От 1 января 100 г. до 31 декабря 99996 г.
Object (Объект)	4 байт	Ссылка на любой объект
String (строка переменной длины)	10 байт + длина строки	Расширяемая от 0 до 2 миллиардов символов
String (фиксированной длины)	Длина строки	Расширяемая от 0 до 65400 символов
Variant	16 байт	Любое численное или текстовое значение

Использование для переменных определенных типов данных определяется назначением переменной: для целочисленных вычислений применяются типы **Integer** и **Long**, для вычислений с дробной точкой – **Single**, **Double**, для операций со строками – **String**, для логических операций – **Boolean**, для операций с датой – **Date** и т.д.

После объявления переменным можно присваивать значения соответствующего типа, затем производить действия над переменными, считывать их значения:

Sub primer()

Dim i As Integer

' Объявление переменной i для хранения целых чисел

i=3

' Запись в переменную i значения 3

i=i+2

' Добавление к значению переменной *i* числа (2)

' и запись его в переменную *i*

MsgBox *i*

' Отображение значения из переменной *i* в окне сообщения

End Sub



Например, для переменных типа **Data** возможны следующие способы (синтаксисы) присвоения:

<Переменная>=#<месяц(№)>/<число>/<год>

<часы>:<минуты>:<секунды>#

<Переменная>=# <часы>:<минуты>#

<Переменная>=#<месяц(№)>/<число>/<год>#

<Переменная>=Now (запись в переменную текущего времени и даты)

и считывания:

<Переменная 1>=Hour(<Переменная>) – считывание количества часов,

<Переменная 1>=Minute(<Переменная>) – считывание количества минут,

<Переменная 1>=Day(<Переменная>) – считывание числа дней,

<Переменная 1>=Year(<Переменная>) – считывание года,

<Переменная 1>=Month(<Переменная>) – считывание месяца

и изменения значений:

<Переменная>=<Переменная> +(-) <значение> – увеличение или уменьшение значения дней,

<Переменная>=<Переменная> +(-) <значение>/24 – увеличение или уменьшение значения часов,

<Переменная>=<Переменная> +(-) <значение>/1440 – увеличение или уменьшение значения минут,

<Переменная>=<Переменная> +(-) <значение>/86400 – увеличение или уменьшение значения секунд.

Пример работы с переменной (типа **Data**):

Sub primer()

Dim mydata As Date

' Объявление переменной mydata для хранения дат

Dim mynum As Integer

' Объявление переменной mynum для хранения целых чисел

mydata = #1/4/2012 11:57:00 AM#

' Запись в переменную mydata даты: 4 января 2012 года, 11 часов, 57 минут, 0 секунд

mynum = Month(mydata)

' Считывание номера месяца из переменной mydata и запись его в переменную mynum

MsgBox mynum

' Отображение окна сообщения со значением переменной mynum

mydata = mydata + 30

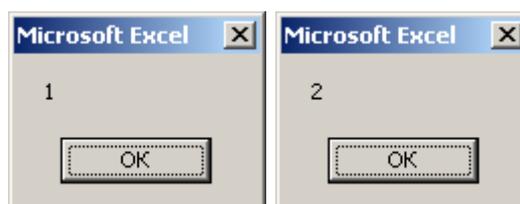
' Добавление к значению даты в переменной mydata 30 дней
mynum = Month(mydata)

' Считывание номера месяца из нового значения переменной mydata и запись его в переменную mynum

MsgBox mynum

' Отображение значения переменной mynum

End Sub



В VB переменным можно присваивать значения, взятые из ячеек MS Excel.

Dim A As Double

```
' Объявление переменной A для хранения вещественных чисел
A=Application.Workbooks("around_book_Exvba.xls")._
Worksheets("Лист1").Cells(2,1)
```

```
' Запись в переменную A значения (числа) из ячейки A2 листа
' "Лист1" электронной таблицы "around_book_Exvba.xls"
```

При отсутствии объявления переменной присваивается тип данных **Variant**, который может соответствовать любому типу данных. Тип **Variant** значительно увеличивает размер файла программы и не позволяет отслеживать ошибки, связанные с некорректным использованием типов данных в программе (н.п., присвоение текста числу).

Проверка типа данных во время выполнения программы производится при помощи таких функций как **TypeName(<переменная>)**, **IsNumeric(<переменная>)** и др.

TypeName(<переменная>) – возвращает тип переменной заданной аргументом в виде ключевого слова: **"Integer"**, **"String"**, **"Double"** и т.п.:

Sub primer()**Dim x As Variant**

```
' Объявление (создание) переменной x типа Variant
```

```
' для хранения значений любого типа
```

```
x = 4
```

```
' Запись в переменную x целого числа (Integer)
```

```
MsgBox TypeName(x)
```

```
' Отображение окна-сообщения со значением типа значения,
```

```
' хранящегося в переменной x
```

```
x = 4.3
```

```
' Запись в переменную x вещественного числа (Double)
```

```
MsgBox TypeName(x)
```

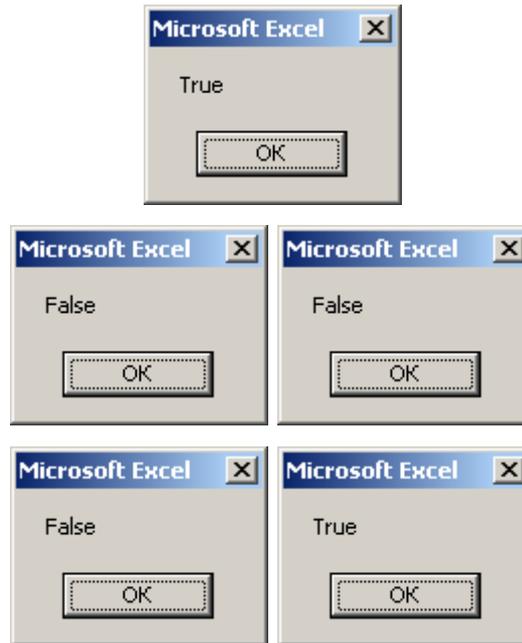
```
x = "слово"
```

```
' Запись в переменную x строки (String)
```

MsgBox TypeName(x)**End Sub**

IsNumeric(<переменная>), IsDate(<переменная>), ... – группа функций, определяющих принадлежность переменной к одному из типов и возвращающих соответственно **True (Истину)** или **False(Ложь)** в зависимости от результата проверки. Функция **IsEmpty(<переменная>)** проверяет, храниться ли в ее параметре-переменной значение.

Sub primer()**Dim x As Variant**' **Объявление (создание) переменной x**' **для хранения любых значений****MsgBox IsEmpty(x)**' **Вывод сообщения об отсутствии значения в переменной x****x = "слово"**' **Запись в переменную x текста****MsgBox IsEmpty(x)**' **Вывод сообщения о наличии значения в переменной x****MsgBox IsNumeric(x)**' **Вывод сообщения об отсутствии в переменной x числа****x = 3.5**' **Запись в переменную x числа****MsgBox IsEmpty(x)**' **Вывод сообщения о наличии значения в переменной x****MsgBox IsNumeric(x)**' **Вывод сообщения о наличии в переменной x числа****End Sub**



Для преобразования типов данных служат специальные функции VB (табл. 2): **CBool**, **CCur**, **CDate**, **Cdbl**, **CInt**, **CSng**, **CStr** и другие, имеющие синтаксис:

значение возвращаемого типа = функция (значение исходного типа)

Таблица 2

Функции преобразования типов данных

Функция	Исходный тип	Возвращаемый тип
CBool	String	Ошибка
	Integer, Long, Single, Double, Date, Currency	Boolean (0 – False)
CCur	String	Currency (для случая наличия в строке числа с указанием единицы измерения (н.п.: "12 р."), в остальных случаях – ошибка)
	Integer, Date	Currency
	Long, Single, Double	Currency (в пределах типа Currency, в обратном случае – ошибка)

	Boolean	Currency (0 для False, – 1 для True)
CDate	String	Date (для записи строки в виде даты (н.п.: "14 декабря 2005"), в обратном случае – ошибка)
	Integer, Long, Single, Double, Currency	Date (в пределах диапазона значений типа Date)
	Boolean	Date (00:00:00 для False, 29.12.1899 для True)
Cdbl	String	Ошибка
	Integer, Long, Single, Date, Currency	Double
	Boolean	Double (0 для False, –1 для True)
CInt	String	Ошибка
	Double, Long, Single, Date, Currency	Integer (в пределах диапазона значений типа Integer)
	Boolean	Integer (0 для False, –1 для True)
CLng	String	Ошибка
	Double, Single, Currency	Long (в пределах диапазона значений типа Long)
	Integer, Date	Long
	Boolean	Long (0 для False, –1 для True)
CSng	String	Ошибка
	Double	Single (в пределах диапазона значений типа Single)
	Integer, Long, Date, Currency	Single
	Boolean	Single (0 для False, –1 для True)
CStr	Single, Double, Integer, Long, Date, Currency	String
	Boolean	String ("False" для 0, "True" для 1 (–1))

Пример использования функции преобразования типа:

Sub primer()

Dim x As Double

' Объявление переменной x для хранения вещественных чисел

x=4.51

' Запись в переменную x вещественного числа

x=CInt(x)

' Преобразование значения числа в переменной x

' к целому типу

MsgBox x

' Отображение преобразованного значения из переменной x

End Sub



Встроенные операторы и функции

В табл. 3–6 представлены встроенные операторы и функции VB.

Таблица 3

Математические операторы

Оператор	Назначение	Синтаксис. Использование
+	Сложение	<число>+<число>...+<число>
-	Вычитание	<число>-<число>...-<число>. Из результата первого вычитания вычитается третье число и так далее
*	Умножение	<число>*<число>...*<число>
/	Деление	<число>/<число>.../<число>. Результат деления первого числа на второе делится на третье и так

		далее
\	Целочис- ленное деление	<число>\<число>...<число>. Результат – целая часть от деления. Делимое и делитель преобразуются к целому типу!
mod	Остаток от деления нацело	<число>mod<число>. Результат – остаток деления первого числа на второе. Делимое и делитель преобразуются к целому типу!
^	Степень	<число1>^<число>. Отрицательные значения <число1> допускаются только для целых значений <число>

Пример использования математических операторов:

Найти значение выражения

$$x = \text{целая часть} \left(\frac{7}{3} \right) * (3 + 3,5^{-2,5})$$

Sub primer()

Dim x As Double

' **Объявление переменной x для хранения вещественных чисел**

x=7\3

' **Запись в переменную x целой части от деления двух чисел**

Msgbox x

' **Отображение значения из переменной x**

x=x*(3+3.5^(-2.5))

' **Нахождение произведения значения переменной x на сумму**

' **числа и степенного выражения;**

' **запись результата в переменную x**

Msgbox x

' **Отображение значения из переменной x**

End Sub

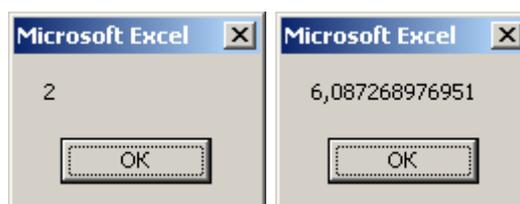


Таблица 4

Математические функции

Функция	Назначение	Синтаксис. Использование
Abs	Модуль числа	Abs(<число>). Если аргумент – не число, возвращает Null
Atn	Арктангенс числа	Atn (<число>). Возвращает угол в радианах от $-\pi/2$ до $+\pi/2$. $\text{Arccos}(\langle \text{число} \rangle) = \text{Atn}(-\langle \text{число} \rangle * \text{Sqr}(-\langle \text{число} \rangle * \langle \text{число} \rangle + 1)) + 2 * \text{Atn}(1)$
Cos	Косинус угла	Cos(<число>). Возвращает результат в диапазоне от -1 до $+1$. Аргумент (<число>) – радианы
Exp	е ^{степень}	Exp(<число>). Возвращает результат возведения числа e в степень, заданную числом
Fix, Int	Целая часть числа	Fix(<число>); Int(<число>). Fix возвращает целое отрицательное число ближайшее большее или равное указанному аргументом отрицательному числу, а Int – ближайшее меньшее или равное
Log	Натуральный логарифм числа	Log(<число>) Для логарифма по произвольному основанию: $\text{Log}_x y = \text{Log}(y) / \text{Log}(x)$
Rnd	Случайное число	Возвращает случайное число в диапазоне [0,1)
Sgn	Знак числа	Sgn(<число>). Возвращает: 1 – если аргумент >0 ; 0 – если аргумент = 0; -1 – если аргумент <0
Sin	Синус угла	Sin (<число>)
Sqr	Квадратный корень	Sqr(<число>). Аргумент (<число>) – радианы

Tan	Тангенс угла	Tan(<число>). Аргумент (<число>) – радианы
-----	--------------	--------------------------------------------

Пример использования математических функций:

Найти значение выражения

$$x = \cos(\text{случайное целое число}[0^\circ \dots 90^\circ])$$

Sub primer8()

Dim x As Double

' Объявление переменной x для хранения вещественных чисел

x = Int(Rnd() * 91 + 0)

' Задание случайного целого числа из диапазона [0...90]

' и запись его в переменную x

MsgBox x

' Отображение значения из переменной x

x = Cos(x * Excel.WorksheetFunction.Pi / 180)

' Перевод значения переменной x в радианы с помощью

' встроенной в лист MS Excel функции Pi, вычисление cos

' и запись его значения в переменную x

MsgBox x

' Отображение значения из переменной x

End Sub

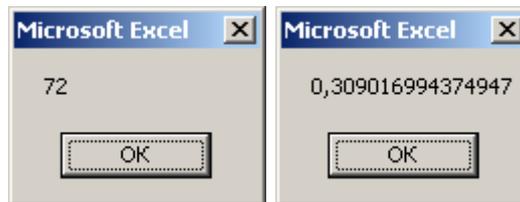


Таблица 5

Логические операторы

Оператор	Результат
And (И)	True And True – возвращает True True And False – возвращает False False And False – возвращает False

Or (ИЛИ)	True Or True – возвращает True True Or False – возвращает True False Or False – возвращает False
Not (НЕ)	Not True – возвращает False Not False – возвращает True
Xor (Исключающее ИЛИ)	True Xor True – возвращает False True Xor False – возвращает True False Xor False – возвращает True
<, >, >=, <=, <>	<число>оператор<число>. Возвращают истину (True), если первое число меньше, больше, больше или равно, меньше или равно, не равно второму соответственно

Пример использования логических операторов:

Определить лежит ли значение x в промежутке [-2...1).

Sub primer()

Dim x As Double

' Объявление переменной x для хранения вещественных чисел

Dim bv As Boolean

' Объявление переменной bv для хранения логических значений

x = -2.1

' Запись в переменную x числа

bv = (x < 1) And (x >= -2)

' Вычисление логического выражения и запись его

' в переменную bv

MsgBox bv

' Отображение значения переменной bv

x = -1.2

bv = (x < 1) And (x >= -2)

MsgBox bv

x = 1

bv = (x < 1) And (x >= -2)

MsgBox bv

End Sub

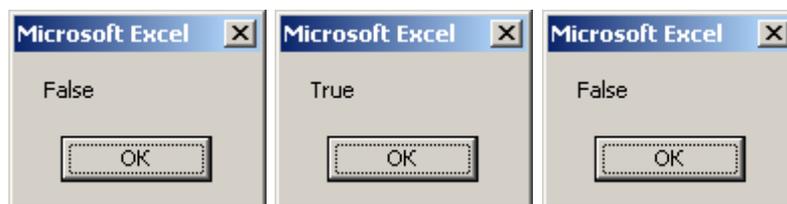


Таблица 6

Текстовые операторы и функции

Функция/ оператор	Назначение	Синтаксис. Использование
& оператор	Слияние (соединение) строк	<строка>&<строка>...&<строка>
UCase	Изменение регистра букв строки на верхний	UCase(<строка>)
LCase	Изменение регистра букв строки на нижний	LCase(<строка>)
InStr	Поиск одной строки в другой с начала строки	InStr([<начало>,]<строка1>, <строка2>[, <тип сравнения>]). Функция возвращает позицию первого вхождения строки2 в строку 1 (при отсутствии вхождения возвращается 0); <начало> определяет начальную позицию поиска в строке1; <тип сравнения> = 1 указывает на посимвольное сравнение
Left	Возврат определенного числа символов с начала строки	Left(<строка>, <количество символов>)

Right	Возврат определенного числа символов с конца строки	Right(<строка>, <количество символов>)
Mid	Возврат определенного числа символов с определенного места в строке	Mid(<строка>, <начало>[, <длина>]) или для замены части строки другой строкой: Mid(<строка1>, <начало>[, <длина>]) = <строка2>. Число замененных символов не должно приводить к превышению длины исходной строки.
LTrim	Удаление пробелов в начале строки	LTrim(<строка>)
RTrim	Удаление пробелов в конце строки	RTrim(<строка>)
Trim	Удаление пробелов с обеих сторон строки	Trim(<строка>)
Len	Определение длины строки	Len(<строка>)
Chr	Преобразование ASCII-кода в символ	Chr(<кодСимвола>). Коды 0–31 соответствуют управляющим символам ASCII. Например, Chr(10) возвращает символ перевода строки, а Chr(13) – возврат каретки
Asc	Преобразование символа в ASCII-код	Asc(<символ>). Возвращаемые значения лежат в диапазоне 0–255 для однобайтовых символьных наборов
StrConv	Изменение регистра в строке символов	StrConv(<строка>, <условие>). Параметр <условие> задает тип обращения: 1 –

		преобразование всех символов к верхнему регистру, 2 – преобразование всех символов к нижнему регистру, 3 – преобразование к верхнему регистру первых символов строк и т.д.
--	--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Пример использования текстовых операторов и функций:

Вернуть из строки предпоследний символ в верхнем регистре.

Sub primer()

Dim s As String

' Объявление переменной s для хранения строк

s = "Пример строки"

' Запись в переменную s строки

s = Mid(s, Len(s) - 1, 1)

' Определение номера позиции предпоследнего символа в строке

' Len(s)-1, извлечение одного этого символа из строки Mid(...),

' запись извлеченного символа в переменную s

s = UCase(s)

' Преобразование к верхнему регистру символа в переменной s

' и перезапись его в переменную

MsgBox s

' Отображение значения переменной s

End Sub



Функция MsgBox

MsgBox выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки, а затем возвращает значение типа **Integer**, указывающее, какая кнопка была нажата.

MsgBox(<Сообщение>[, <Кнопки>] [, <Заголовок>] [, Файл помощи, Раздел помощи])

Синтаксис функции **MsgBox** содержит именованные аргументы (табл. 7).

Таблица 7

Аргументы функции MsgBox

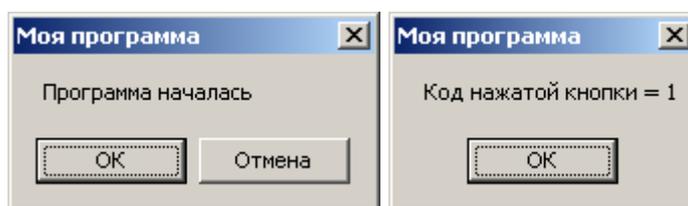
Аргумент	Описание
Сообщение	Строковое выражение, отображаемое как сообщение в диалоговом окне
Кнопки	Числовое выражение (код), которое указывает число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию этого аргумента равняется 0. Вместо чисел можно использовать ключевые слова: н.п., vbOKOnly (или код 0) – окно будет содержать только кнопку ОК, vbOKCancel (или код 1) – окно будет содержать кнопки ОК и Cancel.
Заголовок	Строковое выражение, отображаемое в строке заголовка диалогового окна. Если аргумент опущен, в заголовок помещается имя приложения
Файл помощи	Строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне
Раздел помощи	Числовое выражение, определяющее номер соответствующего раздела справочной системы

Пример использования функции **MsgBox**:

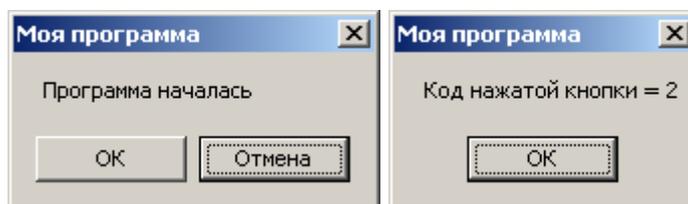
Sub primer()

Dim i As Integer

```
' Объявление переменной i для хранения кода нажатой кнопки
i=MsgBox ("Программа началась",vbOKCancel,"Моя программа")
' Отображение окна-сообщения с текстом "Программа началась",
' кнопками OK и Cancel, заголовком "Моя программа".
' Считывание кода нажатой пользователем кнопки (OK или
' Cancel) в переменную i
MsgBox "Код нажатой кнопки = " & i,,"Моя программа"
' Отображение окна-сообщения с кодом нажатой кнопки
' из переменной i и заголовком "Моя программа"
End Sub
```



ИЛИ



Функция InputBox

InputBox выводит на экран диалоговое окно, содержащее сообщение и поле ввода, устанавливает режим ожидания ввода текста или нажатия кнопки, а затем возвращает значение типа **String**, содержащее текст, введенный в поле.

Синтаксис функции **InputBox**:

```
InputBox(<Пояснение>[, <Заголовок>] [, <Текст>] [, <Горизонталь>]
[, <Вертикаль>] [, Файл помощи, Раздел помощи])
```

содержит именованные аргументы (табл. 8).

Таблица 8

Аргументы функции **InputBox**

Аргумент	Описание
Пояснение	Строка, отображаемая как сообщение в диалоговом окне
Заголовок	Строка, отображаемая в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения
Текст	Строка, отображаемая в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым
Горизонталь	Число, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана (в твипах; 1 дюйм = 1440 твипов). Если этот аргумент опущен, диалоговое окно выравнивается по центру экрана по горизонтали
Вертикаль	Число, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана (в твипах). Если этот аргумент опущен, диалоговое окно помещается по вертикали примерно на одну треть высоты экрана

Если пользователь нажимает кнопку **ОК** или клавишу **ENTER**, функция **InputBox** возвращает содержимое поля ввода. Если пользователь нажимает кнопку **Отмена**, функция возвратит пустую строку ("").

Пример использования функции **InputBox**:

Sub primer()

Dim perem As String

' Объявление переменной **perem** для хранения строк

perem=InputBox("Как Вас зовут?", "Мой диалог")

' Отображение диалогового окна с полем ввода, текстом

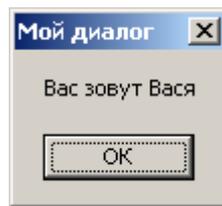
' приглашения "Как Вас зовут?" и заголовком "Мой диалог".

' Запись в переменную **perem** текста введенного пользователем

' в поле после нажатия кнопки **ОК**

MsgBox "Вас зовут " & perem,,"Мой диалог"

' Отображение окна-сообщения с введенным текстом
 ' из переменной `reget` и заголовком "Мой диалог"
 End sub



Пример

Написать программу для выполнения следующей последовательности действий.

1) ввод ...

... с использованием функции **Inputbox** текстовой строки (переменная *S*),

... с использованием функции **Inputbox** двух чисел (переменные *A* и *B*),

... непосредственно в тексте программы двух чисел (переменные *C* и *D*);

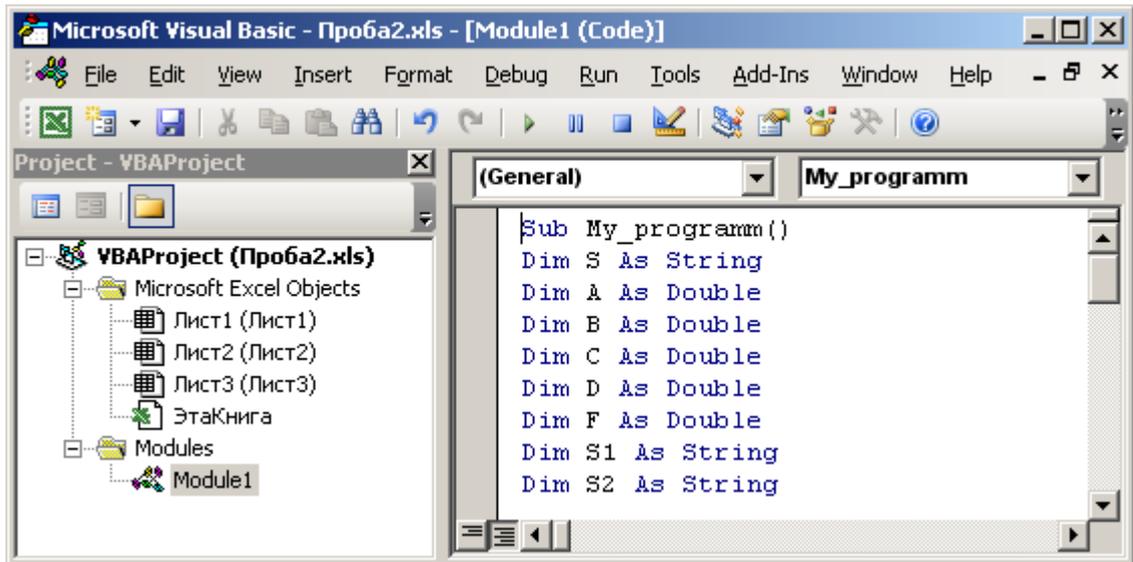
2) определение значения выражения (*F*) и вывод его на экран;

$$F = \frac{1 - \text{целая часть от деления} \left(\frac{\text{остаток от деления } (D / (3 + \sin(A^\circ)))}{100 + 0,01 \times B^C} \right)}{\text{arctg}(\sqrt{B})}$$

3) построение и вывод на экран строки, состоящей из указанных частей:

а) строка *S* без второго и последнего символа; б) строка 1, первый символ, которой заменяется символом с ASCII-кодом, равным 125; в) строка – текущее значение секунд.

Решение



Программа:

Sub My_programm() ' Объявление процедуры My_programm

Dim S As String

' Объявление (создание) переменной по имени S

' для хранения строк

Dim A As Double

' Объявление (создание) переменной по имени A

' для хранения вещественных чисел

Dim B As Double

Dim C As Double

Dim D As Double

Dim F As Double

Dim S1 As String

Dim S2 As String

Dim S3 As String

S = InputBox("Введите строку (S):")

' Отображение диалогового окна с полем ввода, текстом

' приглашения " Введите строку (S):".

' Запись в переменную S текста введенного пользователем

' в поле после нажатия кнопки ОК

```

A = InputBox("Введите число (A):")
B = InputBox("Введите число (B):")
C = 3.5 ' Запись в переменную C значения 3.5
D = 6
F = D Mod 3 + Sin(A * Excel.WorksheetFunction.Pi / 180)
' Запись в переменную F значения рассчитанного выражения.
' При вычислении Sin используется перевод в радианы
F = F \ (100 + 0.01 * (B ^ C))
' Расчет нового значения переменной F на основе предыдущего
F = 1 - F
F = F / (Atn(Sqr(B)))
MsgBox "F = " & F
' Вывод в окне сообщения значения переменной F.
' Сообщение формируется соединением строки "F = "
' со значением переменной F
S1 = Mid(S, 1, 1) & Mid(S, 3, Len(S)-3)
' Запись в переменную S1 построенной строки
S2 = S1
Mid(S2, 1, 1) = Chr(125)
' Замена первого символа строки в переменной S2
' на символ, возвращаемый функцией Chr по его коду (125)
S3 =Second(Now)
' Запись в переменную S3 текущего значения секунд
S = S1 & S2 & S3
' Соединение строк-значений переменных S1, S2 и S3
' в переменную S
MsgBox "S_new = " & S
' Вывод в окне сообщения значения переменной S.
End Sub ' Завершение объявления процедуры My_programm

```

Результаты работы программы:



Задание

Написать программу для выполнения следующей последовательности действий.

1) ввод ...

... с использованием функции **Inputbox** текстовой строки (переменная S),

... с использованием функции **Inputbox** двух чисел (переменные A и B),

... непосредственно в тексте программы двух чисел (переменные C и D);

- 2) определение значения выражения (F, табл. 9) и вывод его на экран;
 3) построение строки, состоящей из указанных частей (табл. 9), и вывод ее на экран.

Таблица 9

Варианты заданий к лабораторной работе № 2

№ варианта	Выражение F; части итоговой строки
1	$F = \frac{ A - B \times \sin(3 \times A^\circ) - \operatorname{tg}(D^\circ)}{1 + \ln B};$ <p>1) строка от первого символа до третьего (включительно) строки S прописными символами; 2) символ пробела; 3) строка от четвертого (включительно) до последнего символа строки S</p>
2	$F = \text{остаток от деления} \left(\frac{A}{B} \right) + \sqrt{1 + e^C \times D} - C^D;$ <p>1) строка от второго (включительно) символа строки S до четвертого (включительно); 2) строка прописными символами от третьего символа от конца строки S до второго от конца строки</p>
3	$F = \text{целая часть от деления} \left(\frac{A}{B} \right) - \operatorname{arctg} \left(\frac{ C - D }{C^D} \right);$ <p>1) строка, состоящая из ASCII-кодов первых трех символов фамилии, разделенных пробелами; 2) строка прописными символами из двух последних символов строки S</p>
4	$F = \text{целая часть от деления} \left(\frac{A + B}{B} \right) + \frac{\operatorname{arctg}(C + D)}{\text{случайное число из диапазона } [1...3]};$ <p>1) строка от первого до второго символа (включительно) строки S; 2) третий символ строки S прописными символами; 3) строка ASCII-кода последнего символа строки S</p>

5	$F = \frac{\frac{A+B}{B} + \operatorname{arctg}(2^C)}{ D };$ <p>1) строка от второго символа (включительно) строки S до конца строки; 2) ASCII-код четвертого символа строки S; 3) строка от третьего символа от конца строки S до второго символа от конца строки S</p>
6	$F = \text{целая часть от деления} \left(\frac{A + (\arccos(D))}{B^C} \right);$ <p>1) строка – ASCII-код последнего символа строки S; 2) строка S без второго и третьего символа; 3) строка от второго символа (включительно) строки S до предпоследнего символа (включительно) строки S</p>
7	$F = \arccos \left(e^{\frac{A+B}{D^3+C}} \right);$ <p>1) строка с обращенным регистром от первого символа (включительно) строки S до второго символа (включительно) строки S; 2) строка S без первого и последнего символа; 3) строка – ASCII-код последнего символа</p>
8	$F = \sqrt{1 - A - B^{\frac{C+D}{\cos(C^{\circ})}}};$ <p>1) строка S без второго и предпоследнего символа; 2) четвертый символ строки S; 3) строка из символов, соответствующих ASCII-кодам: 235, 229, 241</p>
9	$F = (1 - B)^{\sin(2A^{\circ})} + \log_{\text{случайное число из диапазона } [3...5]} C^D ;$ <p>1) строка от третьего (включительно) символа до четвертого (включительно) символа строки S; 2) строка из символов, соответствующих ASCII-кодам: 243, 242, 229, 241; 3) строка – результат расчета: F+1</p>

10	$F = \frac{\text{остаток от деления} \left(\frac{C + \text{ctg}(B^\circ)}{ A^D } \right)}{\sqrt{D}};$ <p>1) строка от четвертого (включительно) до последнего символа строки S; 2) строка – последний символ строки S; 3) строка – количество символов в строках 1 и 2</p>
11	$F = \arctg \left(1 - \frac{A}{B} - \text{случайное число из диапазона } [7...9]^{C^{\sqrt{D}}} \right);$ <p>1) строка из символов, соответствующих ASCII-кодам: 227, 238, 240, 224; 2) строка от третьего символа от конца строки S до конца строки S; 3) символ пробела 4) строка S без второго и третьего символов</p>
12	$F = \frac{1 - \frac{1}{ A } - \frac{\sqrt{B}}{B+C}}{D^{\arccos(A)}};$ <p>1) строка из прописных символов, соответствующих ASCII-кодам: 226, 251, 241, 252; 2) символ пробела; 3) строка S без второго символа</p>
13	$F = \left(A + \frac{1}{e^{B+C}} \right) \times (\ln D)^D;$ <p>1) строка – результат расчета: F+11; 2) строка 1 без последнего символа; 3) строка прописными символами от второго символа (включительно) до четвертого символа (включительно) строки S</p>
14	$F = (\log_7 A) + \frac{\sqrt{B}}{C^D + \cos(B^\circ)};$ <p>1) строка S без второго и последнего символа; 2) строка S, вместо символов с третьего (включительно) по четвертый (включительно) в которой, помещена строка из символов, соответствующих ASCII-кодами: 242, 242</p>

15	$F = \operatorname{tg} \left(\text{остаток от деления} \left(\frac{A^2}{B+C} \right)^\circ \right) - (\text{знак выражения}(D-A)) \times C;$ <p>1) строка – последние два символа текущей даты; 2) строка S, вместо символов с первого (включительно) по третий (включительно) в которой, помещена строка из символов, соответствующих ASCII-кодам: 234, 238, 235</p>
16	$F = \left(\frac{1}{A+B} + D ^e \right) \times \left(\cos(C^\circ) - \text{целая часть от деления} \left(\frac{B}{D} \right) \right);$ <p>1) строка – текущее значение секунд без первого символа; 2) строка S без второго и предпоследнего символа</p>
17	$F = \sqrt{\frac{e^{ 2-B }}{C + (1 - D \times \text{случайное число из диапазона [2...4])}}};$ <p>1) строка S в обращенном регистре без трех последних символов; 2) строка S, вместо символов со второго (включительно) по четвертый (включительно) в которой, помещена строка из символов, соответствующих ASCII-кодами: 242, 238, 240</p>
18	$F = \left(\text{знак выражения} \left(\frac{A^B}{1-B-A} \right) \right) \times \sin((D + 2^e)^\circ);$ <p>1) строка S без четвертого и последнего символа; 2) строка суммы количества секунд и минут; 3) строка – длина строки 2</p>
19	$F = \left(1 - \frac{A}{ C-1 } \right) + \log_{D+3}(B^2);$ <p>1) строка, составленная из символов, имеющих ASCII-коды: 225, 236, 225, 236, причем первый и третий символы – прописные; 2) строка S без второго, третьего и четвертого символа</p>
20	$F = 2^e + \left(\text{остаток от деления} \left(\frac{A}{C + \cos((10 + D)^\circ)} \right) \right) \times (B - 1);$

	<p>1) строка S, вместо двух последних символов в которой, помещена строка из символов, соответствующих ASCII-кодам: 232, 235; 2) строка – текущее значение минут; 3) строка – количество символов в строке 2</p>
21	$F = 2^{A - \frac{B}{C} \times \cos((2 \times D)^\circ)} + \sqrt{\text{случайное значение из диапазона [5...8]};$ <p>1) строка S без третьего и четвертого символа; 2) строка – текущее календарное число; 3) символ пробела; 4) строка – номер текущего месяца</p>
22	$F = \left \frac{\arccos\left(1 - \frac{A}{B}\right)}{e^e + \operatorname{tg}\left(10 + \frac{C}{D}\right)^\circ} \right ;$ <p>1) строка S с символом, ASCII-код которого равен 124, вместо второго символа; 2) строка S без второго и третьего символа от конца строки</p>
23	$F = \sqrt{\cos(A^\circ) + B^D} + \log_{2 \times C } D;$ <p>1) строка S без третьего и четвертого символа от конца строки; 2) строка S, вместо двух первых символов которой, помещена строка из символов, соответствующих ASCII-кодами: 228, 238</p>
24	$F = \left(1 + \ln A \frac{A}{B}\right)^{\cos((1/C)^\circ)} + \sqrt{D};$ <p>1) строка S с символом, ASCII-код которого выбирается случайным образом из диапазона [1 ... 256], вместо третьего символа; 2) символ пробела; 3) строка от третьего (включительно) до предпоследнего (включительно) символа строки S</p>

25	$F = \left(A + \cos((0,3 \times B)^0) \right)^{\frac{(\text{знак выражения}(C-D)) \times A}{2}} ;$ <p>1) строка – текущее значение года без двух первых цифр; 2) строка S без первого и предпоследнего символа</p>
26	$F = e^{\frac{1}{\sqrt{A+2 \times B}}} + \ln(16 \times C) - \arctg(D) ;$ <p>1) строка от первого (включительно) до второго (включительно) от конца строки S символа; 2) строка, составленная из двух символов, ASCII-коды которых выбираются случайным образом из диапазона [1 ... 256]; 3) текущая дата без первого и последнего символа</p>
27	$F = \left(1 - \frac{\sqrt{1 - A^B}}{\cos(C^\circ)} \right) + \left \text{случайное число из диапазона [6...8]} \right ^2 ;$ <p>1) строка S без второго и третьего символа от конца строки; 2) строка прописных символов от третьего (включительно) до четвертого (включительно) символа строки S</p>
28	$F = \frac{\arccos\left(\frac{\sqrt{A}}{D}\right)}{\text{целая часть от деления}\left(\frac{e^B}{B - C }\right)} ;$ <p>1) строка S без четвертого и пятого символа; 2) строка S, вместо символов со второго (включительно) по четвертый (включительно) в которой, помещена строка из символов, соответствующих ASCII-кодами: 234, 238, 228</p>
29	$F = \left(1 - \frac{1}{B} \right)^{\sin(A^\circ)} + \sqrt{C^e + D} ;$ <p>1) строка – результат расчета F + количество символов в строке S; 2) строка S без первого и третьего символов; 3) символ пробела; 4) строка – количество секунд</p>

30	$F = \frac{A}{D^{(1-A-B)} + \sqrt{C}} \times \text{arctg}(D) ;$ <p>1) строка S без двух первых и двух последних символов; 2) строка – текущее значение минут без первого символа</p>
----	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Состав отчета

1. Номер, название и цель работы.
2. Текст задания.
3. Листинг (текст программы).
4. Результаты работы программы, выводимые на экран.

Контрольные вопросы

1. Понятие о переменных. Приведите способы объявления переменных.
2. Перечислите основные типы данных. Для хранения каких данных они предназначены?
3. Приведите примеры использования функций для работы с типом данных *Date*.
4. Какие Вам известны функции для проверки типов данных?
5. Какие Вам известны функции для изменения типов данных?
6. Приведите примеры использования математических операторов и функций.
7. Назначение логических операторов.
8. Приведите примеры использования текстовых функций.
9. Охарактеризуйте назначение функции *MsgBox*. Для чего предназначены опции функции *MsgBox*?
10. Охарактеризуйте назначение функции *InputBox*. Для чего предназначены опции функции *InputBox*?

ЛАБОРАТОРНАЯ РАБОТА № 3. УСЛОВНЫЕ ОПЕРАТОРЫ

Цель работы

Изучить синтаксис условных операторов и получить навыки их использования в программах.

Теоретические сведения

VB может проверять результат некоторого условия и в зависимости от результата проверки выполнять различные действия (ветки программы) с помощью следующих управляющих операторов: **If...Then**, **If...Then...Else**, **Select...Case**.

Операторы If...Then и If...Then...Else

Синтаксис оператора **If...Then**:

If <условие> Then <оператор>

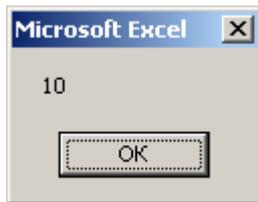
Управляющий оператор **If...Then** предназначен для проверки условия (**<условие>**) и выполнения одного оператора (**<оператор>**) в случае, если результат проверки условия равен **True (Истина)**.

Пример:

```
Sub primer()
Dim diskont As Boolean
' Объявление логической
' переменной diskont
Dim skidka As Double
skidka=0
diskont=True
' Запись в логическую
' переменную значения
' True (Истина)
if diskont Then skidka=10
```

```
Sub primer()
Dim diskont As Boolean
' Объявление логической
' переменной diskont
Dim skidka As Double
skidka=0
diskont=False
' Запись в логическую
' переменную значения
' False (Ложь)
if diskont Then skidka=10
```

```
' Запись в переменную skidka
' значения, если переменная
' diskont содержит True
MsgBox skidka
' Отображение значения
' переменной skidka
End Sub
```



```
' Запись в переменную skidka
' значения, если переменная
' diskont содержит True
MsgBox skidka
' Отображение значения
' переменной skidka
End Sub
```



Если по результатам проверки необходимо выполнить более одного оператора, то следует воспользоваться следующим синтаксисом:

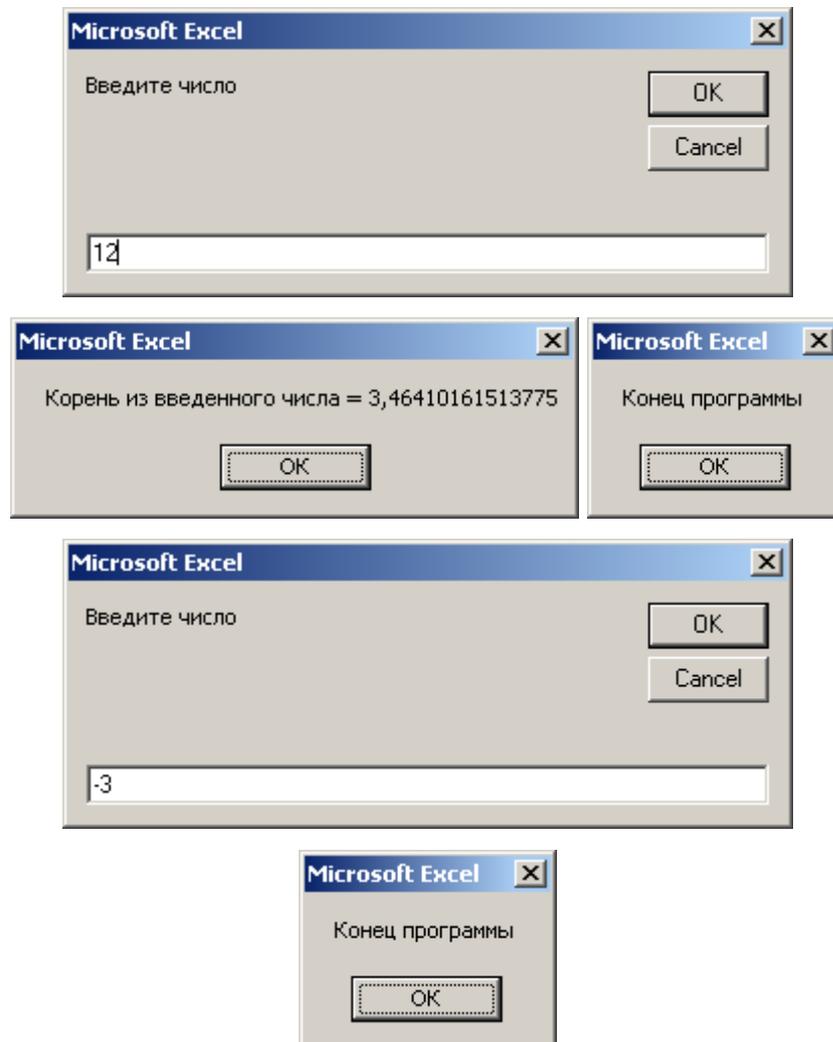
```
If <условие> Then
<оператор1>
<оператор2>
...
End If
```

Пример:

```
Sub primer()
Dim p As Double
' Объявление переменной для хранения вещественных чисел
p=InputBox ("Введите число")
' Отображение окна для ввода значения в переменную p
If p>=0 Then ' Действия в случае неотрицательного значения p
p=sqr(p) ' Запись в переменную p корня из значения p
Msgbox "Корень из введенного числа = " & p
' Отображение окна-сообщения со значением переменной p
End if
```

Msgbox "Конец программы"

' Окно сообщения, отображаемое при любом введенном р
End Sub



С помощью управляющего оператора **If...Then...Else** можно задать две последовательности действий, одна из них будет выполняться, если условие истинно (**Then <операторы>...**), а другая – если условие ложно (**Else <операторы>...**). Синтаксис:

If <условие> Then

<операторы>...

Else

<операторы>...

End If

Для проверки нескольких условий можно использовать логические операторы: **AND**, **OR**, **XOR**, **NOT**, а также их комбинации.

Пример:

Sub primer()

Dim x As Variant ' Объявление переменной для любых значений

x = InputBox("Введите положительное число")

' Отображение окна для ввода значения в переменную x

If (IsNumeric(x) And x >= 0) Then ' Условие на значение переменной

MsgBox "Спасибо, Вы ввели положительное число!"

' Окно-сообщение, отображаемое при выполнении условия

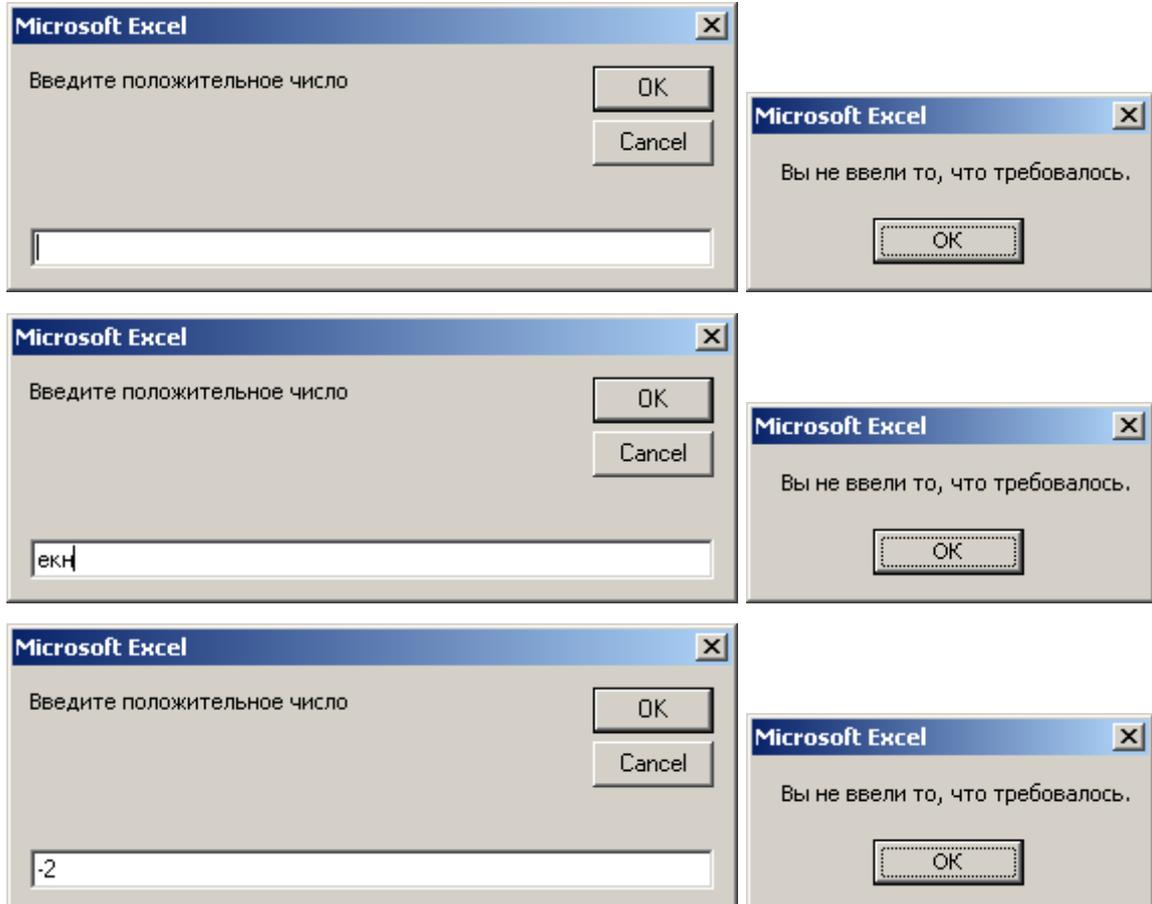
Else

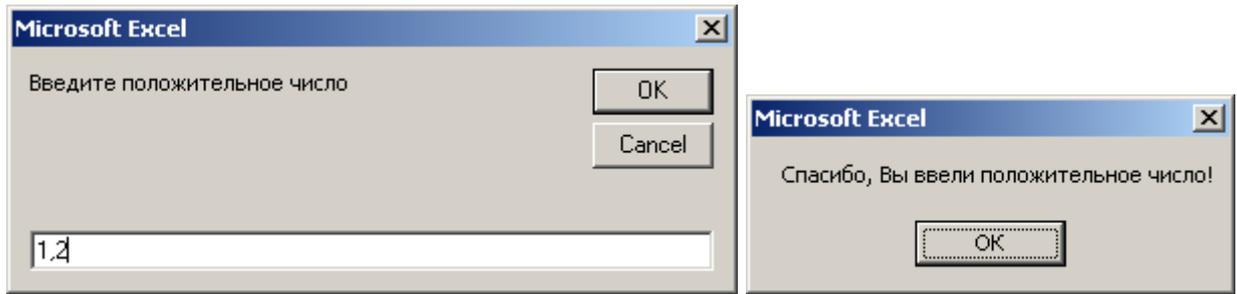
MsgBox "Вы не ввели то, что требовалось."

' Окно-сообщение, отображаемое при невыполнении условия

End If

End Sub





В составе операторов внутри конструкции **If ...** можно применять вложенные конструкции **If ...** , н.п.:

Sub primer()

Dim x As Variant ' Объявление переменной для любых значений

x = InputBox("Введите что-нибудь")

' Отображение окна для ввода значения в переменную x

If (x = "") Then ' Внешний оператор проверки на пустой ввод

MsgBox "Вы ничего не ввели."

Else

If (Not (IsNumeric(x))) Then ' Вложенный оператор проверки

MsgBox "Вы ввели текст."

Else

MsgBox "Вы ввели число."

End If ' Закрытие вложенного оператора проверки

End If ' Закрытие внешнего оператора проверки

End Sub

Иногда вместо вложенных **If...** удобнее использовать оператор **Elsif**. При этом каждая следующая проверка происходит только в том случае, если результат предыдущей равен **False (Ложь)**. Синтаксис:

If <условие> Then

<операторы>...

Elsif <условие> Then

<операторы>...

Else

<операторы>...

End If

Пример:

Sub primer()

Dim p As Double

' Объявление переменной для хранения вещественных чисел

p=InputBox("Введите число")

' Отображение окна для ввода значения в переменную p

If p<3 then ' Проверка первого условия на значение p

Msgbox "p<3" ' Действие при выполнении первого условия

Elseif p>6 then ' Проверка второго условия на значение p

' (при невыполнении первого)

Msgbox "p>6" ' Действие при выполнении второго условия

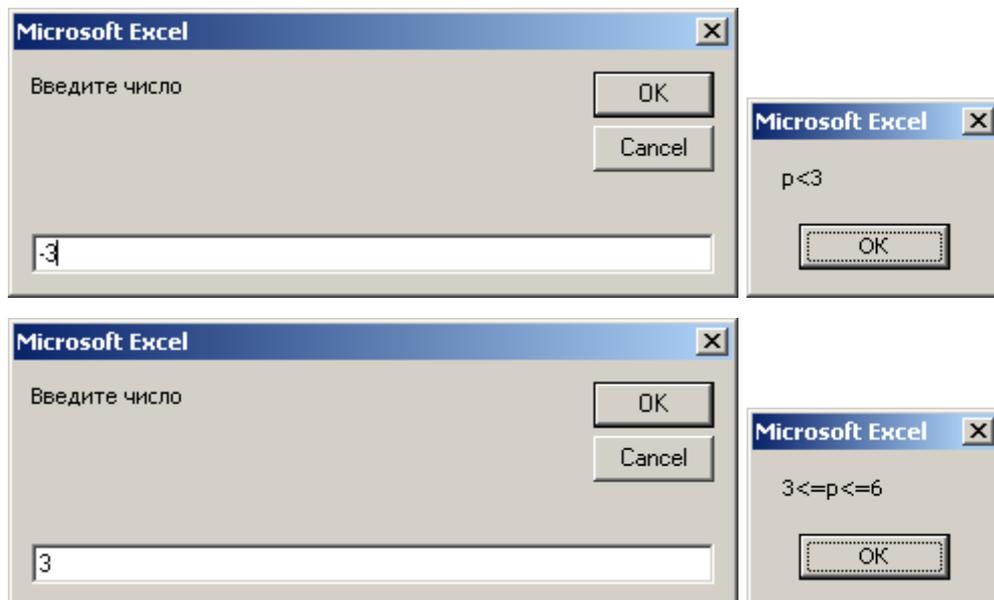
Else

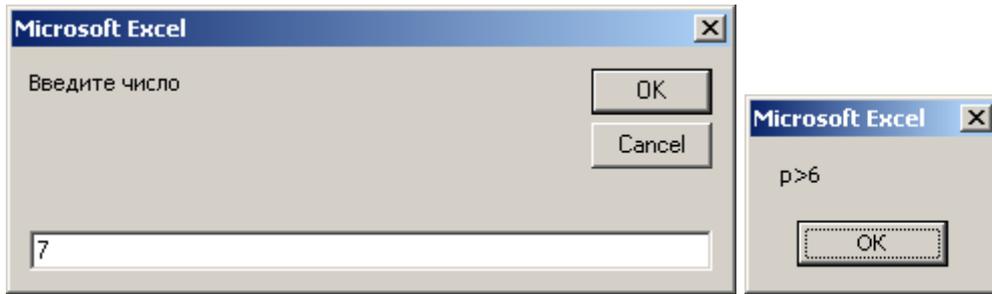
' Действия при невыполнении ни первого ни второго условий

MsgBox "3<=p<=6"

End If

End Sub





Оператор Select Case

Когда проверяется результат одного выражения, который может иметь много различных значений и для каждого из этих значений надо выполнить свою уникальную последовательность действий, удобнее пользоваться управляющим оператором **Select Case** вместо **If...Then...Elseif**. Синтаксис:

Select Case <выражение>

Case <значение 1>

<операторы>...

Case <значение N>

<операторы>...

Case Else

<операторы>...

End Select

Элемент **<значение>** может записываться как непосредственно искомое значение выражения для выполнения группы операций, перечисление искомых значений через ",", диапазон искомых значений (**<начало> To <конец>**), включая границы, или условие при помощи операнда **Is**.

Sub SelectOp()

Dim A As Integer ' Объявление переменной для целых чисел

Dim Msg As String ' Объявление переменной для хранения строк

A = 4 ' Запись в переменную целого числа

Select Case A ' Оператор проверки значения переменной A

Case Is <0 ' Условие, если переменная меньше 0

Msg = "I < 0" ' Запись строки в переменную Msg

Case 0 To 3 ' Условие, если переменная от 0 до 3, включительно

Msg = "1 <= A <= 3"

Case 4, 5, 6, 7, 8, 9, 10 ' Условие, если A от 4 до 10, включительно

Msg = "4 <= A <= 10"

Case 1s<=11 ' A от 10 (невключительно) до 11 (включительно)

Msg = "10 < A <= 11"

Case Else ' Все оставшиеся значения A (Число больше 11)

Msg = "A > 11"

End Select

MsgBox Msg ' Вывод сообщения со значением переменной Msg

End Sub

После альтернативы **Case Else** находятся операторы, которые выполняются в том случае, если число не попало ни в один из проверяемых диапазонов.

Пример 1

Составить алгоритм и нарисовать блок-схему для вычисления функции

$$y = \sqrt{\frac{2-x}{\ln x}}$$

(выражение имеет ограничения в области определения). Составить программу вычисления функции при различных значениях аргумента (x).

Решение примера 1

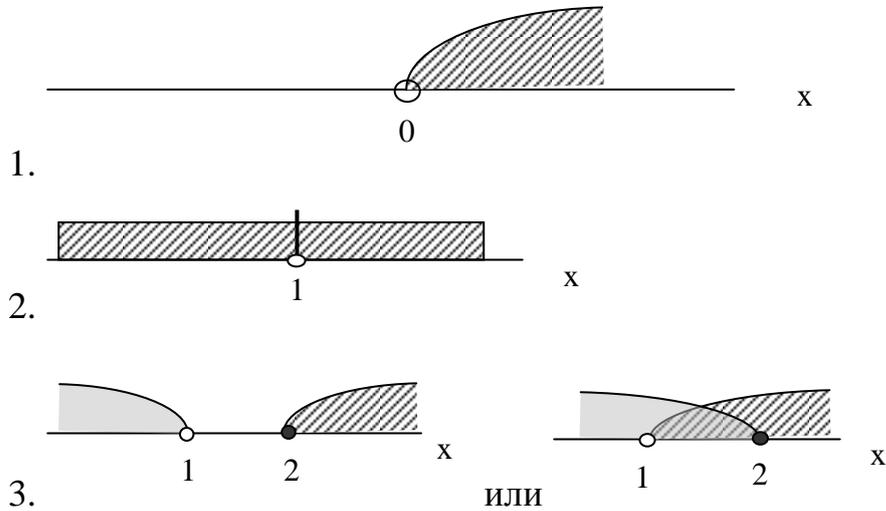
На область определения $y = f(x)$ накладываются следующие ограничения:

$$\begin{cases} 1. x > 0 \\ 2. x \neq 1 \\ 3. (2-x \geq 0 \text{ и } \ln x > 0) \text{ или } (2-x \leq 0 \text{ и } \ln x < 0) \end{cases}$$

Условие 1 обусловлено аргументом логарифма, условие 2 определяется отсутствием 0 в знаменателе дроби, условие 3 определяет неотрицательность

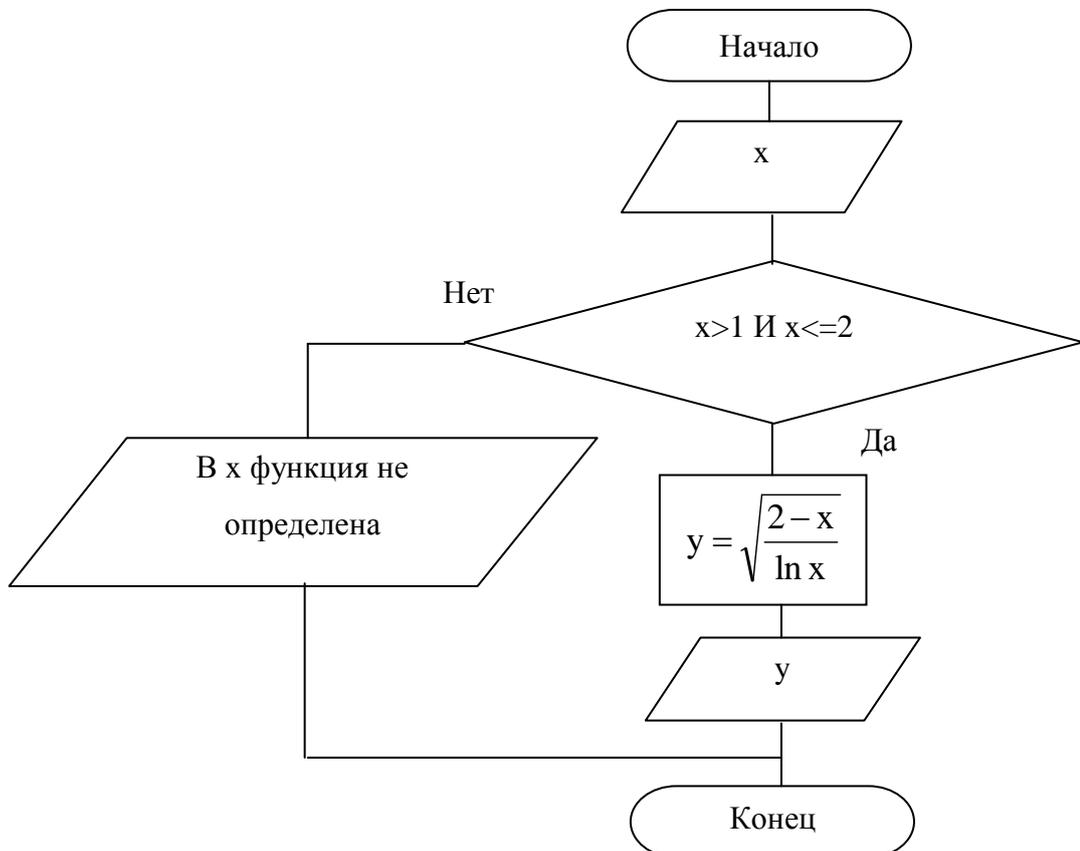
подкоренного выражения. Изобразим ограничения области определения

функции $y = \sqrt{\frac{2-x}{\ln x}}$ на оси x .



Суммируя все ограничения, получим область определения $y = f(x)$: $x = (1, 2]$.

Алгоритм расчета функции $y = \sqrt{\frac{2-x}{\ln x}}$ представлен в блок-схеме.



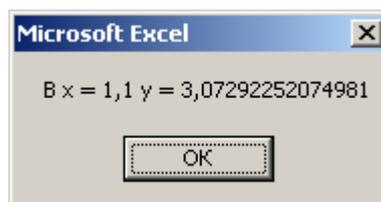
Алгоритм может быть реализован на VB следующим образом:

```

Sub primer1()
Dim x As Double ' Объявление переменной для аргумента
Dim y As Double ' Объявление переменной для функции
x = 1.1 ' Ввод значения аргумента функции
If (x > 1 And x <= 2) Then
' Оператор проверки значения аргумента на вхождение
' в область определения
y = Sqr((2 - x) / Log(x)) ' Вычисление функции
MsgBox "В x = " & x & " y = " & y
' Отображение аргумента и вычисленной для него функции
Else
MsgBox "В x = " & x & Chr(10) & Chr(13) & "функции нет"
' Отображение сообщения о невозможности рассчитать функцию,
' т.к. аргумент не входит в область определения
End If ' Завершение оператора проверки
End Sub

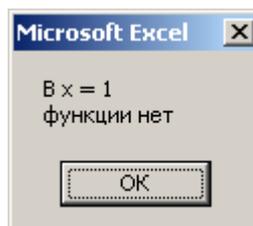
```

Результаты работы программы:



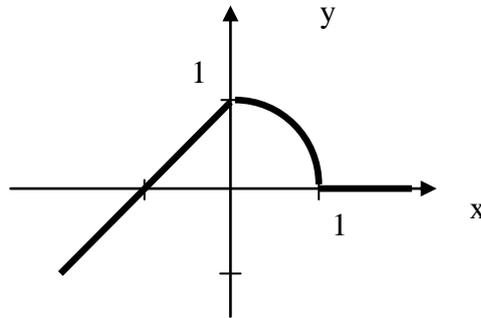
Для

x = 1 ' Ввод значения аргумента функции



Пример 2

Представить аналитическими выражениями функцию $y = f(x)$, заданную графически.



Изобразить блок-схему и написать программу для расчета $y = f(x)$ при различных значениях x .

Решение примера 2

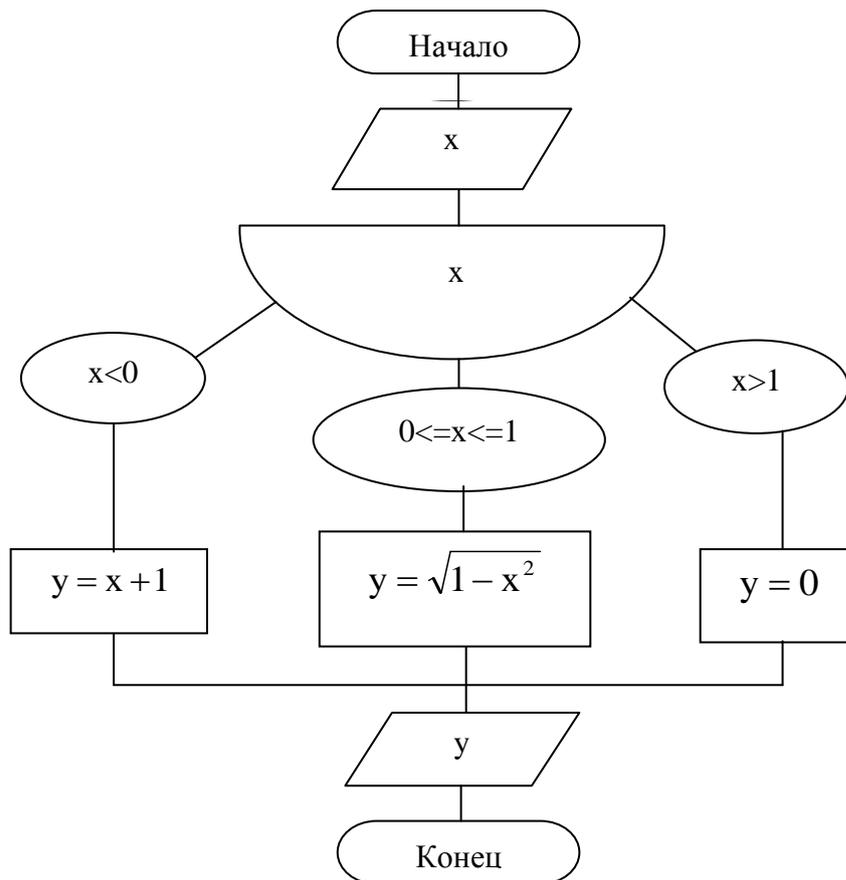
Функция $y = f(x)$ – непрерывная и рассчитывается для трех интервалов значений аргумента: 1. $x = (-\infty, 0)$, 2. $x = [0, 1]$, 3. $x = (1, +\infty)$:

интервал 1 описывается уравнением $y = x + 1$,

интервал 2 описывается уравнением $x^2 + y^2 = 1$ или $y = \sqrt{1 - x^2}$,

интервал 3 описывается уравнением $y = 0$.

Алгоритм (блок-схема) расчета функции $y=f(x)$ представлен на рисунке.



Алгоритм может быть реализован на VB следующим образом:

Sub primer2()

Dim x As Double ' Объявление переменной для аргумента

Dim y As Double ' Объявление аргумента для функции

x = 0.5 ' Ввод значения аргумента функции

Select Case x ' Оператор проверки значения аргумента

Case Is < 0 ' Условие принадлежности к первому интервалу

y = x + 1 ' Вычисление функции на первом интервале

Case Is <= 1 ' Условие принадлежности ко второму интервалу

y = Sqr(1 - x * x) ' Вычисление функции на втором интервале

Case Else ' Принадлежность аргумента третьему интервалу

y = 0 ' Вычисление функции на третьем интервале

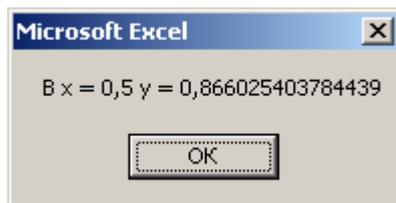
End Select

MsgBox "В x = " & x & " y = " & y

' Отображение аргумента и вычисленной для него функции

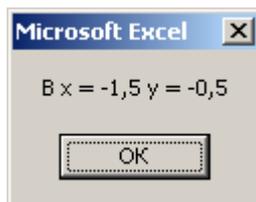
End Sub

Результаты работы программы:



Для

x = -1.5 ' Ввод значения аргумента функции

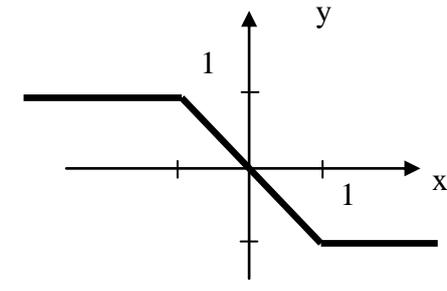
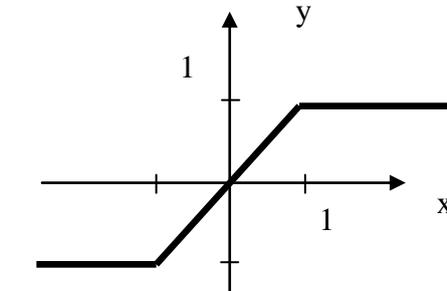
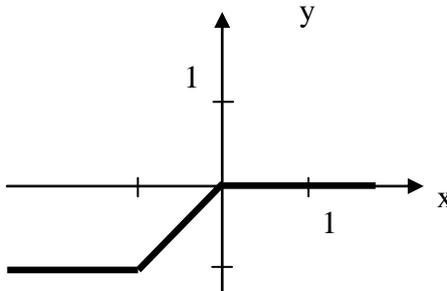
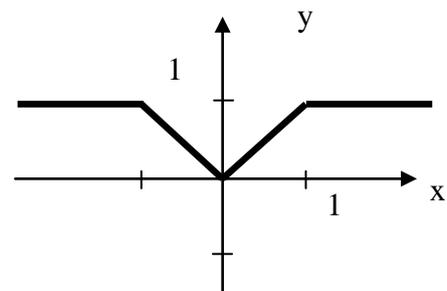
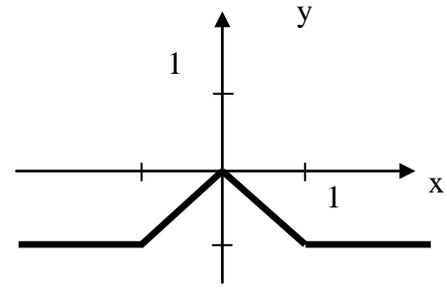


Задание

Таблица 10

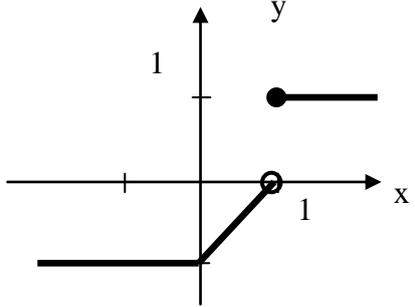
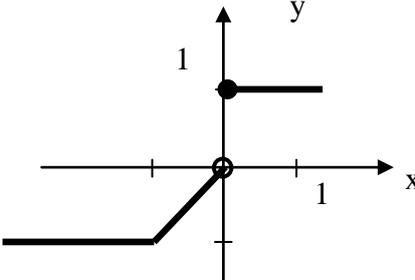
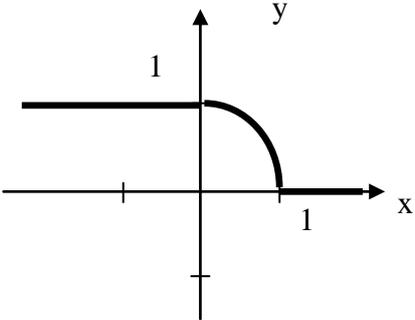
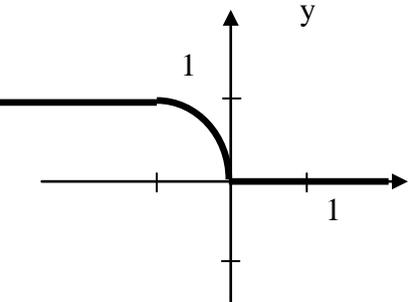
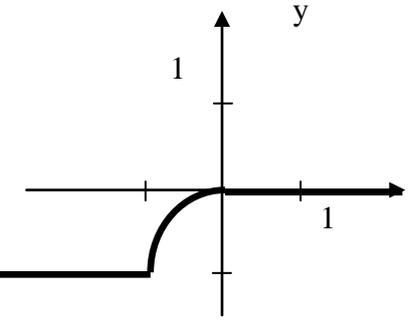
Варианты заданий к лабораторной работе № 3

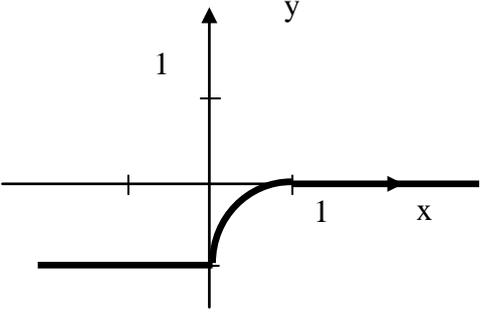
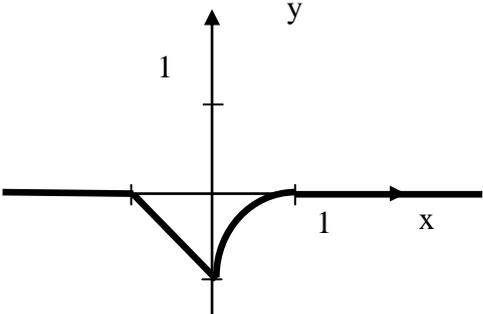
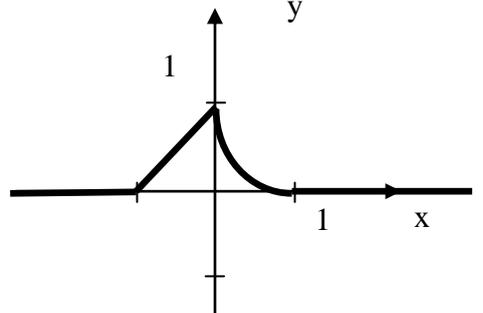
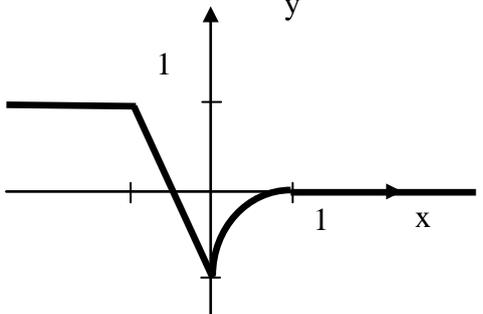
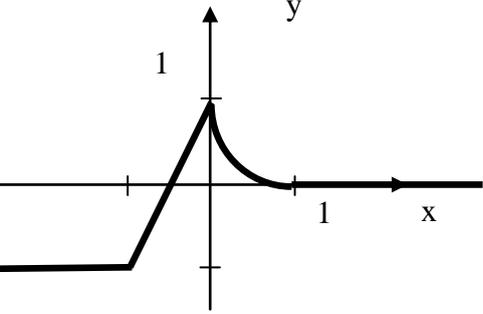
№ варианта	А. Составить алгоритм и нарисовать блок-схему для вычисления функции $y = f(x)$ (выражение имеет ограничения в области определения). Составить программу вычисления $y = f(x)$ при различных x .	Б. Представить аналитическими выражениями функцию $y = f(x)$, заданную графически. Изобразить блок-схему и написать программу для расчета $y = f(x)$ при различных x .
1	$y = \frac{1}{x}$	

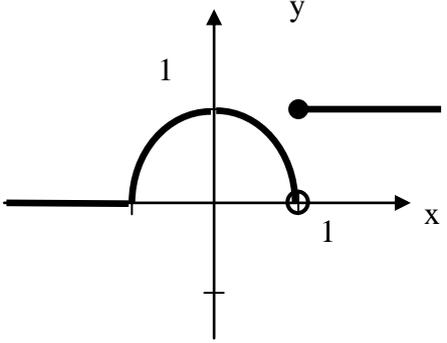
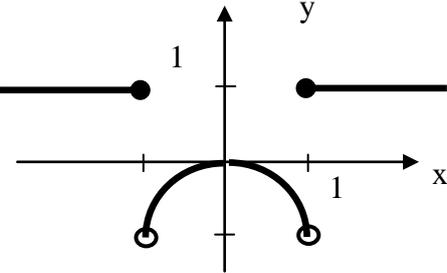
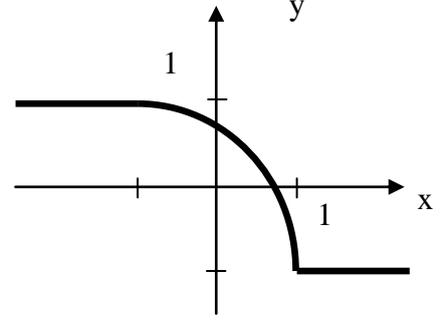
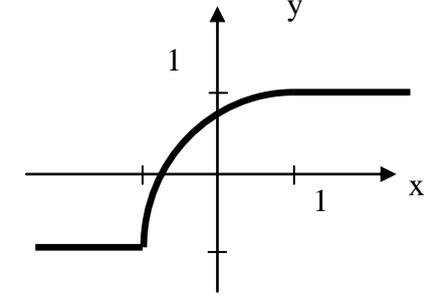
2	$y = \frac{1}{x^2 - x - 12}$	
3	$\sqrt{1-x}$	
4	$\sqrt{\frac{\ln x}{x}}$	
5	$y = \frac{1}{\sqrt{2-x}}$	
6	$y = \frac{\ln x}{x-1}$	

7	$y = \sqrt{1 - \ln x}$	
8	$y = \sqrt{\frac{x+2}{x-2}}$	
9	$y = \frac{1}{\sqrt{\ln x}}$	
10	$y = \sqrt{\ln(x-1) \times \ln x}$	
11	$y = \frac{1}{x-1}$	

12	\sqrt{x}	
13	$\ln x$	
14	$y = \frac{1}{x \times (x-1)}$	
15	$y = \frac{\ln(4-x)}{\sqrt{x-3}}$	
16	$y = \frac{\ln(3-x)}{x^2 - x - 12}$	

17	$y = \sqrt{\frac{x^3}{x-2}}$	 <p>The graph shows a piecewise function on a Cartesian coordinate system. The x-axis has a tick mark at 1, and the y-axis has a tick mark at 1. The function consists of three segments: a horizontal line at a negative y-value for x < 0; a line segment from (0,0) to (1,1) with an open circle at (1,1); and a horizontal line at a positive y-value for x > 1, starting with a closed circle at (1,1).</p>
18	$y = \sqrt{\frac{x-2}{(x+2)(3-x)}}$	 <p>The graph shows a piecewise function on a Cartesian coordinate system. The x-axis has a tick mark at 1, and the y-axis has a tick mark at 1. The function consists of three segments: a horizontal line at a negative y-value for x < -2; a line segment from (-2,0) to (0,0) with an open circle at (0,0); and a horizontal line at a positive y-value for x > 0, starting with a closed circle at (0,1).</p>
19	$y = \frac{1}{1 - \ln x}$	 <p>The graph shows a curve on a Cartesian coordinate system. The x-axis has a tick mark at 1, and the y-axis has a tick mark at 1. The curve starts at a high positive y-value for small x, passes through the point (1,1), and approaches the x-axis as x increases.</p>
20	$y = (\ln x) \times \ln(2-x)$	 <p>The graph shows a curve on a Cartesian coordinate system. The x-axis has a tick mark at 1, and the y-axis has a tick mark at 1. The curve starts at a high positive y-value for small x, passes through the origin (0,0), and approaches the x-axis as x increases.</p>
21	$y = \frac{1}{x^2 - x - 2}$	 <p>The graph shows a curve on a Cartesian coordinate system. The x-axis has a tick mark at 1, and the y-axis has a tick mark at 1. The curve starts at a high negative y-value for small x, passes through the origin (0,0), and approaches the x-axis as x increases.</p>

22	$\sqrt{\frac{1}{x}}$	
23	$\ln(x-1)$	
24	$y = \frac{\ln x}{x \times (x-2)}$	
25	$y = \frac{\ln(3-x)}{x^2 - x - 2}$	
26	$y = \sqrt{x^2 - x - 2}$	

27	$y = \sqrt{\frac{x-2}{x+2}}$	
28	$y = \frac{1}{\ln x}$	
29	$y = \sqrt{(x^2 - x - 2) \times \ln x}$	
30	$y = \frac{\ln x}{\ln(2-x)}$	

Состав отчета

1. Номер, название и цель работы.
2. Данные к варианту заданий А и Б.
3. Алгоритмы решения заданий А и Б.
4. Листинги (тексты программы) заданий А и Б.
5. Результаты работы программ (информация, выводимая на экран) заданий А и Б.

Контрольные вопросы

1. Охарактеризуйте разные виды синтаксиса управляющего оператора **If....**
2. Перечислите способы указания вариантов искомых значений в управляющем операторе **Case....**
3. Каким образом происходит проверка значения в операторе **Case...?**
4. Каким образом отображаются операторы **If...** и **Case...** на блок-схемах?

ЛАБОРАТОРНАЯ РАБОТА № 4. ОПЕРАТОРЫ ЦИКЛОВ

Цель работы

Изучить синтаксис операторов цикла и получить навыки их использования в программах.

Теоретические сведения

Оператор For...Next

Управляющий оператор **For...Next** предназначен для циклического выполнения некоторой последовательности действий заданное число раз.

Синтаксис:

For <переменная-счетчик> = <начало> **To** <конец> [**Step** <шаг>]

<операторы>...

[Exit For]

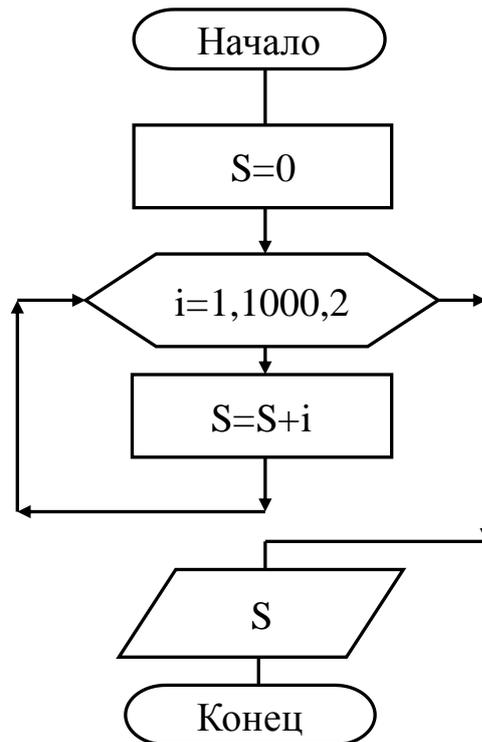
<операторы>...

Next <переменная-счетчик>

Здесь <начало> и <конец> – значения, в пределах которых изменяется <переменная-счетчик> с определенным шагом. Выход из цикла по некоторому условию, до исчерпания счетчика осуществляется при помощи оператора **Exit For**.

Пример использования оператора For...Next: вычисление суммы всех нечетных чисел, лежащих в диапазоне [1...1000].

Алгоритм (блок-схема):



Программа:

Sub primer()

Dim i As Integer ' Переменная для хранения нечетных чисел

Dim S As Long ' Переменная для хранения суммы чисел

S = 0 ' Задание начального значения суммы (0)

For i = 1 To 1000 Step 2

' Задание повторяющихся действий для значений

' переменной-счетчика (i) от 1 до 1000 с шагом 2

S = S + i

' Увеличение суммы (S) на следующее вычисленное

' с учетом шага (2) значение переменной-счетчика (i)

Next i

' Инструкция для вычисления следующего значения

' переменной-счетчика (переход к строке For i=...).

' Выход из цикла (переход к строке MsgBox...) происходит,

' когда i становится равным 1001

MsgBox S ' Отображение суммы нечетных чисел

End Sub

Результат:



Оператор Do...Loop

Управляющий оператор **Do...Loop** используется для организации циклов с заранее неизвестным числом повторений. Цикл **Do...Loop** повторяется до тех пор, пока не выполнится какое-то произвольно заданное условие. Причем в цикле **Do...Loop** проверка условия для выхода из цикла, может выполняться как перед началом цикла, так и после выполнения цикла.

Do [{While | Until} <условие>]

[<операторы>]

[Exit Do]

[<операторы>]

Loop

или

Do

[<операторы>]

[Exit Do]

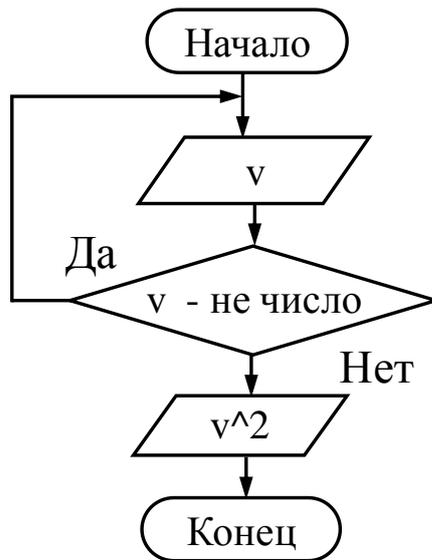
[<операторы>]

Loop [{While | Until} <условие>]

Ключевое слово **While** указывает на выполнение операторов в цикле, пока <условие> истинно (**True**), а **Until** – на выполнение цикла, пока <условие> ложно (**False**). Оператор **Exit Do** прекращает выполнение цикла по какому-либо внутреннему условию.

Пример использования оператора Do...Loop: возведение в квадрат введенного пользователем числа.

Алгоритм (блок-схема):



Программа:

Sub primer()

Dim v As Variant ' Объявление переменной для любых значений

Do

v = InputBox("Введите число")

' Отображение окна для ввода значения переменной v

Loop While IsNumeric(v) = False

' Инструкция, проверяющая отсутствие числа в переменной v

' и возвращающая (в этом случае) пользователя к строке Do...,

' т.е. повторному вводу переменной v

MsgBox v^2

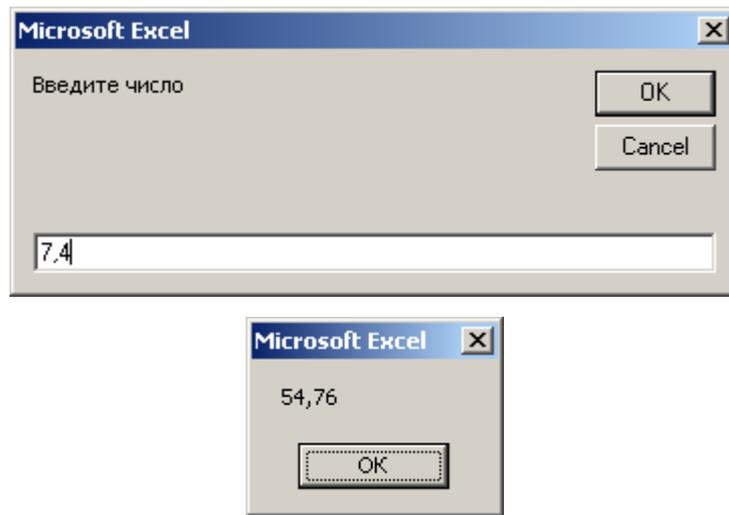
' Отображение квадрата значения переменной v

' после того, как в нее все-таки было введено число

End Sub

Результат:



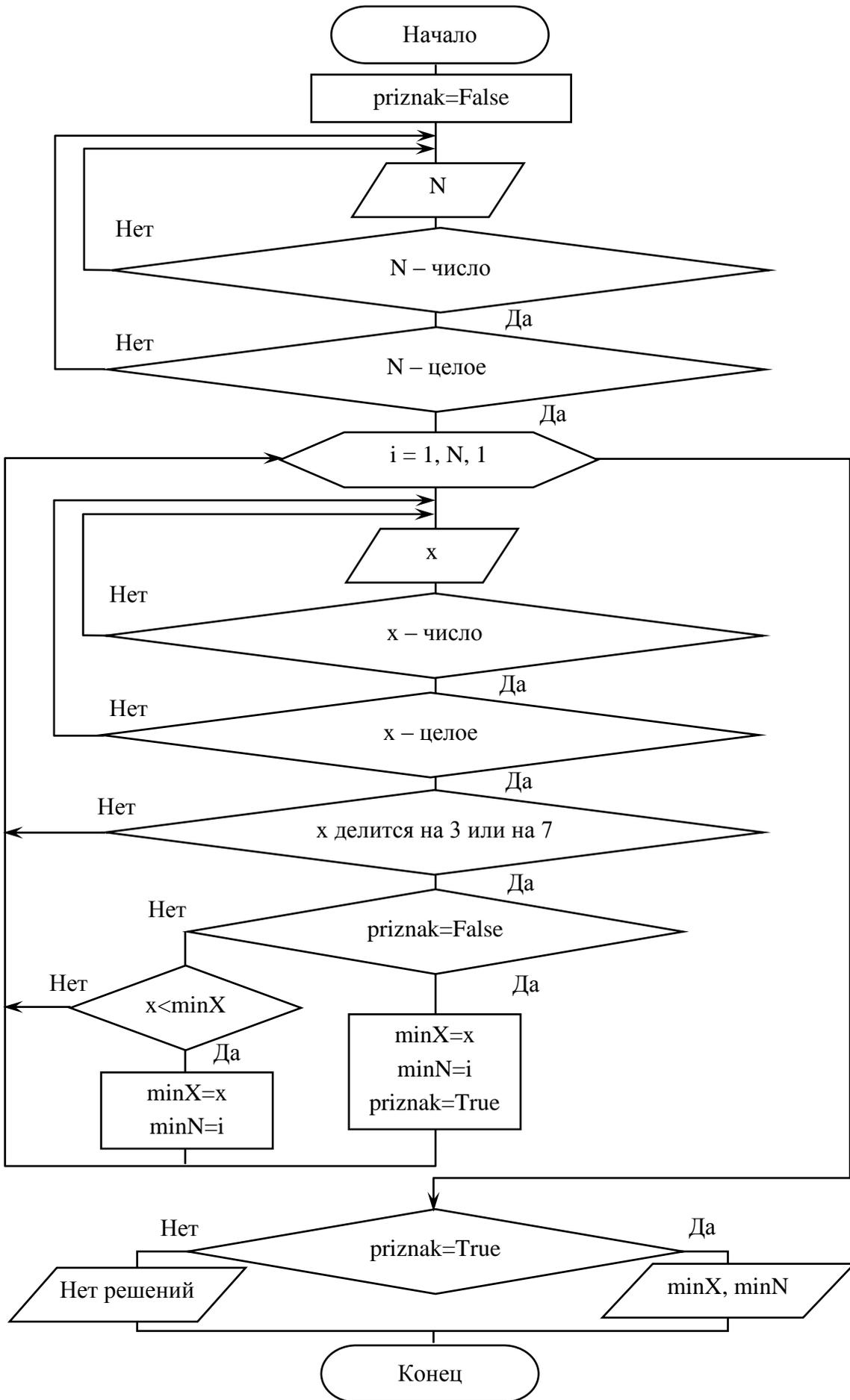


Пример

Дана произвольной длины последовательность целых чисел. Составить алгоритм и написать программу для нахождения наименьшего из чисел последовательности, делящихся нацело на 3 или на 7, и номера этого числа в последовательности. Количество чисел последовательности и сами числа задаются пользователем после запуска программы. Предусмотреть контроль значений, вводимых пользователем.

Решение

Алгоритм (блок-схема) решения задачи:



Программа:

Option Explicit

Sub primer()

Dim N As Variant

Dim i As Integer

Dim x As Variant

Dim priznak As Boolean

Dim minX As Integer

Dim minN As Integer

priznak = False

Do

Do

N = InputBox("Введите кол-во чисел в последовательности")

Loop Until IsNumeric(N)

Loop Until N = CInt(N)

For i = 1 To N Step 1

Do

Do

x = InputBox("Введите целое число № " & i)

Loop Until IsNumeric(x)

Loop Until x = CInt(x)

If x Mod 3 = 0 Or x Mod 7 = 0 Then

If priznak = False Then

minX = x

minN = i

priznak = True

Elseif x < minX Then

minX = x

minN = i

End If

End If

Next i

If признак = True Then

**MsgBox "Наименьшее число, делящееся на 3 или на 7, = " _
& minX & Chr(13) & "его № = " & minN**

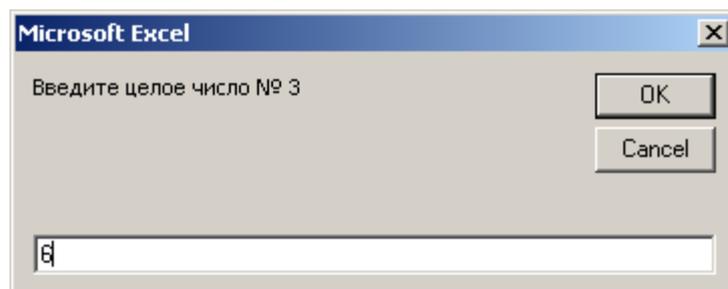
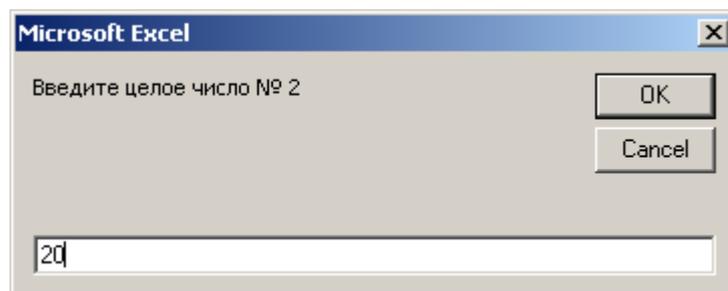
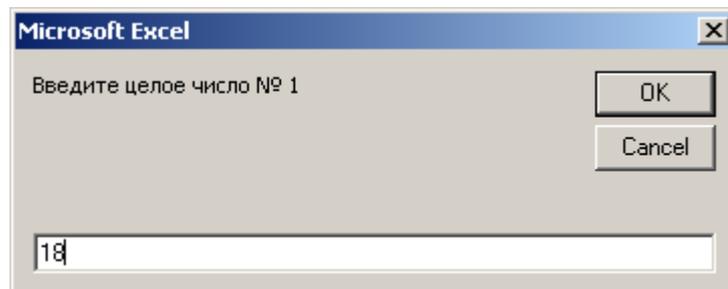
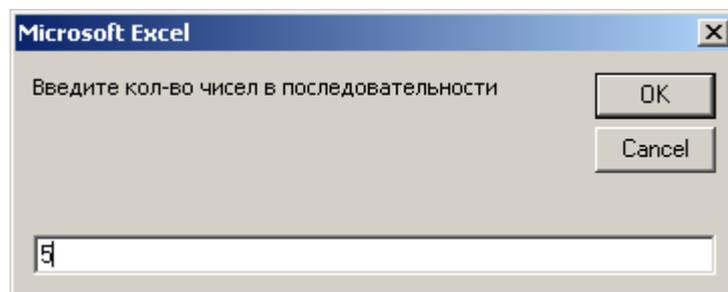
Else

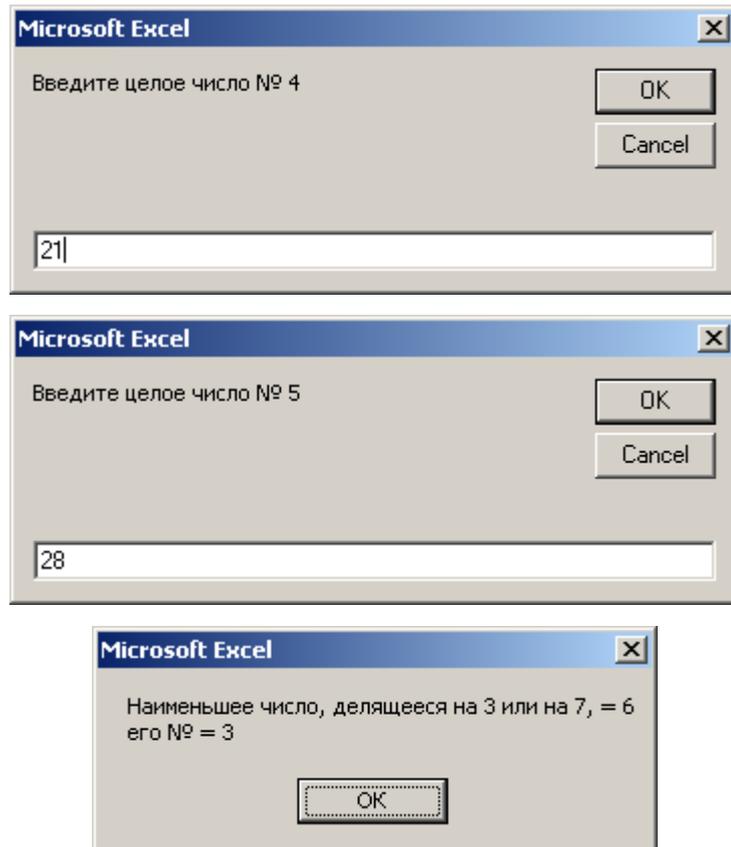
MsgBox "Нужных чисел нет"

End If

End Sub

Результаты работы программы:





Задание

Таблица 11

Варианты заданий к лабораторной работе № 4

№ варианта	Составить алгоритм и написать программу для выполнения описанных в варианте действий. Количество чисел последовательности и сами числа задаются пользователем после запуска программы. Предусмотреть контроль значений, вводимых пользователем.
1	Дана произвольной длины последовательность натуральных чисел. Найти произведение чисел, не делящихся нацело на 5, наибольшее из таких чисел, и номер этого числа в последовательности.
2	Дана произвольной длины последовательность целых чисел. Найти произведение чисел, заканчивающихся цифрой 2, наибольшее из таких чисел, и номер этого числа в последовательности.

	последовательности.
3	Дана произвольной длины последовательность вещественных чисел. Найти сумму чисел, не заканчивающихся цифрой 1, наименьшее из таких чисел, и номер этого числа в последовательности.
4	Дана произвольной длины последовательность вещественных чисел. Найти произведение отрицательных чисел, наибольшее из отрицательных чисел, и номер этого числа в последовательности.
5	Дана произвольной длины последовательность целых чисел. Найти сумму положительных чисел, наименьшее из положительных чисел, и номер этого числа в последовательности.
6	Дана произвольной длины последовательность натуральных чисел. Найти произведение нечетных чисел, наибольшее из нечетных чисел, и номер этого числа в последовательности.
7	Дана произвольной длины последовательность натуральных чисел. Найти сумму четных чисел, наименьшее из четных чисел, и номер этого числа в последовательности.
8	Дана произвольной длины последовательность натуральных чисел. Найти произведение четных чисел, наименьшее из четных чисел и номер этого числа в последовательности.
9	Дана произвольной длины последовательность вещественных чисел. Найти сумму чисел, лежащих на отрезке $[1,10]$, наибольшее из таких чисел, и номер этого числа в последовательности.
10	Дана произвольной длины последовательность целых чисел. Найти произведение чисел, лежащих на отрезке $[-5,5]$, наименьшее из таких чисел, и номер этого числа в последовательности.
11	Дана произвольной длины последовательность натуральных

	чисел. Найти сумму чисел, заканчивающихся цифрой 0, наибольшее из таких чисел, и номер этого числа в последовательности.
12	Дана произвольной длины последовательность вещественных чисел. Найти произведение чисел с нулевой дробной частью, наименьшее из таких чисел, и номер этого числа в последовательности.
13	Дана произвольной длины последовательность целых чисел. Найти сумму неотрицательных чисел, наибольшее из таких чисел, и номер этого числа в последовательности.
14	Дана произвольной длины последовательность целых чисел. Найти произведение неположительных чисел, наименьшее из этих чисел, и номер этого числа в последовательности.
15	Дана произвольной длины последовательность натуральных чисел. Найти сумму чисел, не делящихся нацело на 2 и 11, наибольшее из таких чисел, и номер этого числа в последовательности.
16	Дана произвольной длины последовательность вещественных чисел. Найти произведение чисел с нечетной целой частью, наименьшее из таких чисел, и номер этого числа в последовательности.
17	Дана произвольной длины последовательность вещественных чисел. Найти сумму чисел с четной целой частью, наибольшее из таких чисел, и номер этого числа в последовательности.
18	Дана произвольной длины последовательность целых чисел. Найти произведение чисел, абсолютная величина которых не превосходит 10, наибольшее из таких чисел, и номер этого числа в последовательности.
19	Дана произвольной длины последовательность натуральных

	чисел. Найти произведение чисел, делящихся нацело на 3 и не делящихся нацело на 5, наибольшее из таких чисел, и номер этого числа в последовательности.
20	Дана произвольной длины последовательность вещественных чисел. Найти сумму чисел, целая часть которых делится нацело на 3, наименьшее из таких чисел, и номер этого числа в последовательности.
21	Дана произвольной длины последовательность целых чисел. Найти произведение чисел, заканчивающихся нечетной цифрой, наибольшее из таких чисел, и номер этого числа в последовательности.
22	Дана произвольной длины последовательность вещественных чисел. Найти сумму чисел с ненулевой дробной частью, наименьшее из таких чисел, и номер этого числа в последовательности.
23	Дана произвольной длины последовательность целых чисел. Найти сумму чисел, абсолютная величина которых больше 3, наибольшее из таких чисел, и номер этого числа в последовательности.
24	Дана произвольной длины последовательность целых чисел. Найти сумму чисел, не делящихся нацело на 3 и на 5, наименьшее из таких чисел, и номер этого числа в последовательности.
25	Дана произвольной длины последовательность натуральных чисел. Найти произведение чисел, не делящихся нацело на 6, наименьшее из таких чисел, и номер этого числа в последовательности.
26	Дана произвольной длины последовательность целых чисел. Найти сумму чисел, заканчивающихся цифрой 4, наибольшее из таких чисел, и номер этого числа в последовательности.

27	Дана произвольной длины последовательность натуральных чисел. Найти произведение чисел, не заканчивающихся цифрами 1 и 2, наименьшее из таких чисел, и номер этого числа в последовательности.
28	Дана произвольной длины последовательность вещественных чисел. Найти сумму отрицательных чисел, наибольшее из отрицательных чисел, и номер этого числа в последовательности.
29	Дана произвольной длины последовательность целых чисел. Найти произведение положительных чисел, не превосходящих 20, наибольшее из таких чисел, и номер этого числа в последовательности.
30	Дана произвольной длины последовательность натуральных чисел. Найти сумму нечетных чисел, наибольшее из нечетных чисел, и номер этого числа в последовательности.

Состав отчета

1. Номер, название и цель работы.
2. Текст задания.
3. Алгоритм решения задания.
4. Листинг (текст) программы.
5. Результаты работы программы.

Контрольные вопросы

1. Поясните назначение операторов **Do...Loop** и **For...Next**.
2. Перечислите элементы синтаксиса оператора цикла **For...Next** и поясните их назначение.
3. Характеризуйте две формы оператора цикла **Do...Loop**.
4. Покажите, каким образом на блок-схеме отображаются циклы типа **For...Next** и **Do...Loop**.

ЛАБОРАТОРНАЯ РАБОТА № 5. МАССИВЫ

Цель работы

Освоить методику хранения данных программ в массивах и научиться использовать массивы при решении практических задач.

Теоретические сведения

В программах для хранения данных одного типа помимо переменных могут использоваться специальные языковые конструкции – массивы. Синтаксис объявления массива (одномерного) следующий:

Dim <идентификатор> (размерность) As <тип>

Объявление и использование массивов похоже на использование переменных. Также как и переменная, массив имеет имя (<идентификатор>), но для обращения к данным в массиве требуется помимо идентификатора указать номер (индекс), под которым данные находятся в массиве (как его элементы). Количество номеров (индексов) элементов в массиве определяется при его объявлении размерностью (<размерность>). Адресация элементов массива определяется от 0 до размерности, т.е. кол-во элементов данных больше размерности на 1. Синтаксис записи данных в элементы массива и чтения данных из них представляется в следующем виде:

<идентификатор>(индекс)=<значение>

<значение>=<идентификатор>(индекс)

Пример использования массива:

Sub primer()

Dim A(1) As Integer

' Объявление (создание) массива по имени A из 2-х элементов

' для хранения целых чисел

A(0)=23 ' Запись в первый элемент массива числа

A(1)=35 ' Запись во второй элемент массива числа

MsgBox A(0)+A(1)

```
' Считывание значений из двух элементов массива,
' нахождение суммы этих значений и его отображение
End Sub
```



Массивы удобно использовать в сочетании с циклами для выполнения повторяющихся операций над данными:

Sub primer()**Dim A(2) As Double**

```
' Объявление массива A для хранения трех вещественных чисел
```

```
Dim i As Integer ' Объявление переменной-счетчика i
```

```
For i=0 To 2 Step 1
```

```
' Задания цикла, выполняемого от значения
```

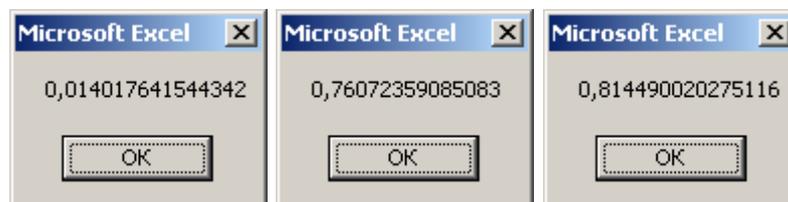
```
' переменной-счетчика i=0 до 2 при шаге ее изменения равном 1
```

```
A(i)=Rnd() ' Запись в i-ый элемент массива случайного числа
```

```
MsgBox A(i) ' Отображение значения из i-ого элемента массива
```

```
Next i ' Переход к следующему значению переменной-счетчика
```

```
End Sub
```



Для массивов возможно задание размерности в ходе работы программы по следующему синтаксису:

```
Dim <идентификатор>() As <тип>
```

```
...(программный код)...
```

```
ReDim <идентификатор>(<размерность>)
```

Пример переопределения размерности массива:

Sub primer()

Dim A() As Double

' Создание массива неизвестного до запуска программы размера

Dim i As Integer ' Создание переменной для хранения целых чисел

Dim j As Integer ' Создание переменной для хранения целых чисел

i = InputBox("Размерность массива (ЦЕЛОЕ ЧИСЛО >0):")

' Ввод в переменную i желаемой размерности массива

ReDim A(i - 1) ' Задание размерности массива из переменной i

For j = 0 To i - 1 Step 1

' Задания цикла, выполняемого от значения

' переменной-счетчика j=0 до i-1 при шаге ее изменения равном 1

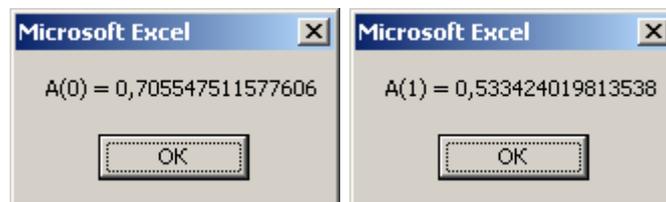
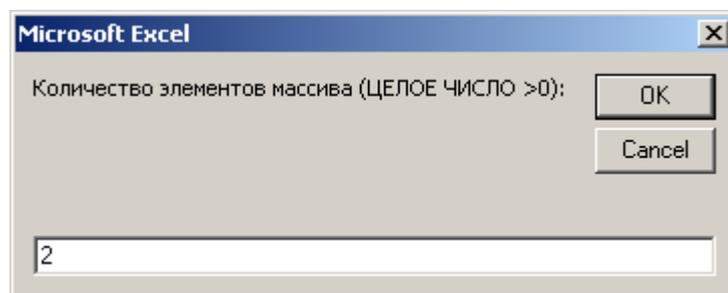
A(j) = Rnd() ' Запись массива случайного числа в j-й элемент

MsgBox "A(" & j & ") = " & A(j)

' Отображение значения j-ого элемента массива

Next j ' Переход к следующему значению переменной-счетчика

End Sub



Помимо одномерных используются многомерные массивы, например объявление двумерного массива имеет следующий синтаксис:

Dim <идентификатор> (размерность 1, размерность 2) As <тип>

Пример использования двумерного массива:

Sub primer6()

Dim A(2, 2) As Integer ' Объявление двумерного массива 3×3

```

Dim i As Integer, j As Integer ' Объявление переменных циклов
For i = 0 To 2 Step 1 ' Задание внешнего цикла по переменной i
For j = 0 To 2 Step 1 ' Задание вложенного цикла по j
A(i, j) = (i + 1) * (j + 1)
' Запись вычисленного значения в элемент двумерного массива
Next j
' Переход к следующему значению переменной вложенного цикла
Next i
' Переход к следующему значению переменной внешнего цикла
End Sub

```

Программа заполняет двухмерный массив по следующей схеме:

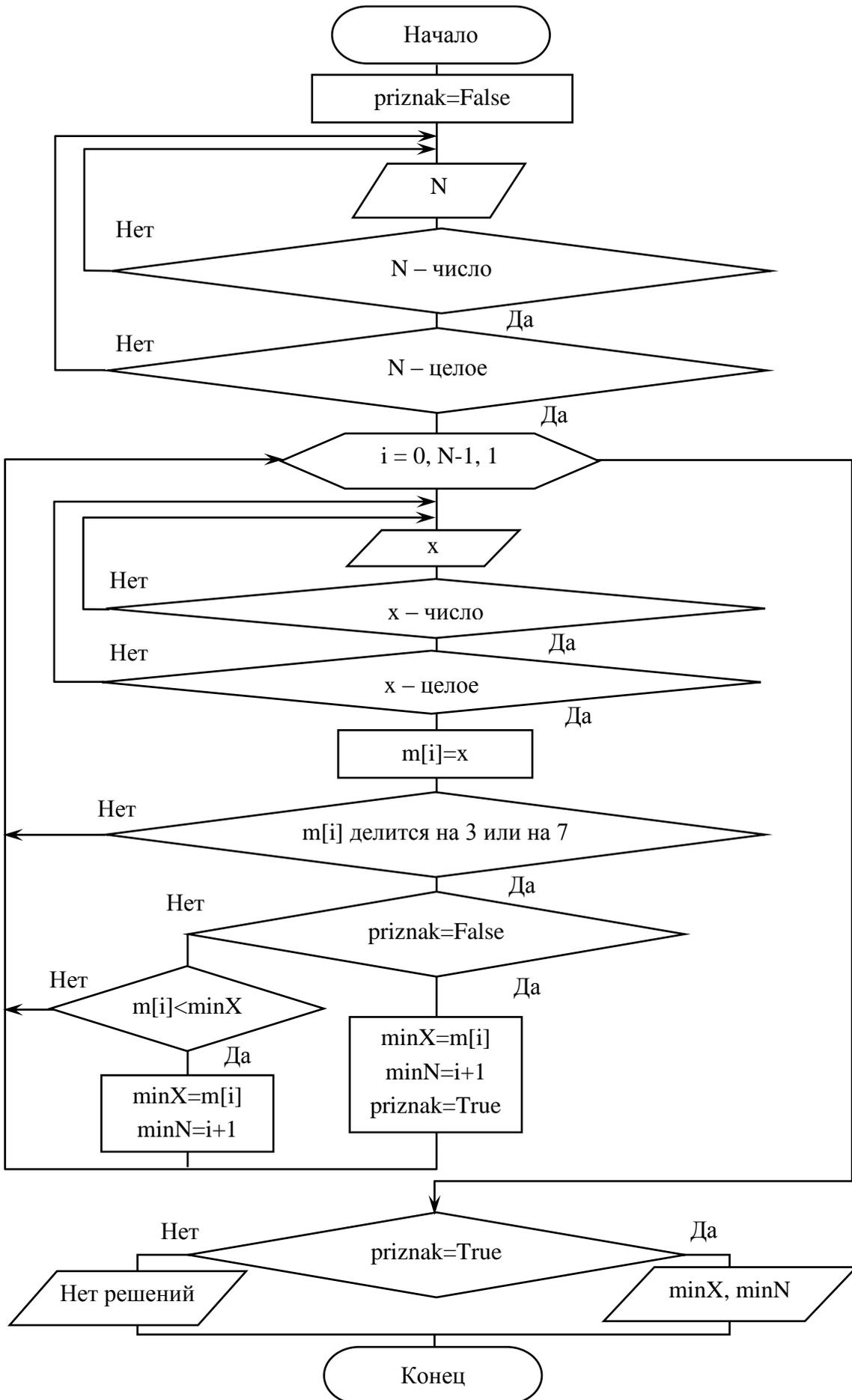
A(i,j)	j	0	1	2
i				
0		1	2	3
1		2	4	6
2		3	6	9

Пример

Дана произвольной длины последовательность целых чисел. Составить алгоритм и написать программу для нахождения наименьшего из чисел последовательности, делящихся нацело на 3 или на 7, и номера этого числа в последовательности. Количество чисел последовательности и сами числа задаются пользователем после запуска программы. Предусмотреть контроль значений, вводимых пользователем. Числа, вводимые пользователем, хранить в массиве.

Решение

Алгоритм (блок-схема) решения задачи:



Программа:

```
Sub primer8()  
Dim N As Variant  
Dim i As Integer  
Dim x As Variant  
Dim признак As Boolean  
Dim minX As Integer  
Dim minN As Integer  
Dim m() As Integer  
признак = False  
Do  
Do  
N = InputBox("Введите кол-во чисел в последовательности")  
Loop Until IsNumeric(N)  
Loop Until N = CInt(N)  
ReDim m(N - 1)  
For i = 0 To N - 1 Step 1  
Do  
Do  
x = InputBox("Введите целое число № " & i + 1)  
Loop Until IsNumeric(x)  
Loop Until x = CInt(x)  
m(i) = x  
If m(i) Mod 3 = 0 Or m(i) Mod 7 = 0 Then  
If признак = False Then  
minX = m(i)  
minN = i + 1  
признак = True  
Elseif x < minX Then
```

minX = m(i)

minN = i + 1

End If

End If

Next i

If признак = True Then

**MsgBox "Наименьшее число, делящееся на 3 или на 7, = " _
& minX & Chr(13) & "его № = " & minN**

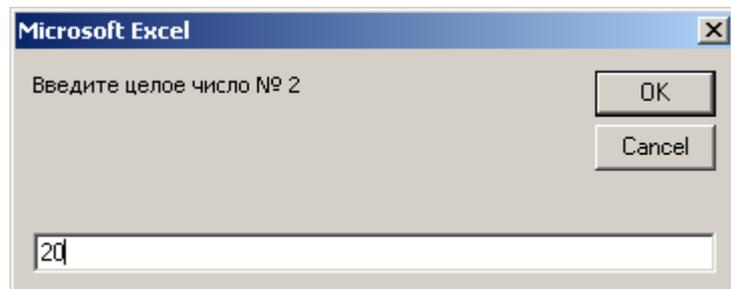
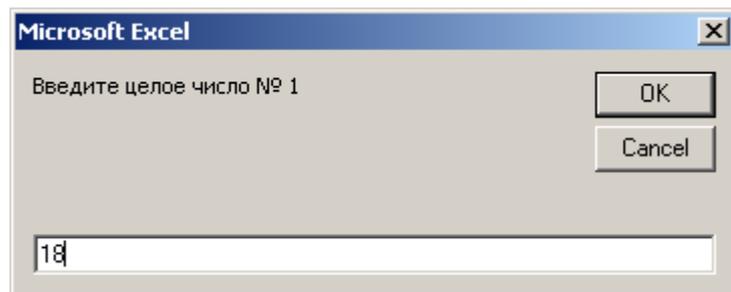
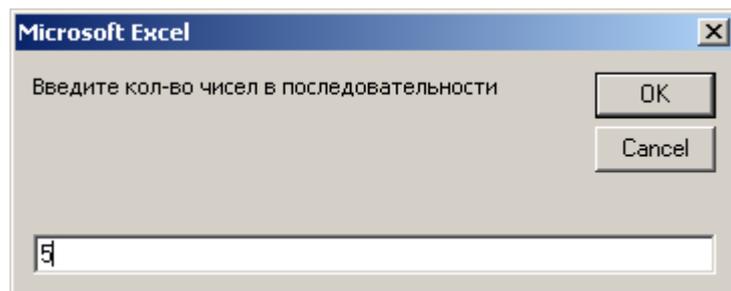
Else

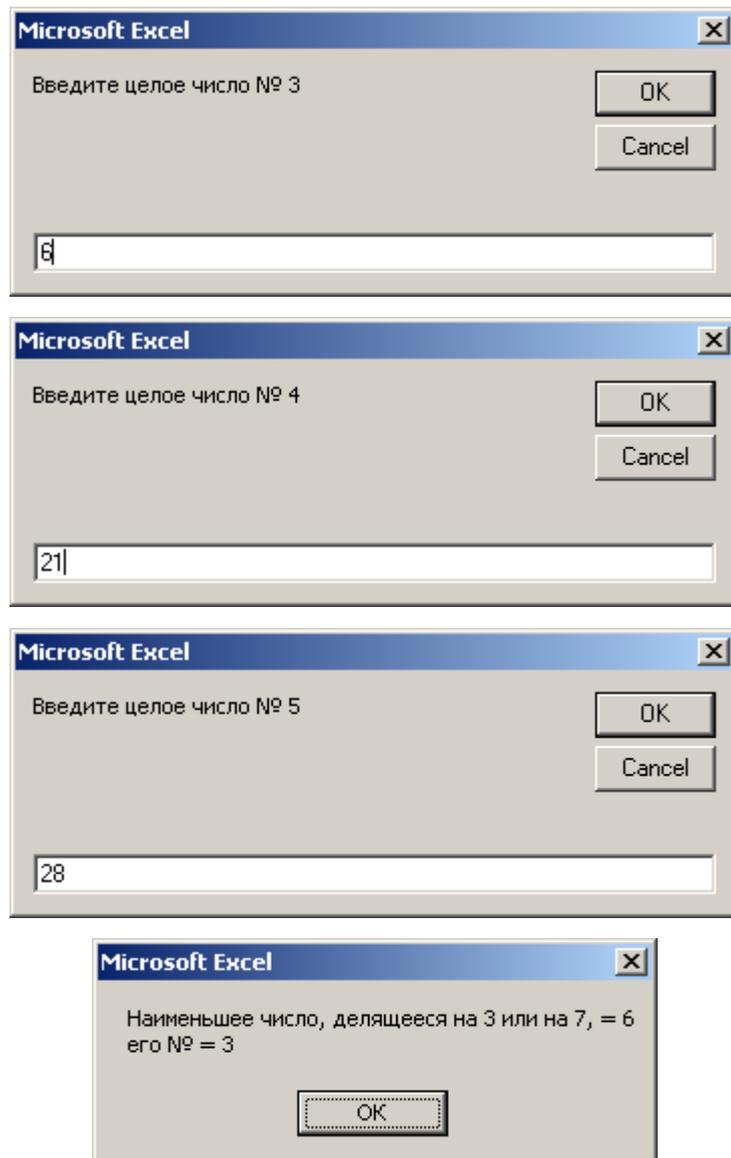
MsgBox "Нужных чисел нет"

End If

End Sub

Результаты работы программы:





Задание

Использовать задание из лабораторной работы № 4 "Операторы циклов", но числа, вводимые пользователем хранить в массиве.

Состав отчета

1. Номер, название и цель работы.
2. Текст задания.
3. Алгоритм решения задания.
4. Листинг (текст) программы.
5. Результаты работы программы.

Контрольные вопросы

1. Поясните назначение операторов **Do...Loop** и **For...Next**.
2. Перечислите элементы синтаксиса оператора цикла **For...Next** и поясните их назначение.
3. Характеризуйте две формы оператора цикла **Do...Loop**.
4. Покажите, каким образом на блок-схеме отображаются циклы типа **For...Next** и **Do...Loop**.

ЛАБОРАТОРНАЯ РАБОТА № 6. ПРОЦЕДУРЫ И ФУНКЦИИ

Цель работы

Научиться создавать и применять функции и процедуры, освоить методы передачи параметров.

Теоретические сведения

Процедуры и функции

Процедуры и функции являются основными структурными элементами программных модулей проектов VB и выполняют действия, предусмотренные разработчиком в соответствии с назначением программы.

Процедура представляет собой последовательность инструкций VB, которая выполняет действия, но не возвращает значение. Процедура может получать аргументы, как например константы, переменные, или выражения, передаваемые ей вызывающей процедурой.

Функция представляет собой последовательность инструкций VB, однако в отличие от процедуры она возвращает значения. Возврат значения осуществляется путем его присвоения имени функции в одной или нескольких инструкциях процедуры.

Отдельный модуль VB может содержать произвольное количество процедур и функций.

Определения процедур записываются в следующем виде (квадратные скобки означают возможность опускания элемента описания):

[Private | Public] [Static] Sub <Имя> ([Список аргументов])

[Инструкции]

[Exit Sub]

[Инструкции]

End Sub

Определения функций записываются в следующем виде:

[Public | Private] [Static] Function <Имя> ([Список аргументов]) [As <Тип>]

[Инструкции]

[<Имя> = выражение]

[Exit Function]

[Инструкции]

[<имя> = выражение]

End Function

Описание функции от описания процедуры отличается только ключевым словом (**Function** вместо **Sub**), указанием типа возвращаемого значения (**[As <тип>]**) и оператором возврата значения **[<имя> = выражение]**.

Рассмотрим назначение элементов описания функций и процедур (табл. 12).

Таблица 12

Элементы синтаксиса функций и процедур

Элемент	Описание
Public	Указывает, что процедура (функция (ф)) доступна для всех других процедур (ф) во всех модулях. Используется по умолчанию
Private	Указывает, что процедура (ф) доступна для других процедур (ф) только того модуля, в котором она описана
Static	Необязательный. Указывает, что переменные процедуры (ф) сохраняются в промежутках времени между вызовами этой процедуры. Атрибут Static не действует на переменные, описанные вне процедуры (ф), даже если они используются в этой процедуре
Имя	Имя процедуры (ф)
Список аргументов	Список переменных, представляющий аргументы, которые передаются в процедуру (ф) при ее вызове. Имена переменных разделяются запятой

Инструкции	Любая группа инструкций (операторов), выполняемых внутри процедуры (ф).
Exit	Инструкция, прекращающая выполнение процедуры (ф).

Аргумент **[Список аргументов]** имеет следующий синтаксис и элементы (табл. 13):

[Optional] [ByVal | ByVal] [ParamArray] <Имя переменной> [As <тип>] [= По умолчанию]

Таблица 13

Элементы синтаксиса списка аргументов процедур и функций

Элемент	Описание
Optional	Указывает, что этот аргумент необязателен
ByVal	Указывает, что этот аргумент передается по значению, то есть передается только копия значения переменной, служащей в качестве аргумента. Это означает, что переменная не может быть изменена в использующей ее процедуре (ф)
ByRef	Указывает, что этот аргумент передается по ссылке, то есть передается адрес расположения переменной, служащей в качестве аргумента. Это означает, что переменная может быть изменена в использующей ее процедуре (ф). Описание ByVal используется в VB по умолчанию
ParamArray	Используется только в качестве последнего элемента в списке [Список аргументов] для указания, что конечным аргументом является описанный как Optional массив типа Variant. Ключевое слово ParamArray позволяет задавать произвольное количество аргументов. Оно не может быть использовано с ключами ByVal, ByRef или Optional
Имя	Имя переменной

переменной	
Тип	Тип данных переменной
По умолчанию	Любая константа или выражение, дающее константу. Используется только вместе с параметром Optional для передачи значения по умолчанию при опускании параметра при вызове процедуры (ф)

Чтобы вызвать процедуру **Sub** из другой процедуры, следует указать имя этой процедуры и значения для всех требуемых аргументов через запятую в скобках или без них.

Sub proc1() ' Объявление вызывающей процедуры proc1

proc2 "Привет!"

' Вызов процедуры по имени proc2 с передачей ей

' параметра – строки "Привет!"

End Sub ' Завершение объявления процедуры proc1

Sub proc2(par As String)

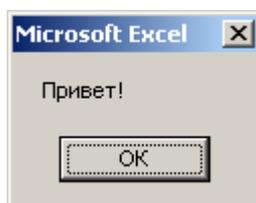
' Объявление вызываемой процедуры proc2, принимающей

' один параметр типа String (строка)

MsgBox par

' Отображение значения параметра, переданного в процедуру

End Sub ' Завершение объявления процедуры proc2



При использовании инструкции **Call** перед именем процедуры аргументы должны быть заключены в скобки. Для предыдущего примера:

Call proc2("Привет!")

Чтобы получить возвращаемое значение функции, надо указать ее имя, после которого в скобках ввести параметры, заданные в соответствии с определением функции.

Sub proc3() **Объявление вызывающей процедуры proc3**

MsgBox kvadrat(2.3)

' **Отображение значения, возвращаемого вызываемой функцией**

' **kvadrat, в которую передается число 2.3**

End Sub ' **Завершение объявления процедуры proc1**

Function kvadrat(par As Double) As Double

' **Объявление функции по имени kvadrat, возвращающей**

' **вещественное число и принимающей один**

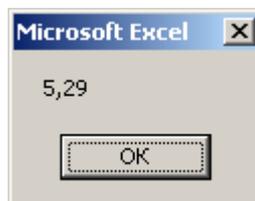
' **вещественный параметр**

kvadrat = par ^ 2

' **Запись рассчитанного значения в функцию kvadrat**

' **для возврата его в месте вызова функции (kvadrat)**

End Function ' **Завершение объявления функции**



Если возвращаемое значение функции не требуется, можно вызвать функцию точно так же, как процедуру **Sub**. Надо указать имя функции, опустить скобки и указать список аргументов. Для предыдущего примера:

kvadrat 2.3

Если процедура (ф) не имеет аргументов, инструкция объявления **Sub** или **Function** должна содержать пустые скобки.

При вызове процедур (ф) предполагается позиционная передача аргументов, т.е. в порядке следования в описании. Возможна также передача аргументов по именам, вне зависимости от их позиции в описании процедуры (ф).

Sub mycall()

```
msgbox "qwerty", 1234
```

```
' Вызов процедуры с передачей параметров по порядку
msgbox mykey:=1234, mystr:="qwerty"
```

```
' Вызов процедуры с передачей параметров по их именам
End sub
```

```
Sub msgbox(mystr As String, mykey As Integer)
```

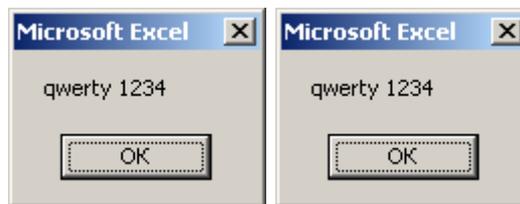
```
' Объявление процедуры, принимающей параметр по имени
```

```
' mystr типа String (строка) и параметр по имени mykey
```

```
' типа Integer (целое число)
```

```
MsgBox mystr & " " & mykey
```

```
End Sub
```



Проверить отсутствие необязательного параметра (отмеченного ключом **Optional**) можно с помощью функции **IsMissing**, возвращающей значение **True (Истина)**, если опускаемый аргумент отсутствует.

```
Sub CallDemo() ' Вызывающая процедура
```

```
DemoArg 10, 5
```

```
' Вызов процедуры DemoArg с передачей двух параметров
```

```
DemoArg 12
```

```
' Вызов процедуры DemoArg с передачей одного параметра
```

```
End Sub
```

```
Sub DemoArg(I As Integer, Optional J)
```

```
' Вызываемая процедура
```

```
If IsMissing(J) Then J = 10
```

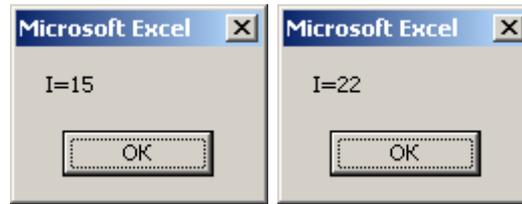
```
' При отсутствии параметра J в вызове процедуры DemoArg
```

```
' параметру J присваивается 10
```

```
I = I + J
```

MsgBox "I=" & I

End Sub



Не допускается определение процедуры (ф) внутри другой процедуры (ф).

В случае, если процедура (ф) должна быть доступна из процедур (ф), находящихся в других модулях перед ключевым словом ее объявления **Sub** или **Function** ставится инструкция **Public**:

Модуль Module1

Sub proc1() ' Описание процедуры proc1 в модуле Module1

proc2 "Ура !"

' Вызов процедуры proc2, описанной в другом модуле

End Sub

Модуль Module2

Public Sub proc2(p1 As String)

' Описание процедуры proc2 в модуле Module2

MsgBox UCase(p1)

End Sub



Область видимости переменных

При описании переменных обычно используется инструкция **Dim**. Для создания переменной, доступной на уровне процедуры (ф), инструкция описания помещается внутри процедуры (ф). Чтобы создать переменную доступную всем процедурам (ф) на уровне модуля, инструкция описания располагается в начале модуля, в разделе описаний.

В следующем примере создается переменная **strName** для хранения строковых данных (**String**).

Dim strName As String

Когда эта инструкция располагается в процедуре, переменная **strName** может использоваться только в данной процедуре. Если же такая инструкция находится в разделе описаний модуля, то переменная **strName** доступна для всех процедур данного модуля, но не может использоваться процедурами из других модулей проекта. В этом случае, чтобы сделать переменную, доступной для всех процедур проекта, перед ней надо поставить инструкцию **Public**.

Public strName As String

В языке VB можно неявно описать переменную, просто используя ее в инструкции присвоения. Все неявно описанные переменные имеют тип **Variant**. Переменные типа **Variant** более требовательны к ресурсам памяти, чем большинство других переменных. Явное описание всех переменных уменьшает вероятность конфликтов имен и ошибок, связанных с опечатками. Если неявные описания нежелательны, то всем процедурам в модуле должна предшествовать инструкция **Option Explicit**. Если модуль содержит инструкцию **Option Explicit**, при попытке использования неопisanного или неверно введенного имени переменной возникает ошибка во время компиляции.

Пример использования видимости переменных:

Модуль Module1:

Option Explicit ' Требование объявления переменных

Public perem1 As Integer

Sub proc1()

Dim perem2 As Integer

perem1 = 12

perem2 = 5

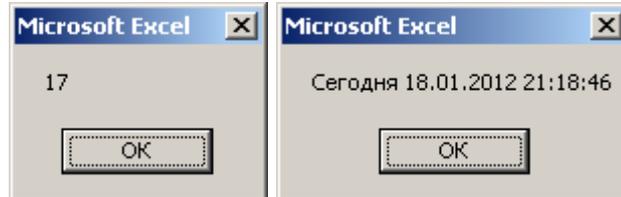
MsgBox perem1 + perem2

proc2 ' Вызов процедуры proc2 из модуля Module2**End Sub**

Модуль Module2:

Option Explicit**Dim str1 As String****Public Sub proc2()****str1 = "Сегодня "****MsgBox func1()****End Sub****Function func1() As String****Dim str2 As Variant****str2=Now()+perem1****func1 = str1 & str2****End Function**

Запуск процедуры proc1 из модуля Module1:



Видимость (доступность), используемых в примере переменных:

Процедура (функция) Переменная	proc1 (Module1)	proc2 (Module2)	func1 (Module2)
perem1	+	+	+
perem2	+	-	-
str1	-	+	+
str2	-	-	+

Пример

Для условий лабораторной работы № 3 "Условные операторы":

Задание А:

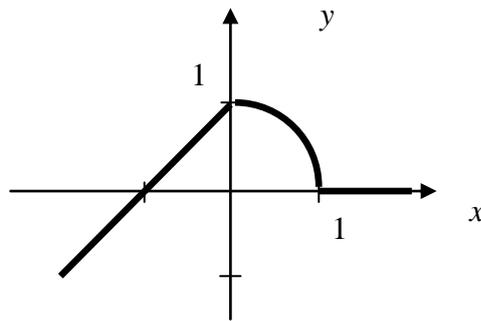
Составить алгоритм и нарисовать блок-схему для вычисления функции

$$y = \sqrt{\frac{2-x}{\ln x}}$$

(выражение имеет ограничения в области определения). Составить программу вычисления функции при различных значениях аргумента (x).

Задание Б:

Представить аналитическими выражениями функцию $y = f(x)$, заданную графически.



Изобразить блок-схему и написать программу для расчета $y = f(x)$ при различных значениях x.

... выполнить следующие требования:

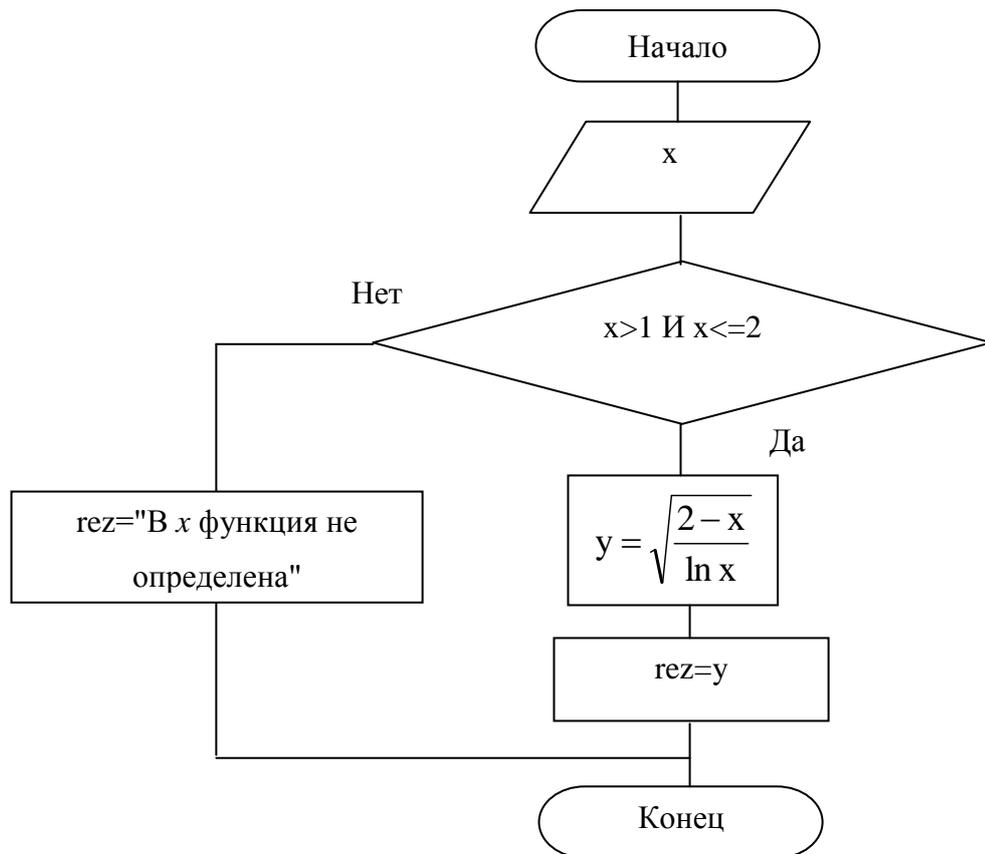
- 1) Задание А оформить как процедуру VB, в которую передается для расчета значение аргумента функции.
- 2) Задание Б оформить как функцию VB, которая возвращает значение $y = f(x)$ и в которую передается для расчета значение аргумента (x).
- 3) Создать глобальную переменную, в которую записывать результаты работы процедуры на основе Задания А (требование 1) и функции на основе Задания Б (требование 2).
- 4) Поместить процедуру (требование 1), функцию (требование 2) и глобальную переменную (требование 3) в отдельный модуль VB

5) В отдельном модуле VB (еще одном) создать процедуру, в которой выполняется вызов процедуры на основе задания А (требование 1), функции на основе Задания Б (требование 2) и глобальной переменной с результатами работы (требование 3)

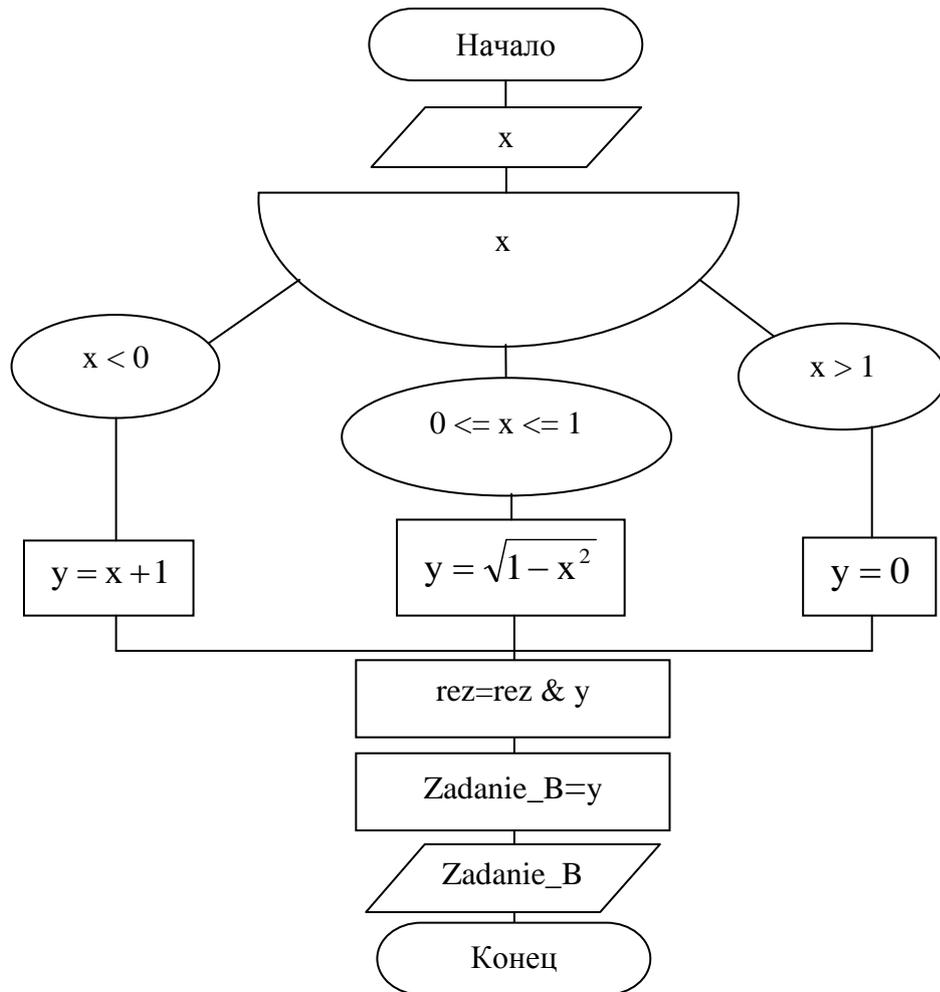
Решение

Алгоритм (блок-схема) решения задачи:

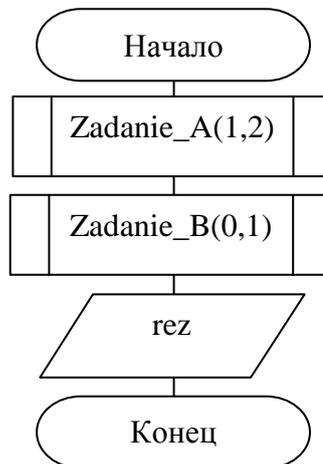
Процедура Zадание_A



Функция Zадание_B



Процедура вызова Zadanie_A, Zadanie_B, rez



Программа:

Модуль 1:

Option Explicit

' **Требование обязательного объявления переменных**

Public rez As String

' **Объявление глобальной (доступной во всех модулях)**

' переменной по имени rez для хранения результатов

Public Sub Zadanie_A(x As Double)

' Объявление глобальной (доступной во всех модулях)

' процедуры по имени Zadanie_A, для работы которой

' требуется один входной параметр, хранимый в переменной x

Dim y As Double

' Объявление локальной (доступной только в текущей процедуре)

' переменной по имени y для хранения вещественных чисел

If (x > 1 And x <= 2) Then ' Проверка условия для расчета

y = Sqr((2 - x) / Log(x))

' Запись результата расчета в локальную переменную

rez = "Задание A: В x = " & x & " y = " & y

' Запись в глобальную переменную результатов расчета

Else

rez = "Задание A: В x = " & x & " функции нет"

' Запись в глобальную переменную результатов расчета

End If

End Sub ' Завершение объявления процедуры Zadanie_A

Public Function Zadanie_B(x As Double) As Double

' Объявление глобальной (доступной во всех модулях)

' функции по имени Zadanie_B, возвращающей

' значение типа вещественного числа. Для работы функции

' требуется один входной параметр, хранимый в переменной x

Dim y As Double

' Объявление локальной (доступной только в текущей функции)

' переменной по имени y для хранения вещественных чисел

Select Case x

' Оператор Select для выполнения расчета и записи результата

```

' в локальную переменную y
Case Is < 0
y = x + 1
Case Is <= 1
y = Sqr(1 - x * x)
Case Else
y = 0
End Select ' Завершение оператора Select
rez = rez & Chr(13) & "Задание Б: В x = " & x & " y = " & y
' Дописывание в глобальную переменную результатов расчета
Zadanie_B = y
' Запись из локальной переменной y в функцию Zadanie_B
значения для возврата его в месте вызова функции (Zadanie_B)
End Function ' Завершение объявления функции Zadanie_B

```

Модуль 2:

Option Explicit

```

' Требование обязательного объявления переменных
Sub glavnaJa() ' Объявление процедуры по имени glavnaJa
Zadanie_A 1.2
' Вызов (выполнение) процедуры Zadanie_A с переданным ей
' параметром равным 1.2
MsgBox Zadanie_B(0.1)
' Отображение значения, возвращаемого функцией Zadanie_B,
' после ее вызова с переданным параметром равным 0.1
MsgBox rez
' Отображение значения глобальной (доступной во всех
' модулях) переменной rez
End Sub

```

Результаты работы программы:

Вызов процедуры **glavnaja**:



Задание

Для условий лабораторной работы № 3 "Условные операторы" (Задание А и Задание Б) выполнить следующие требования:

- 1) Задание А оформить как процедуру VB, в которую передается для расчета значение аргумента функции.
- 2) Задание Б оформить как функцию VB, которая возвращает значение $y = f(x)$ и в которую передается для расчета значение аргумента (x).
- 3) Создать глобальную переменную, в которую записывать результаты работы процедуры на основе Задания А (требование 1) и функции на основе Задания Б (требование 2).
- 4) Поместить процедуру (требование 1), функцию (требование 2) и глобальную переменную (требование 3) в отдельный модуль VB
- 5) В отдельном модуле VB (еще одном) создать процедуру, в которой выполняется вызов процедуры на основе задания А (требование 1), функции на основе Задания Б (требование 2) и глобальной переменной с результатами работы (требование 3)

Состав отчета

1. Номер, название и цель работы.
2. Текст задания.
3. Алгоритм решения задания.
4. Листинг (текст) программы.
5. Результаты работы программы.

Контрольные вопросы

1. Охарактеризуйте понятие процедуры в языке VB.
2. Охарактеризуйте понятие функции в языке VB.
3. Приведите схему синтаксиса процедуры в языке VB.
4. Приведите схему синтаксиса функции в языке VB.
5. Приведите примеры вызова процедур и функций в языке VB. Как осуществляется вызов процедур и функций между модулями?
6. Какие варианты области видимости переменных предусмотрены в VB и как они задаются?

ЛАБОРАТОРНАЯ РАБОТА № 7. ФОРМЫ И ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Цель работы

Получить навыки в построении пользовательского интерфейса программы с использованием стандартных элементов управления.

Теоретические сведения

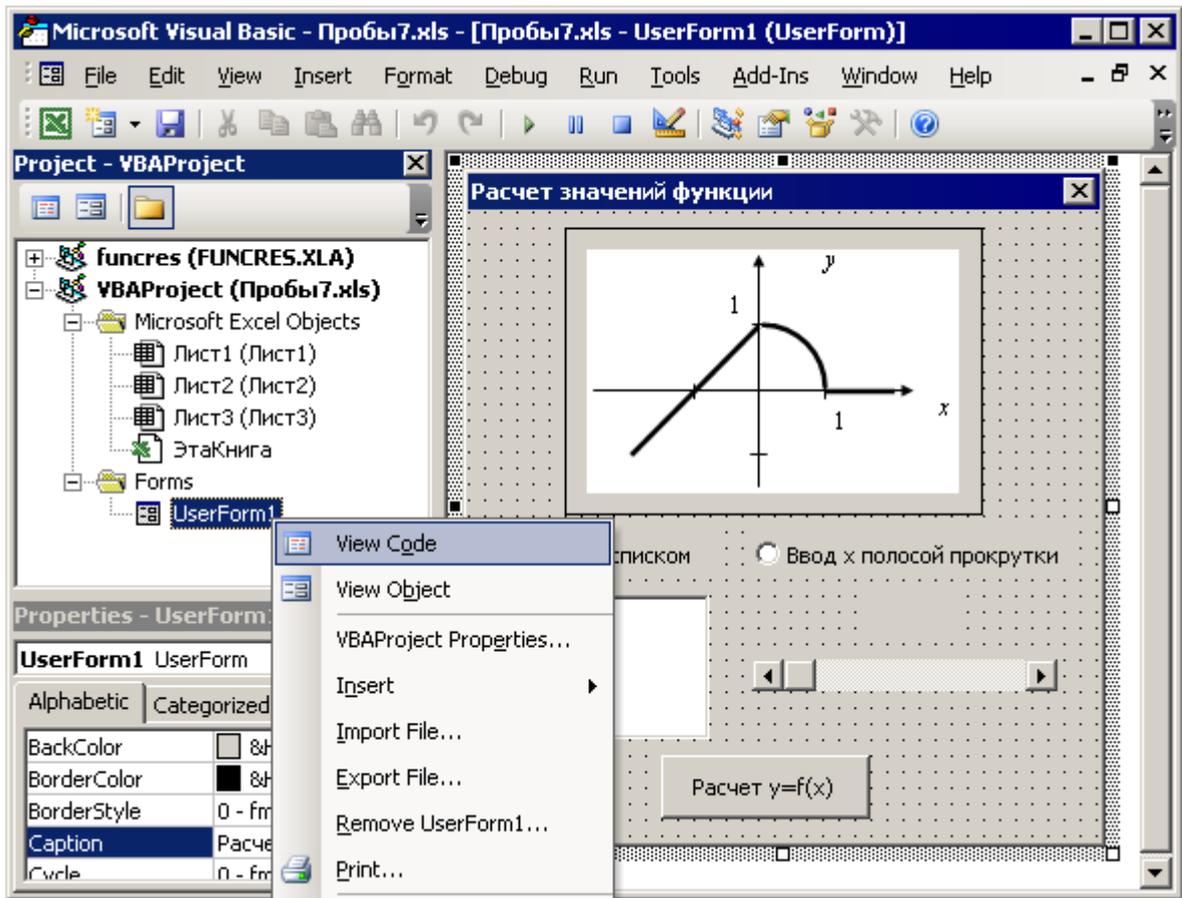
Формы: свойства, методы, обработка событий

Все объекты, которые пользователь видит на экране при работе программ, – визуальные компоненты. Внешний вид и функции визуальных компонентов настраиваются путем изменения свойств, методов и кода процедур обработки событий. Модули кода для взаимодействия с визуальными компонентами – программные компоненты. Основные визуальные компоненты программ – формы – контейнеры, содержащие различные элементы управления (кнопки, флажки, переключатели, списки и др.).

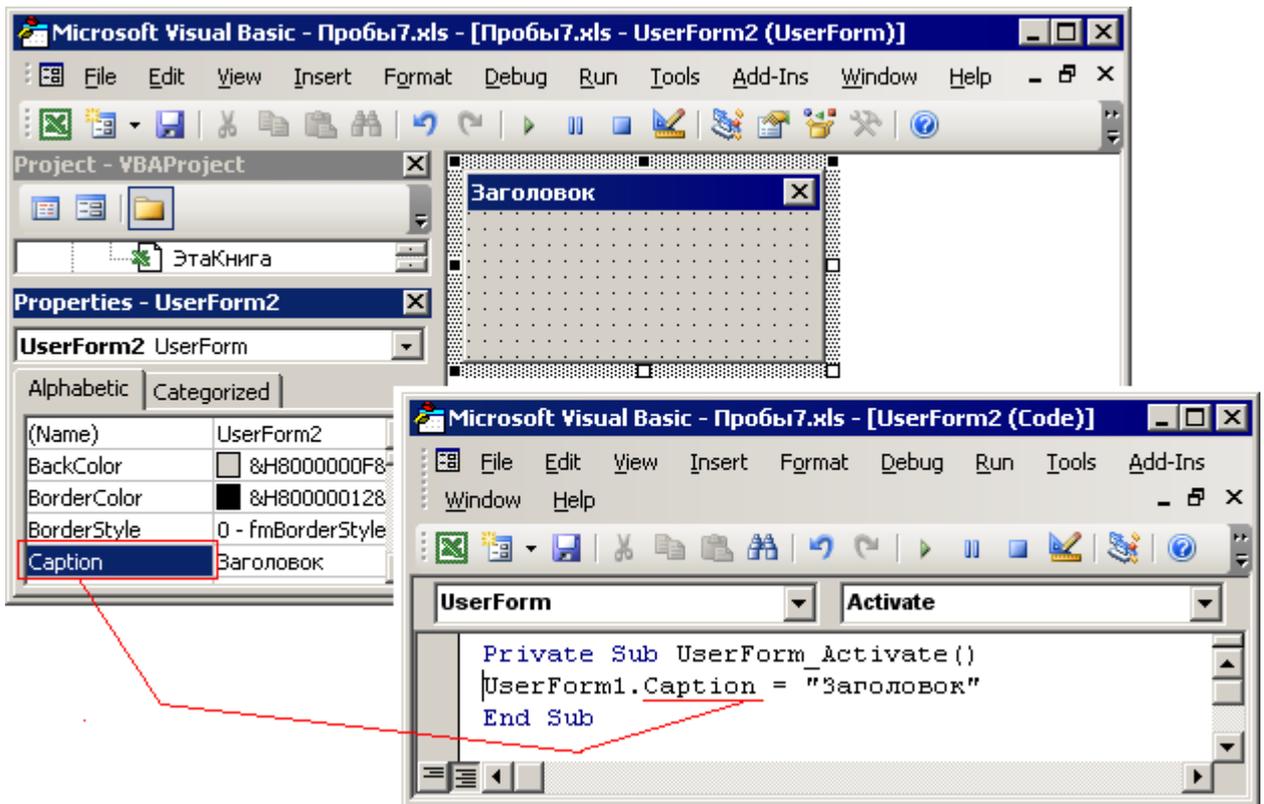
Добавление формы в проект VB производится командой оболочки VB: ***Insert (Вставка) > User Form (Пользовательская форма)***. Пустая форма состоит из: строки заголовка, кнопки закрытия окна формы, а также рабочей области.



Весь программный код, относящийся к объектам одной формы, хранится в одном программном модуле, относящимся к форме (этот код можно открывать на редактирование контекстной командой ***View Code (Посмотреть код)***).



Для каждого типа объектов VB предусмотрен свой набор свойств. Во время выполнения программы характеристики объектов изменяются путем изменения свойств. В режиме разработки при редактировании и форматировании объектов автоматически изменяются и значения свойств. Управлять значением свойств объектов можно с помощью окна *Properties* (*View > Properties Window*) или используя имена свойств в коде VB.



Некоторые свойства являются общими для большинства объектов: **Name**, **Left**, **Top**, **Height**, **Width**.

Положение формы на экране зависит от значения свойств **Left** (расстояние от левого края экрана в твипах) и **Top** (расстояние от верхнего края экрана), а размеры – от значения свойств **Width** и **Height**. Свойства **Left** и **Top** устанавливаются относительно объекта-контейнера (экрана для формы, формы для элемента управления). Для изменения положения объекта можно использовать методы или свойства.

MyObj.Left = 100

' Задание расстояния в 100 точек от верхнего левого угла

' объекта MyObj до левой границы объекта, содержащего MyObj

MyObj.Top = 200

' Задание расстояния в 200 точек от верхнего левого угла

' объекта MyObj до верхней границы объекта, содержащего MyObj

ИЛИ

MyObj.Move 100, 200

Текст, отображаемый в строке заголовка формы, является значением свойства **Caption**. Свойство **Font** позволяет выбрать тип, размер и стиль шрифта текста.

MyObj.Font.Size = 10

Свойство **StartPosition** влияет на начальное положение формы на экране, которое она занимает при запуске приложения.

Свойство **Name** служит для присвоения объекту уникального идентификатора, по которому к объекту можно обратиться из программы. Все формы проекта должны иметь разные имена. Однако имена элементов управления должны быть уникальны только в пределах той формы, в которой они находятся.

С помощью оператора **Load** форму можно загрузить в память, однако при этом на экране она не отобразится.

Load MyForm ' Загрузка в память формы по имени MyForm

Данный оператор позволяет загрузить форму (**MyForm**) в память явным образом. При обращении к свойству, методу или любому элементу управления формы последняя будет загружена в память автоматически, т.е. неявным образом. Для форм предусмотрено специальное событие **Form_Load**, обработчик которого будет вызван сразу после загрузки формы в память.

Для отображения формы на экране используется метод **Show**. Данный метод работает независимо от того, загружена форма в память или нет. В последнем случае при вызове метода **Show**, форма будет загружена неявным образом, после чего она появится на экране.

MyForm.Show 'Отображение (и загрузка) формы MyForm на экране

Убрать форму с экрана можно двумя способами. Один из них – воспользоваться методом **Hide**.

MyForm.Hide ' Скрытие формы MyForm

Данный метод просто удаляет форму с экрана, но не выгружает ее из памяти. Когда работа с формой закончена и информация, которая находится в ее

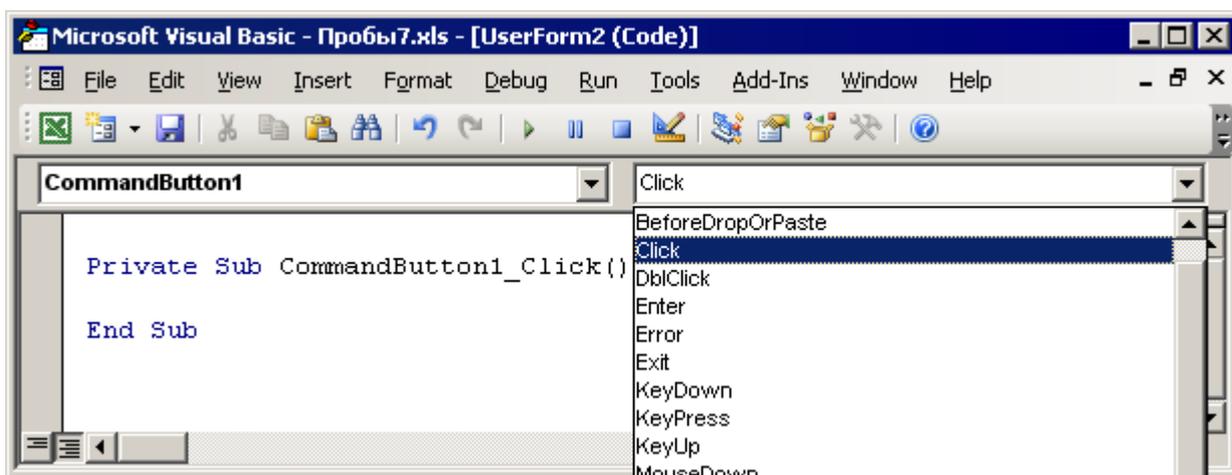
элементах управления, будет больше не нужна, форму можно убрать с экрана и удалить из памяти одной командой – **Unload**.

Unload MyForm ' Скрытие формы MyForm и удаление ее из памяти

Если **Unload** используется для выгрузки текущей формы, то вместо имени формы к нему следует добавить **Me**.

Объекты могут реагировать на события, возникающие в результате каких-либо действий пользователя или генерируемых операционной системой. Например, после щелчка мышью на кнопке возникает событие **Click**, поступающее от объекта типа **CommandButton** (кнопка).

Для реакции на определенное событие требуется написать специальный код и поместить его в нужную процедуру обработки события. В списке слева над окном кода выбирается объект, а в списке справа – событие для которого необходимо описать определенную последовательность действий. При этом автоматически создается соответствующая процедура.



Например, при отображении формы на экране возникает событие **Activate** (Активация) для данной формы и для данного события можно записать обработчик.

Private Sub MyForm_Activate()

' Описание процедуры, выполняемой при активации (Activate)

' формы MyForm

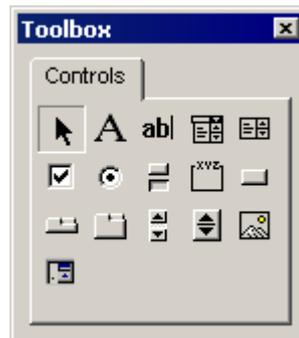
MyForm.BackColor = vbGreen

' Установка для формы MyForm фонового цвета (BackColor) –

' зеленый (vbGreen)

End Sub

Для обеспечения функциональных возможностей формы снабжаются элементами управления – объектами, выполняющими определенные функции. Как и у форм у элементов управления существует набор свойств, методов и событий. В поставку VB входит базовый набор элементов управления: *View (Вид) > ToolBox (Панель элементов)*.



Все элементы управления обладают двумя важными свойствами: **Visible** и **Enabled**. Свойство **Visible** определяет, будет ли объект виден (**True/False**) на экране, а от свойства **Enabled** зависит, сможет ли пользователь работать с данным элементом управления.

MyObj.Visible=False ' Объект MyObj

Далее коротко рассмотрим приемы работы с различными элементами управления.

Надпись (Label)

Основным назначением элементов управления типа **Label**  является отображение текста на форме с помощью свойства **Caption**.

Для **Label** предусмотрен определенный набор свойств (**Font**, **Alignment** (выравнивание), **ForeColor** и **BackColor** (цвет текста и фона), **Appearance** (стиль)) и событий (щелчки мыши, наведение указателя), позволяющий управлять элементом с помощью программного кода.

Максимальный размер текста в надписи не должен превышать 65528 символов. Когда свойство **AutoSize** равно **False**, размер поля не зависит от

значения свойства **Caption** и остается постоянным. Если размер надписи не достаточен для отображения всего текста, часть текста, выходящая за границы поля, не отображается. Если значение свойства **AutoSize** равно **True**, размеры надписи будут автоматически подстраиваться под длину текста.

Если значение **Wordwrap = False**, размеры надписи будут увеличиваться только по горизонтали, пока весь текст не будет виден на экране. Если свойство **Wordwrap = True**, размеры надписи будут увеличиваться по вертикали так, чтобы был виден весь текст. При этом будет происходить автоматический перенос слов на новую строку.

Пример программной работы с надписью:

```
Dim t1 As String ' Объявление строковой переменной t1
Label1.Caption="Пример"
' Запись текста "Пример" в надпись Label с помощью
' ее свойства Caption
t1=Label1.Caption
' Считывание текста из надписи Label1 в переменную t1
```

Поле ввода (TextBox)

TextBox  позволяет отредактировать помещенный в него текст или ввести новый. Для ограничения количества вводимых символов используется свойство **MaxLength**.

Свойство **MultiLine** определяет, будет ли текст находиться в одной строке, или он будет переноситься по словам на несколько строк. Если значение свойства **MultiLine** равно **True**, текст будет переноситься по словам на несколько строк. Чтобы перейти на новую строку при вводе текста нужно нажать *Shift+Enter*. Свойство **ScrollBar** определяет отображение полос прокрутки и их тип.

Для **TextBox** существуют такие события, как **Change** (при каждом действии редактирования в поле), **Enter/Exit** (при получении (потере) фокуса **TextBox**-м при нажатии **Tab** в форме).

Программное считывание данных из поля ввода производится с помощью свойства **Text**:

MsgBox TextBox1.Text

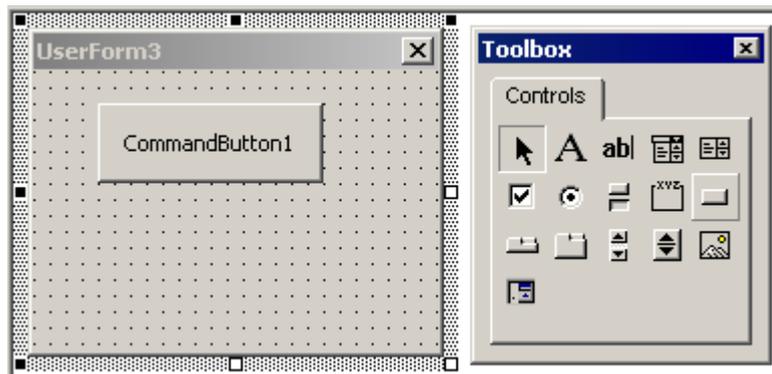
' Отображение текста, содержащегося (введенного)

' в поле ввода **TextBox1**

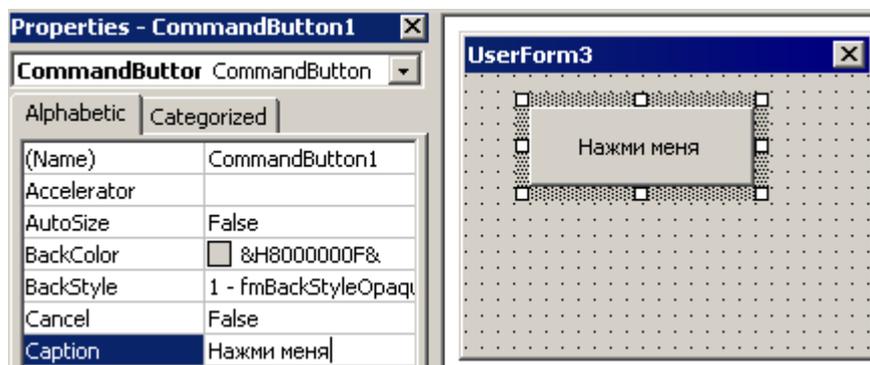
Кнопка (CommandButton)

CommandButton  предназначен для инициирования пользователем некоторых действий, которые начинают происходить после щелчка на нем кнопкой мыши. Порядок использования **CommandButton**:

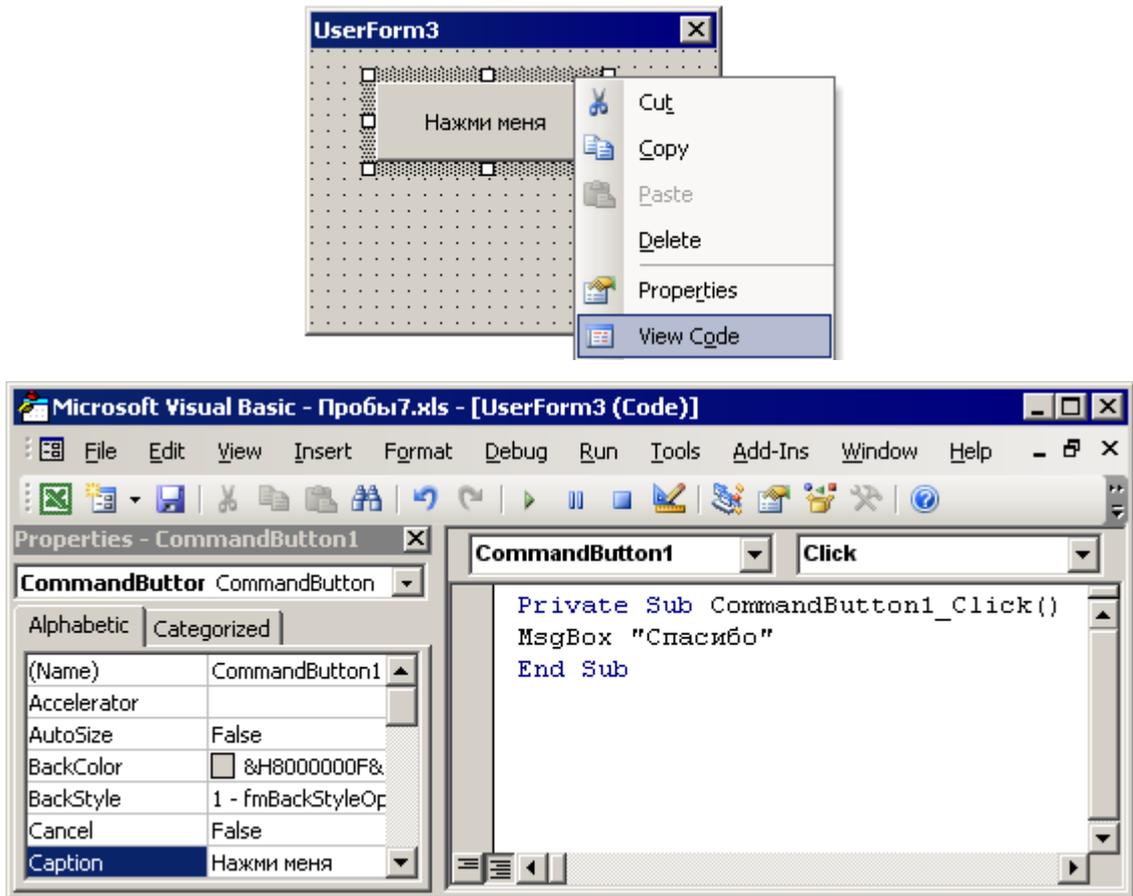
1) добавить **CommandButton** в форму из **Toolbox**,



2) присвоить свойству **Name** имя кнопки, а свойству **Caption** – текст (для подписи),



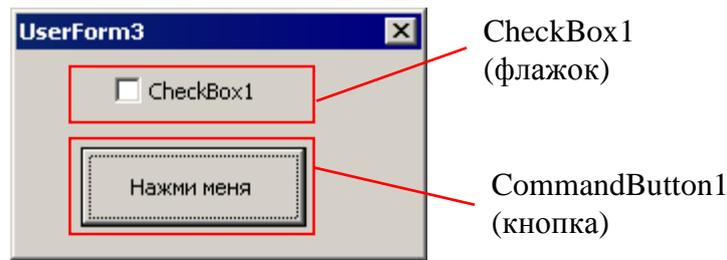
3) поместить программный код, выполняющий нужные действия, в процедуру обработки события **Click**, создать которую можно, выполнив на кнопке контекстную команду **View Code (Смотреть программный код)**.



В приведенном случае действия по нажатию (**Click-y**) на кнопку **CommandButton1** заключаются в выводе на экран сообщения со словом "Спасибо".

Флажок (CheckBox)

CheckBox используются для отображения в форме логических значений (**True** или **False**). Когда флажок находится "во включенном состоянии", на нем изображена галочка, в этом случае его значение будет равно **True (Истина)**. Считывание (и запись) значения из (в) **CheckBox** производится с помощью его свойства **Value**. На практике значение, которое принял **CheckBox**, анализируется в программе и в зависимости от результата анализа выполняются различные действия.



Private Sub CommandButton1_Click()

' Процедура обработки щелчка по кнопке

If CheckBox1.Value = True Then

' Проверка, установлен ли флажок CheckBox1

MsgBox "Флажок установлен"

' Действия (вывод сообщения), если флажок установлен

Else

MsgBox "Флажок НЕ установлен"

' Действия (вывод сообщения), если флажок не установлен

End If

End Sub

Переключатель (OptionButton)

Переключатели  или радио-кнопки всегда должны находиться в группах, причем только один из переключателей группы может быть "нажат" или активизирован. Переключатель служит для выбора одного параметра из группы. Свойство **Group (группа)** – имя группы радиокнопок, позволяет разместить в одной форме несколько групп переключателей, таким образом, что переключение кнопок в одной группе не влияет на значения кнопок в другой.

Для переключателей, флажков и кнопок генерируются и обрабатываются события: щелчки и наведения мыши, **Change, Enter, Exit** и др.

Использование **OptionButton** также как и **CheckBox** основано на чтении свойства **Value** (Значение) этого элемента управления. У выбранного пользователем в форме **OptionButton** значение **Value** равно **True**

(Истина), а у всех остальных одногруппных с ним **OptionButton** это свойство будет равно **False (Ложь)**.

Список (ListBox)

ListBox  используется для предоставления пользователю возможности выбора одного элемента из представленных в списке.

Основные элементы списка: Список элементов для выбора, Выбранный элемент, Полоса прокрутки. Формирование списка происходит на этапе разработки программы либо самой программой во время ее выполнения.

Для наполнения списка новыми элементами используют метод **AddItem** при загрузке формы или действию с элементом управления.

MyColorList.AddItem "зеленый"

' Добавление элемента в виде слова "зеленый" в список
' по имени MyColorList

Для очистки списка от его содержимого существует метод **Clear**.

К элементу списка можно обратиться с помощью свойства **List**, используя следующий синтаксис:

<имя ListBox>.List(<индекс>)

Нумерация индексов списка ведется с 0, т.е. второй элемент списка имеет индекс 1.

Свойство **ListCount** возвращает количество элементов списка; свойство **ListIndex** содержит номер выбранного элемента списка.

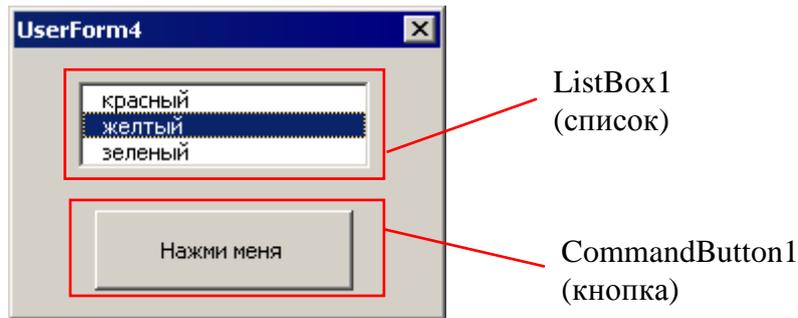
Значение текущего элемента списка можно получить, используя свойство **Value**.

Для удаления элемента из списка используют метод **RemoveItem**. В качестве параметра методу нужно передать индекс элемента списка.

Синтаксис удаления элемента списка следующий:

<имя ListBox>.RemoveItem(<индекс>)

Пример использования ListBox:



Private Sub UserForm_Activate()

' Описание процедуры, выполняющейся при активации (Activate)

' формы UserForm, содержащей список и кнопку

ListBox1.AddItem "красный"

' Добавление в элемент управления список ListBox1 первого

' элемента (с индексом 0) – слова "красный"

ListBox1.AddItem "желтый"

ListBox1.AddItem "зеленый"

End Sub

Private Sub CommandButton1_Click()

' Описание процедуры, выполняемой при щелчке (Click)

' на кнопке CommandButton1

MsgBox ListBox1.List(ListBox1.ListIndex)

' Отображение возвращаемого методом List выбранного

' элемента списка, по его индексу (ListBox1.ListIndex)

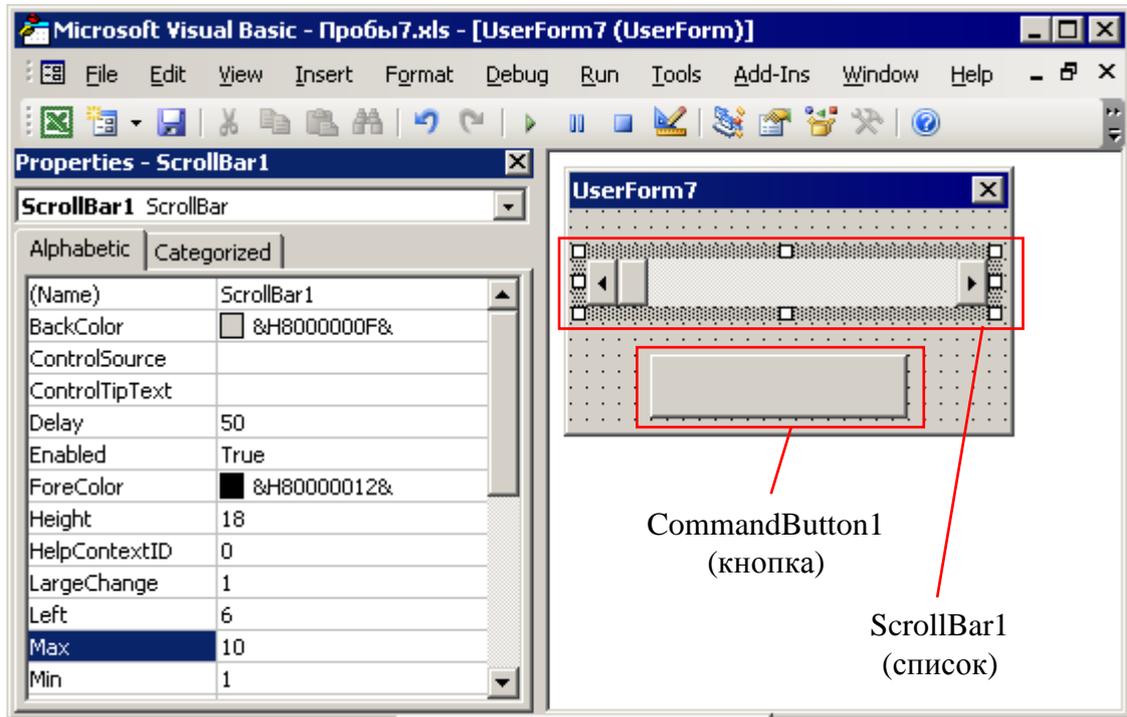
End Sub

Полоса прокрутки (ScrollBar)

Полосы прокрутки  используются для установки параметра, значение которого может меняться в некотором диапазоне – от минимума до максимума. **ScrollBar** можно использовать для отображения и ввода чисел.

Перед использованием **ScrollBar** необходимо определить (программно или с использованием окна **Properties**) диапазон вводимых чисел, установив значения свойств **Min** и **Max** (целые числа). Дальнейшая работа заключается в чтении и установке свойства **Value** объекта **ScrollBar**. При изменении

значения свойства **Value**, описываемого **ScrollBar**, автоматически возникает событие **Change**, обработку которого можно выполнять программно. Пример использования **ScrollBar**:



Private Sub CommandButton1_Click()

' Описание процедуры, выполняемой при щелчке (Click)

' на кнопке CommandButton1

MsgBox ScrollBar1.Value

' Отображение значения (Value) из полосы прокрутки (ScrollBar1)

End Sub

Private Sub ScrollBar1_Change()

' Описание процедуры, выполняемой при изменении (Change)

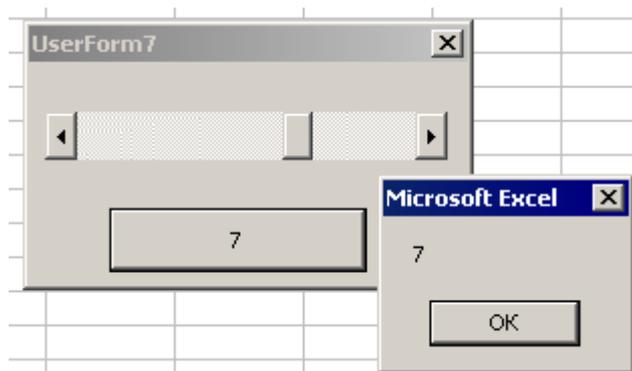
' значения полосы прокрутки ScrollBar1

CommandButton1.Caption = ScrollBar1.Value

' Запись в заголовок (Caption) кнопки CommandButton1

' значения (Value) из полосы прокрутки (ScrollBar1)

End Sub

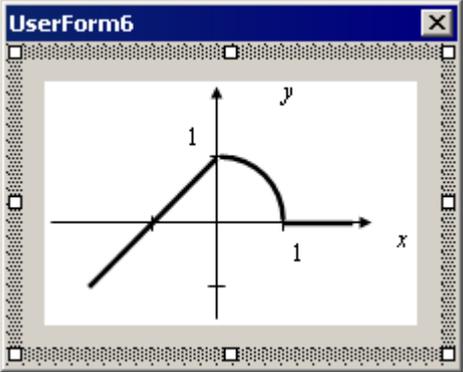


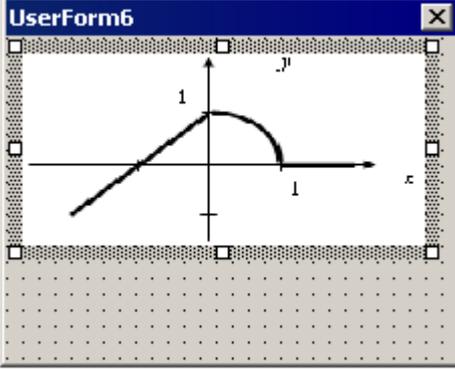
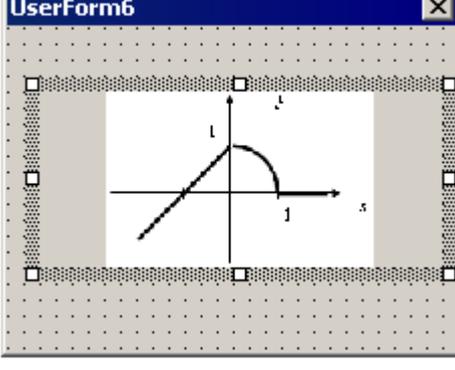
Картинка (Image)

С помощью элемента управления типа **Image**  можно помещать на форму графические файлы различных форматов (bmp, gif, jpeg, wmf и др.). Файл иллюстрации добавляется в элемент управления типа **Image** при задании свойства **Picture** (например, в окне **Properties**). Для элемента **Image** доступны свойства изменения положения на форме (**Left, Top**) и размера (**Width, Height**). При изменении размеров Image важную роль играет свойство **PictureSizeMode** (табл. 14).

Таблица 14

Влияние свойства PictureSizeMode на элемент типа Image

Значение свойства PictureSizeMode	Действие свойства	Описание свойства
fmPictureSizeModeClip		Файл изображения добавляется в элемент Image в своем оригинальном размере. Размеры Image не влияют на размер изображения, но при уменьшении могут часть его скрыть

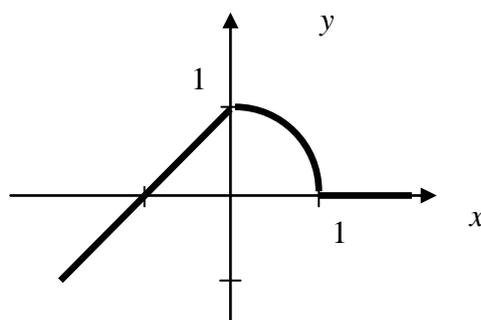
fmPictureSizeModeStretch		<p>Изображение в элементе Image изменяет свои размеры вслед за элементом как по ширине, так и по высоте (при этом возможно непропорциональное изменение размеров)</p>
fmPictureSizeModeZoom		<p>Изображение в элементе Image изменяет свои размеры вслед за элементом, сохраняя исходное отношение высоты к ширине (возможное в этом случае незаполненное пространство Image окрашивается фоновым цветом BackColor)</p>

Пример

Для Задания Б из лабораторной работы № 3 "Условные операторы":

Задание Б.

Представить аналитическими выражениями функцию $y = f(x)$, заданную графически.



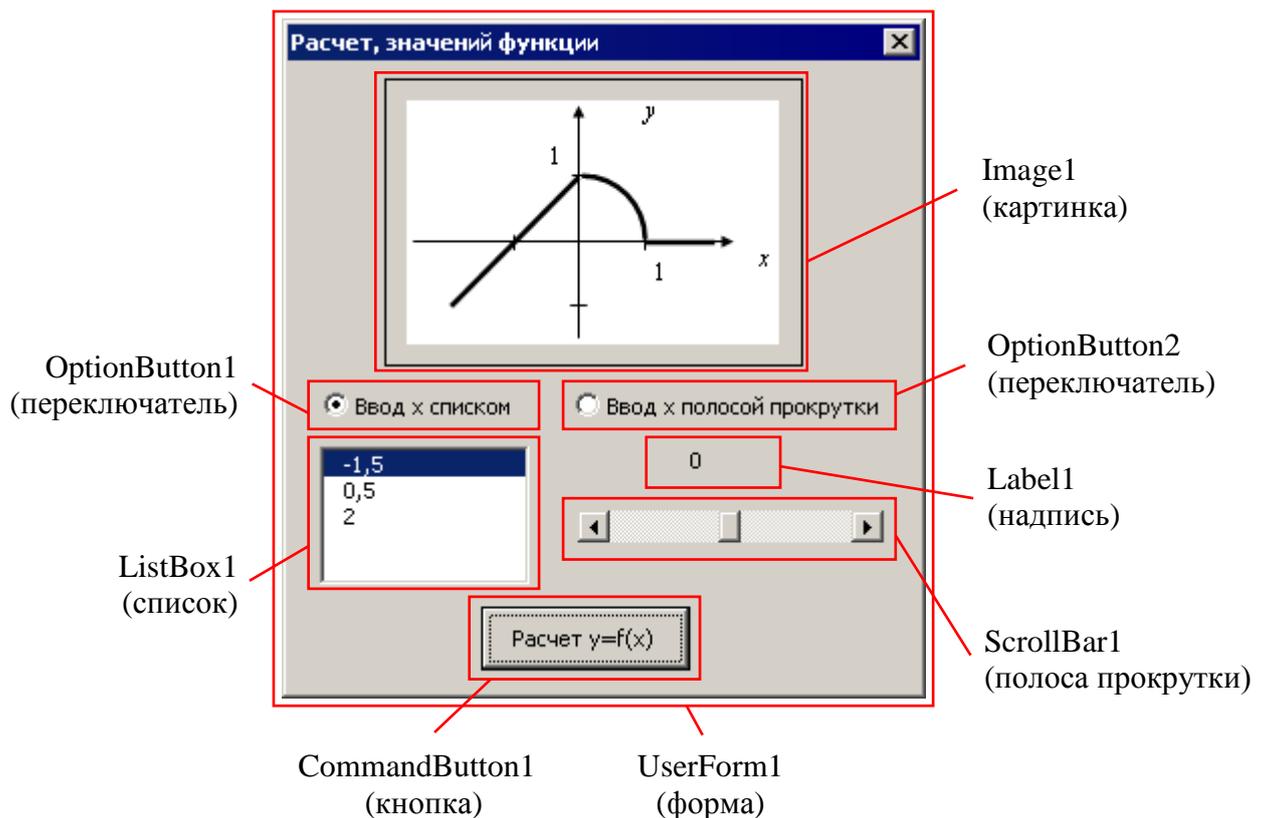
Изобразить блок-схему и написать программу для расчета $y = f(x)$ при различных значениях x .

... построить пользовательский интерфейс на основе формы VB, включающий:

- 1) изображение ("картинку") графического задания функции,
- 2) "кнопку" запуска расчета значения функции,
- 3) два "переключателя", дающие возможность выбрать один из двух вариантов ввода аргумента x для расчета функции,
- 4) элементы управления (для двух вариантов ввода аргумента (x)) следующих типов: а) "список", б) "полоса прокрутки".

Решение

Схема пользовательского интерфейса:



Значения аргумента, задаваемые с помощью списка: -1.5, 0.5, 2.

Диапазон значений аргумента, задаваемый с помощью полосы прокрутки: [-2...2].

Свойства элементов пользовательского интерфейса, задаваемые с помощью окна *Propeties*. приведены в табл. 15.

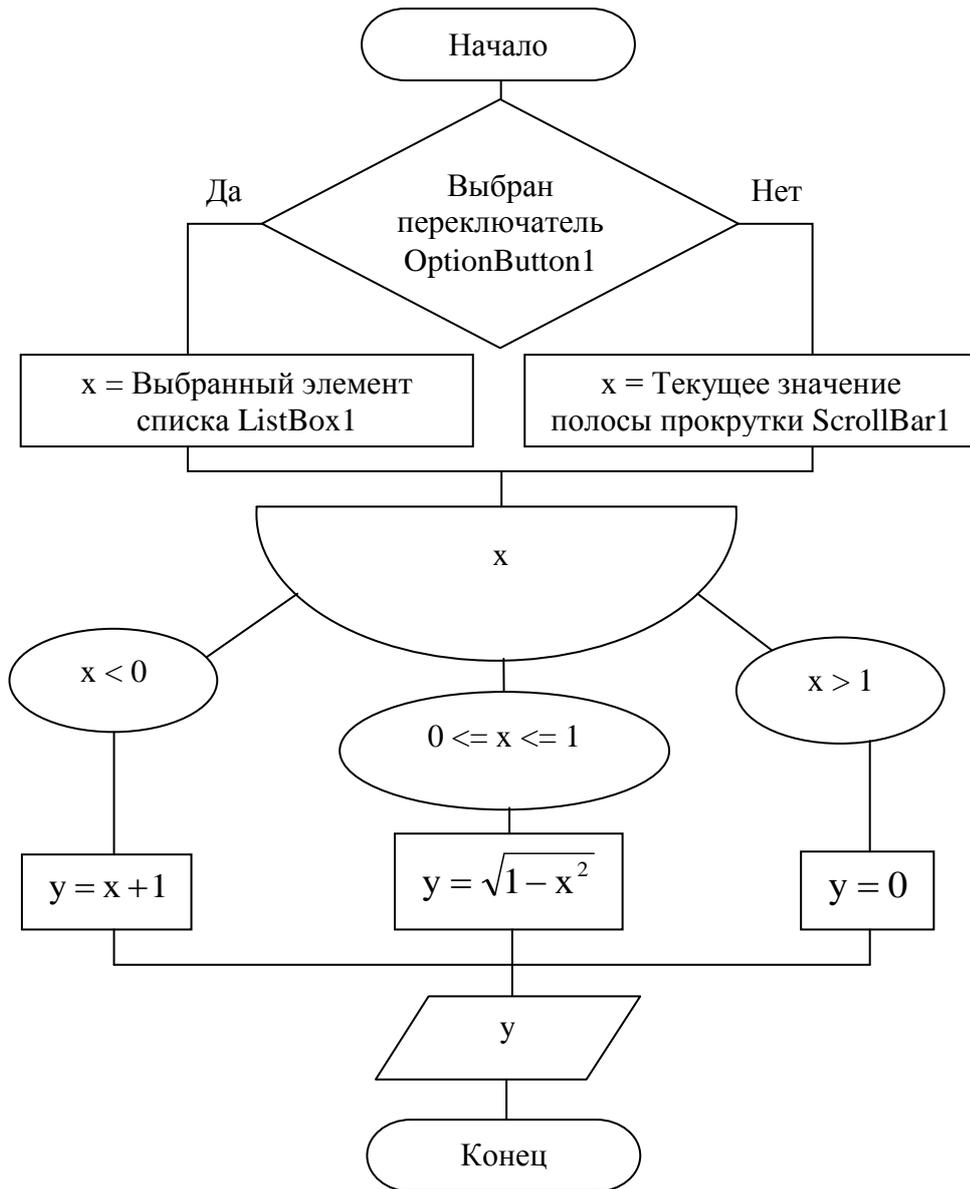
Таблица 15

Свойства элементов управления (пользовательского интерфейса)

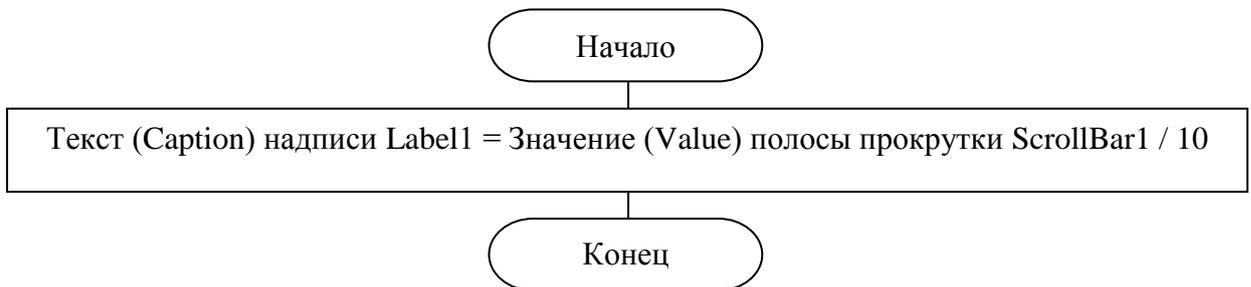
Свойство	Элемент управления							
	UserForm1 (форма)	OptionButton1 (переключатель)	OptionButton2 (переключатель)	ListBox1 (список)	Label1 (надпись)	ScrollBar1 (полоса прокрутки)	CommandButton1 (кнопка)	Image1 (картинка)
Name	User-Form1	Option-Button1	Option-Button2	ListBox1	Label1	Scroll-Bar1	Command-Button1	Image1
Width	240	84	122	78	36	114	78	156
Height	255	18	18	54	18	12	24	108
Left		12	106	12	150	106	72	36
Top		120	120	144	144	168	204	6
Caption	Расчет значений функции	Ввод x списком	Ввод x полосой прокрутки		""		Расчет $y=f(x)$	
GroupName		r1	r1					
Picture								sh.bmp

Алгоритм (блок-схема) решения задачи:

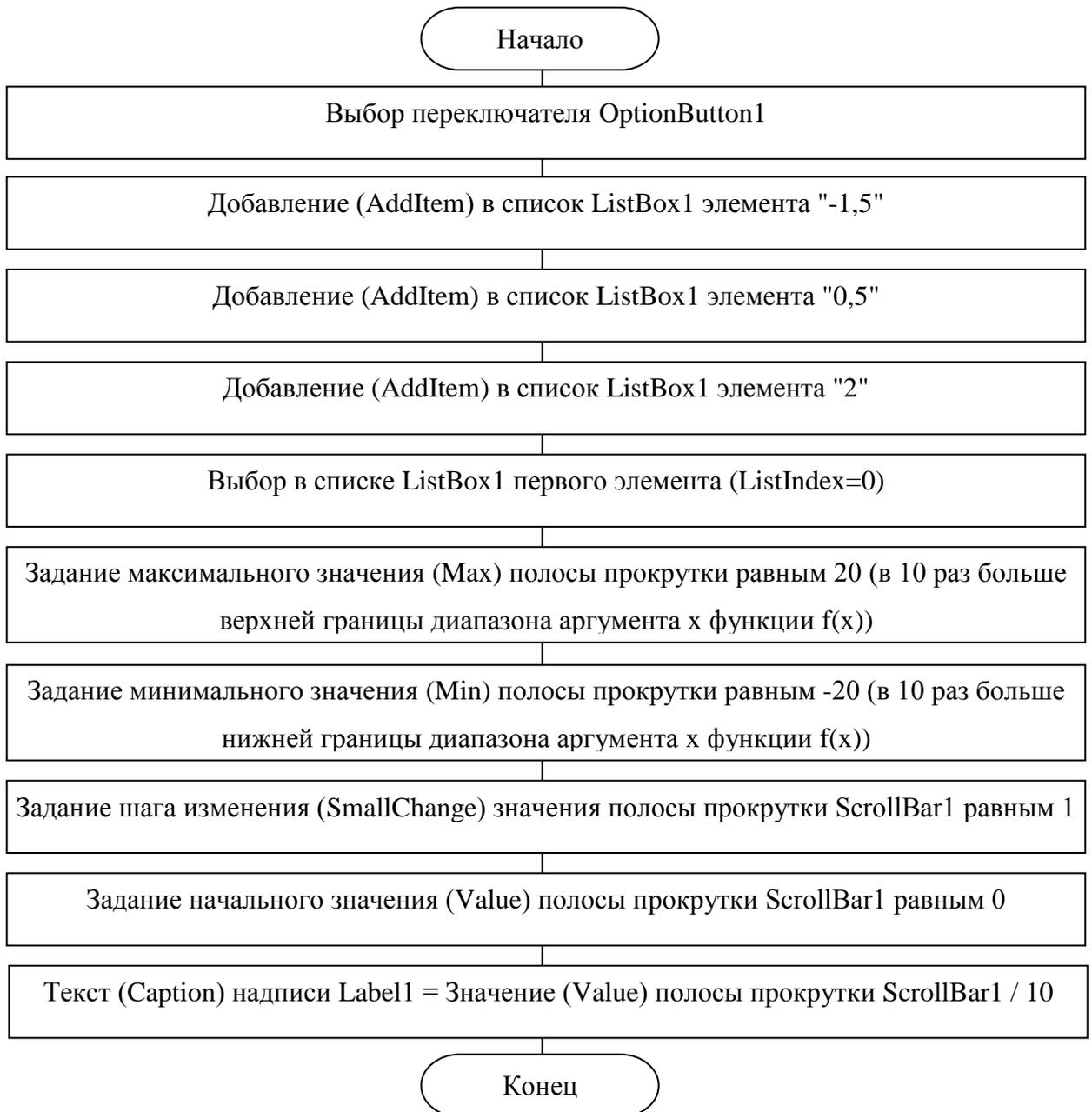
Процедура обработки щелчка по кнопке CommandButton1:



Процедура обработки изменения значения полосы прокрутки ScrollBar1:



Процедура обработки загрузки (отображения) формы UserForm1:



Программные модули:

Модуль пользовательской формы UserForm1

Private Sub CommandButton1_Click()

' Процедура обработки щелчка (Click) по кнопке

' по имени CommandButton1 (создается контекстной командой

' View Code для кнопки CommandButton1 формы UserForm1)

Dim x As Double ' Объявление переменной для аргумента

Dim y As Double ' Объявление переменной для функции

If OptionButton1.Value = True Then

' Проверка выбора первого способа ввода аргумента

```
' с помощью переключателя OptionButton1
x = CDbI(ListBox1.Value)
' Считывание текущего значения (Value) из списка ListBox1
' в переменную x с преобразованием к типу вещественное число
Else
' Вариант второго способа ввода аргумента, когда не выбран
' переключатель OptionButton1, а значит, выбран OptionButton2
x = ScrollBar1.Value / 10
' Считывание текущего значения (Value) из полосы прокрутки
' ScrollBar1 в переменную x с делением на 10 для перевода
' в диапазон задаваемых чисел
End If
Select Case x
Case Is < 0
y = x + 1
Case Is <= 1
y = Sqr(1 - x * x)
Case Else
y = 0
End Select
MsgBox "В x = " & x & " y = " & y
End Sub
' Завершение объявления процедуры CommandButton1_Click

Private Sub ScrollBar1_Change()
' Объявление процедуры обработки изменения (Change) значения
' в полосе прокрутки по имени ScrollBar1 (создается контекстной
' командой View Code для объекта ScrollBar1)
Label1.Caption = ScrollBar1.Value / 10
' Считывание в подпись (Caption) объекта надписи (Label1)
```

```
' значения из полосы прокрутки ScrollBar1  
' с делением на 10 для перевода в диапазон задания аргумента  
End Sub ' Завершение объявления процедуры ScrollBar1_Change
```

```
Private Sub UserForm_Activate()
```

```
' Объявление процедуры при активации или отображении
```

```
' (Activate) формы пользовательского интерфейса (UserForm)
```

```
OptionButton1.Value = True
```

```
' Начальный выбор переключателя OptionButton1
```

```
ListBox1.AddItem "-1,5"
```

```
' Добавление элемента-значения в список по имени ListBox1
```

```
ListBox1.AddItem "0,5"
```

```
ListBox1.AddItem "2"
```

```
ListBox1.ListIndex = 0
```

```
' Начальный выбор первого элемента-значения в списке
```

```
ScrollBar1.Max = 20
```

```
' Задание максимального значения, устанавливаемого
```

```
' в полосе прокрутки: в 10 раз больше верхней границы
```

```
' диапазона аргумента
```

```
ScrollBar1.Min = -20
```

```
' Задание минимального значения, устанавливаемого
```

```
' в полосе прокрутки: в 10 раз больше нижней границы
```

```
' диапазона аргумента
```

```
ScrollBar1.SmallChange = 1
```

```
' Задание шага изменения значения полосы прокрутки ScrollBar1
```

```
ScrollBar1.Value = 0
```

```
' Задание начального значения полосы прокрутки ScrollBar1
```

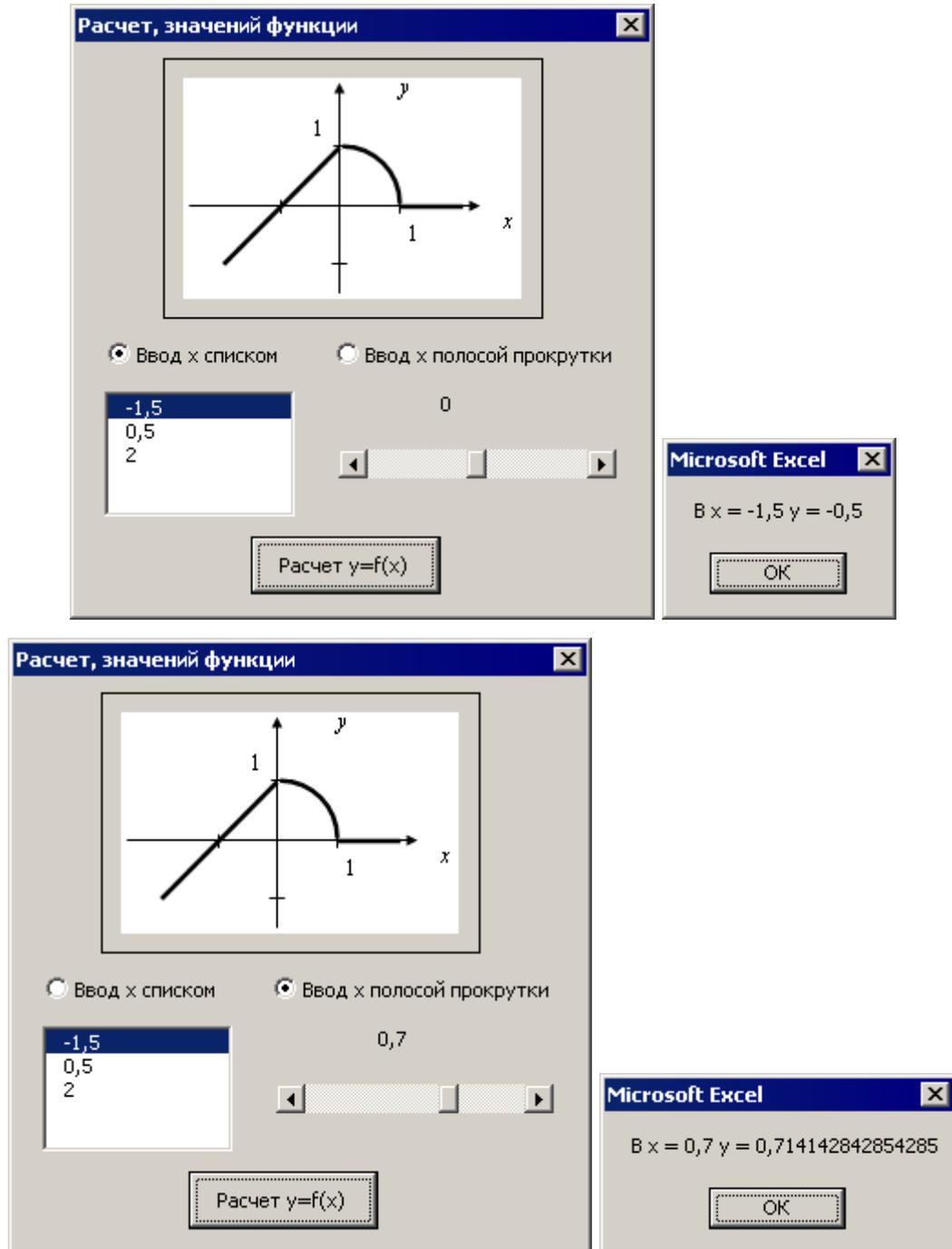
```
Label1.Caption = ScrollBar1.Value / 10
```

```
' Считывание в подпись (Caption) объекта надписи (Label1)
```

```
' значения из полосы прокрутки ScrollBar1
```

' с делением на 10 для перевода в диапазон задания аргумента
 End Sub ' Завершение объявления процедуры UserForm_Activate

Результаты работы программы:



Задание

Для Задания Б из лабораторной работы № 3 "Условные операторы" построить пользовательский интерфейс на основе формы VB, включающий:

1) изображение ("картинку") графического задания функции,

- 2) "кнопку" запуска расчета значения функции,
- 3) два "переключателя", дающие возможность выбрать один из двух способов ввода аргумента x для расчета функции,
- 4) элементы управления (для двух способов ввода аргумента (x)) указанных в табл. 16 типов.

Таблица 16

Варианты заданий к лабораторной работе № 7

№ варианта	Типы элементов управления для двух способов ввода аргумента (x) для расчета функции, заданной графически
1, 4, 7, 10, 13, 16, 19, 22, 25, 28	а) "поле ввода", б) "список"
2, 5, 8, 11, 14, 17, 20, 23, 26, 29	а) "поле ввода", б) "полоса прокрутки"
3, 6, 9, 12, 15, 18, 21, 24, 27, 30	а) "список", б) "полоса прокрутки"

Состав отчета

1. Номер, название и цель работы.
2. Текст задания.
3. Алгоритм решения задания.
4. Схема пользовательского интерфейса
4. Листинг (текст) программы.
5. Результаты работы программы.

Контрольные вопросы

1. Перечислите типы элементов управления. Каково их назначение?
2. Какие Вам известны события для элементов управления?
3. Перечислите и охарактеризуйте свойства, необходимые для работы с элементом управления типа "список".

3. Перечислите и охарактеризуйте свойства, необходимые для работы с элементом управления типа "полоса прокрутки".
4. Перечислите и охарактеризуйте основные свойства, необходимые для работы с элементами управления типа "поле ввода" и "надпись".
5. Перечислите и охарактеризуйте основные свойства, необходимые для работы с элементами управления типа "флажок" и "переключатель". В чем отличие между этими элементами управления? Как оно отражается на использовании их свойств?
6. Перечислите и охарактеризуйте основные свойства, необходимые для работы с элементами управления типа "картинка".

ЛАБОРАТОРНАЯ РАБОТА № 8. ЧИСЛЕННЫЕ МЕТОДЫ

Цель работы

Изучить подходы к решению прикладных задач математики с использованием ЭВМ.

Теоретические сведения

Численные методы – методы решения задач, сводящиеся к арифметическим и логическим действиям над числами, выполнить которые наиболее удобно с использованием ЭВМ. Использование численных методов обусловлено наличием задач, решение которых неудобно (невозможно) представить в виде известных элементарных функций. Решение, полученное численным методом, обычно является приближенным, т.е. содержит некоторую погрешность.

1. Решение уравнений

Рассмотрим решение уравнения одной переменной вида: $f(x) = 0$.

Данное уравнение не всегда точно разрешимо относительно x . С использованием численных методов решение всегда может быть найдено с какой-то погрешностью.

Метод Ньютона

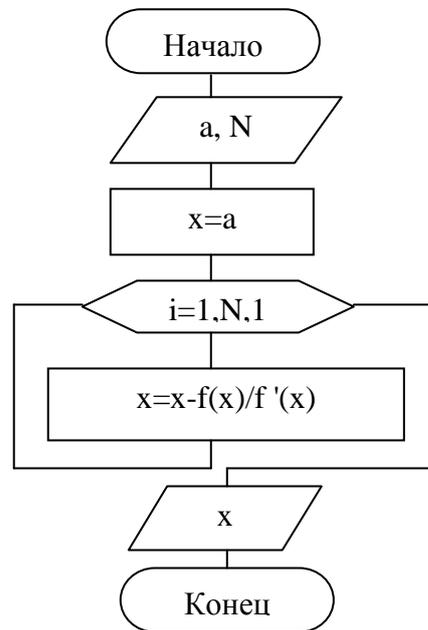
Предназначен для поиска решения уравнения на интервале значений x , содержащем единственное значение для которого соблюдается $f(x) = 0$.

Метод заключается в следующем:

- 1) строится касательная в начальной точке интервала x к графику функции $f(x)$;
- 2) находится точка пересечения касательной с осью x – приближенное решение;
- 3) точка пересечения касательной с осью x становится начальной точкой, и последовательность 1)–3) повторяется необходимое число раз. При этом, чем

больше будет произведено повторов, тем ближе приближенное решение к точному.

Алгоритм метода Ньютона представлен в блок-схеме:



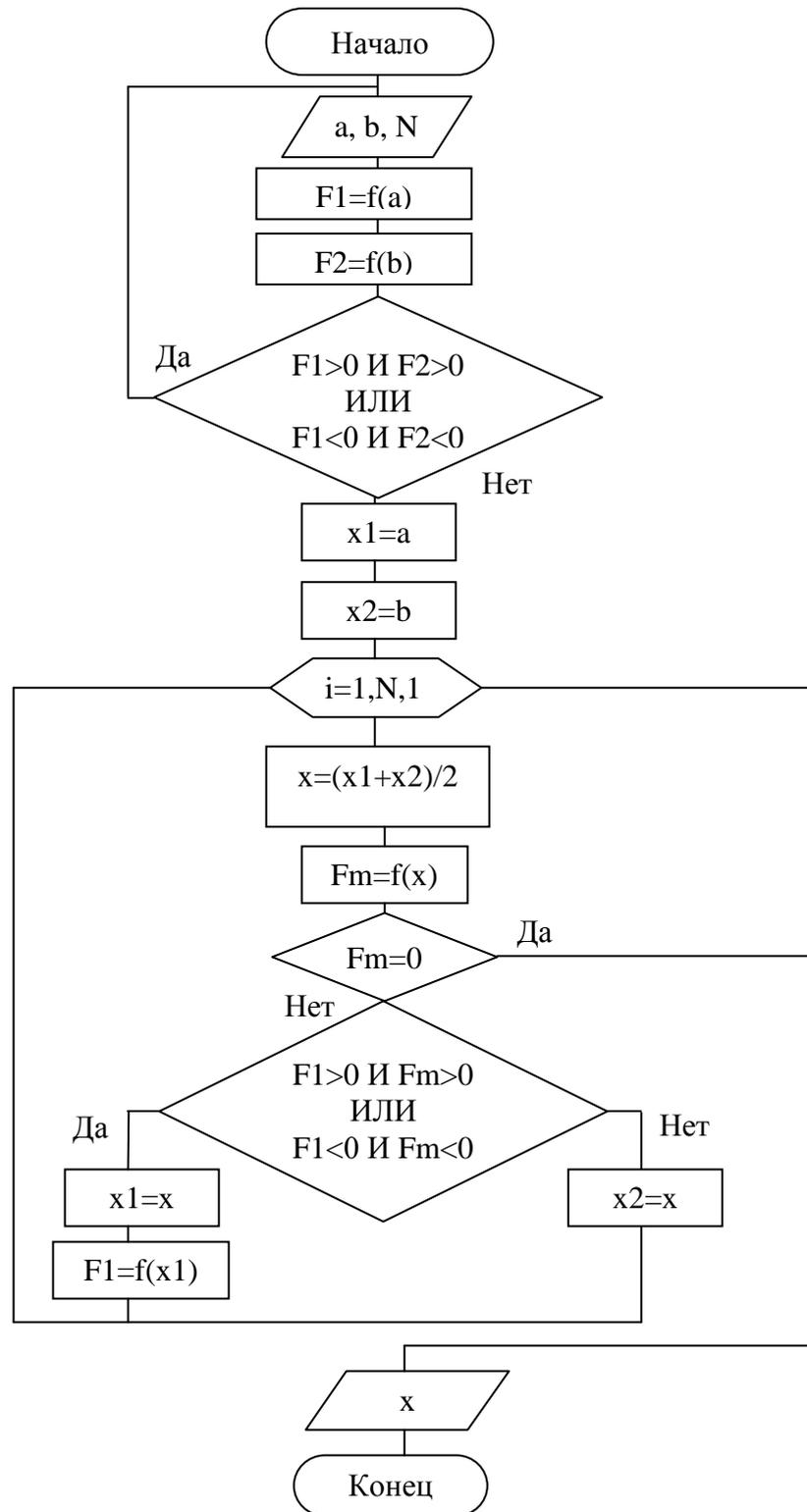
В блок-схеме a – одна из границ интервала поиска x , $x = x - f(x)/f'(x)$ – рекуррентная формула для поиска точки пересечения касательной с осью x , $f'(x)$ – производная функции в текущей точке, N – количество приближений.

Метод деления отрезка пополам

Суть метода:

- 1) на интервале, включающем решение уравнения, вычисляются значения $f(x)$ для его границ;
- 2) интервал делится пополам и вычисляется $f(x)$ для его середины;
- 3) значение шага 2 сравнивается со значениями шага 1;
- 4) значение шага 2 становится граничным значением вместо общего с ним по знаку значения шага 1, а соответствующее значение x – соответствующей границей интервала и текущим решением. Шаги 1–4 повторяются для достижения необходимой точности.

Алгоритм метода представлен в блок-схеме:



В алгоритме: a и b – исходный интервал расположения решения.

2. Решение систем уравнений

Рассмотрим решение системы линейных уравнений с N неизвестными

$(X_1 \dots X_N)$

$$\begin{array}{cccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1(N)}x_N & = & a_{1(N+1)} \\
a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2(N)}x_N & = & a_{2(N+1)} \\
\dots & & \dots & & \dots & & \dots & & \dots \\
a_{(N)1}x_1 & + & a_{(N)2}x_2 & + & \dots & + & a_{NN}x_N & = & a_{N(N+1)}
\end{array}$$

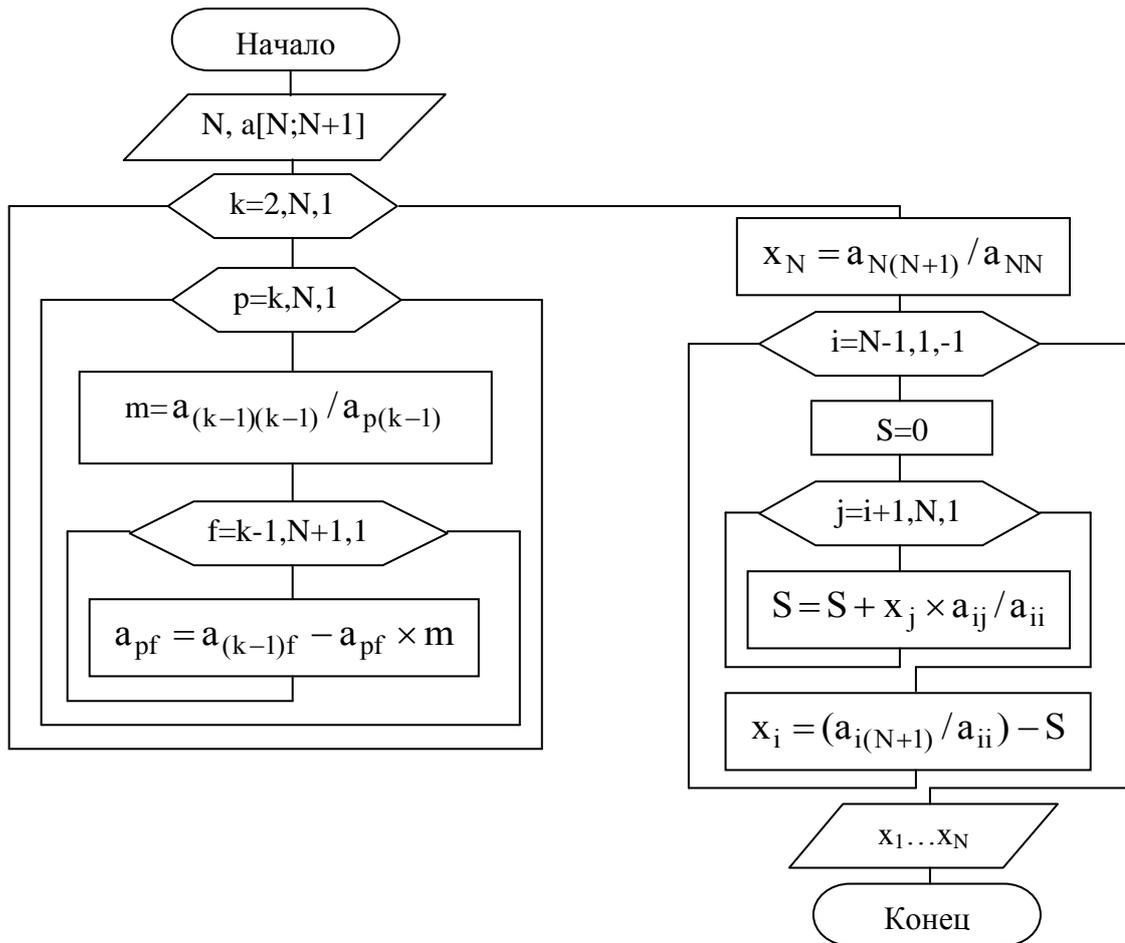
методом Гаусса.

Сущность метода заключается в преобразовании исходной системы в треугольную путем последовательного исключения неизвестных x при помощи дополнительных множителей к коэффициентам a и вычитания нижестоящих строк системы из вышестоящих.

$$\begin{array}{cccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1(N)}x_N & = & a_{1(N+1)} \\
& & a'_{22}x_2 & + & \dots & + & a'_{2(N)}x_N & = & a'_{2(N+1)} \\
& & \dots & & \dots & & \dots & & \dots \\
& & & & & & a'_{NN}x_N & = & a'_{N(N+1)}
\end{array}$$

Из треугольной системы, начиная с последнего уравнения, последовательно находятся x_N, x_{N-1}, \dots, x_1 .

Алгоритм метода представлен в блок-схеме:



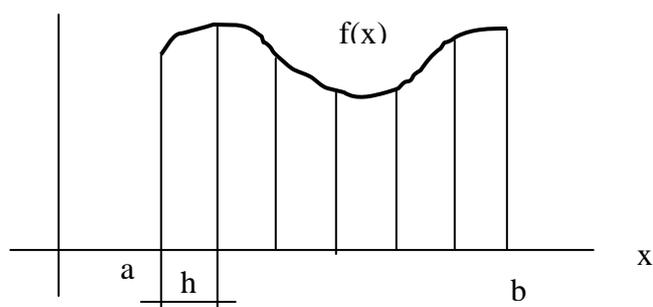
В первой половине алгоритма происходит построение треугольной матрицы (m – дополнительные множители), а во второй – расчет корней системы $x_1 \dots x_N$.

3. Расчет определенных интегралов

Задача расчета определенного интеграла вида

$$I = \int_a^b f(x) dx$$

сводится к определению площади фигуры, ограниченной графиком функции $f(x)$, осью x и вертикалями, опущенными на ось x в границах интервала интегрирования:

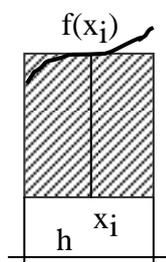


Для определения площади фигура разбивается на элементарные фрагменты с определенным шагом h по оси x .

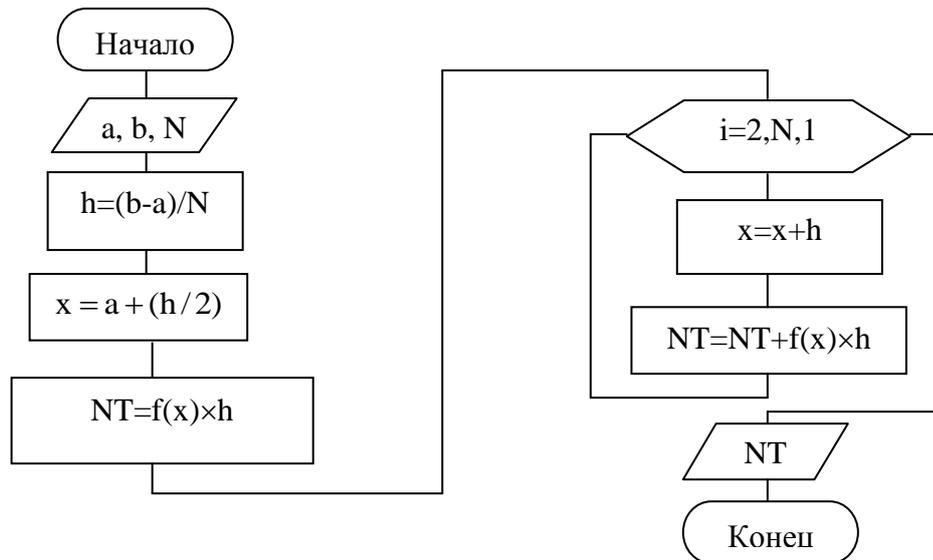
Общая площадь определяется суммой площадей элементарных фрагментов. Площадь каждого фрагмента легко определить, если задать упрощенный вид его "верхней" границы. В зависимости от способа задания этой границы существует несколько методик расчета определенных интегралов.

Метод прямоугольников

По этому методу верхняя граница элементарного фрагмента представляется горизонтальной линией, высота расположения которой определяется значением функции $f(x)$ для середины фрагмента:



Алгоритм расчета определенного интеграла (NT) этим методом для N фрагментов представлен в блок-схеме:



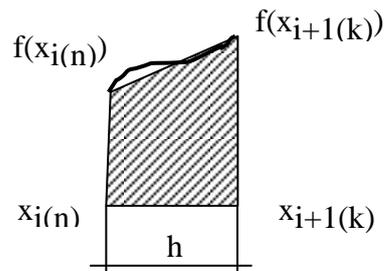
Погрешность метода прямоугольников

$$T = \frac{h^3}{24} f''(\xi) ,$$

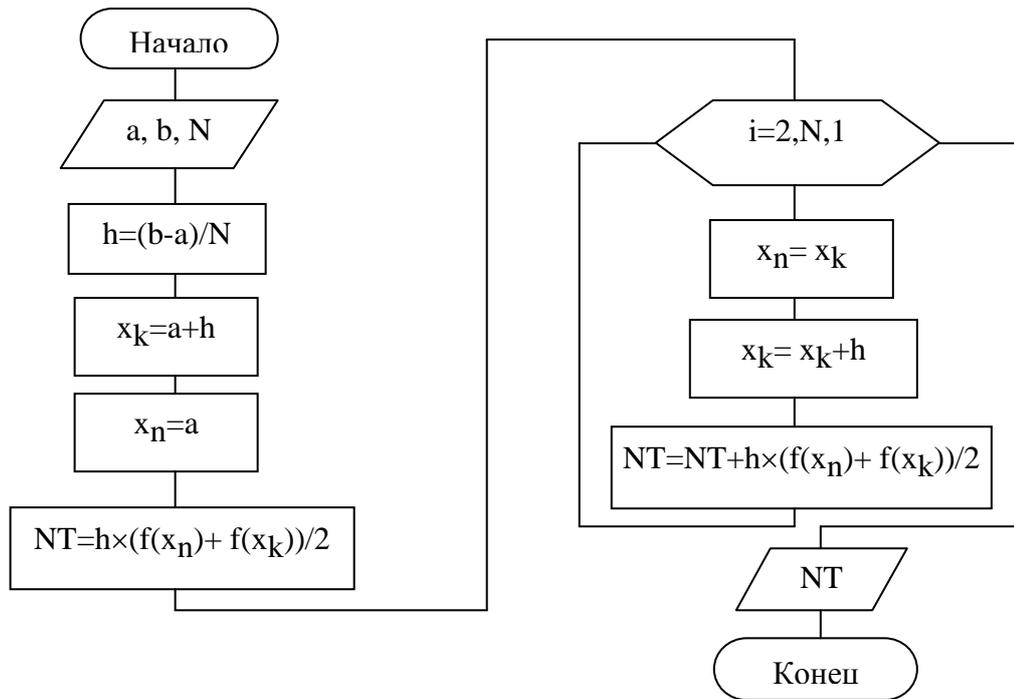
где $f''(\xi)$ – максимум второй производной на интервале $[a, b]$.

Метод трапеций

По этому методу верхняя граница элементарного фрагмента представляется прямой линией, соединяющей значения функции для границ фрагмента, таким образом, элементарный фрагмент – трапеция:



Алгоритм расчета определенного интеграла (NT) этим методом для N фрагментов представлен в блок-схеме:



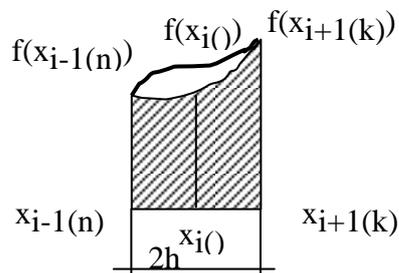
Погрешность метода прямоугольников

$$T = \frac{h^3}{12} f''(\xi),$$

где $f''(\xi)$ – максимальное значение второй производной на интервале $[a, b]$.

Метод Симпсона (парабол)

По этому методу верхняя граница элементарного фрагмента представляется участком параболы, проходящей через точки со значениями функции $f(x)$ для границ фрагмента и его середины:



Парабола для вершины фрагмента задается формулой

$$f(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_{i-1})}{2h} x + \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1}))}{2h^2} x^2,$$

тогда определенный интеграл (площадь) для фрагмента (в диапазоне $2h$) будет

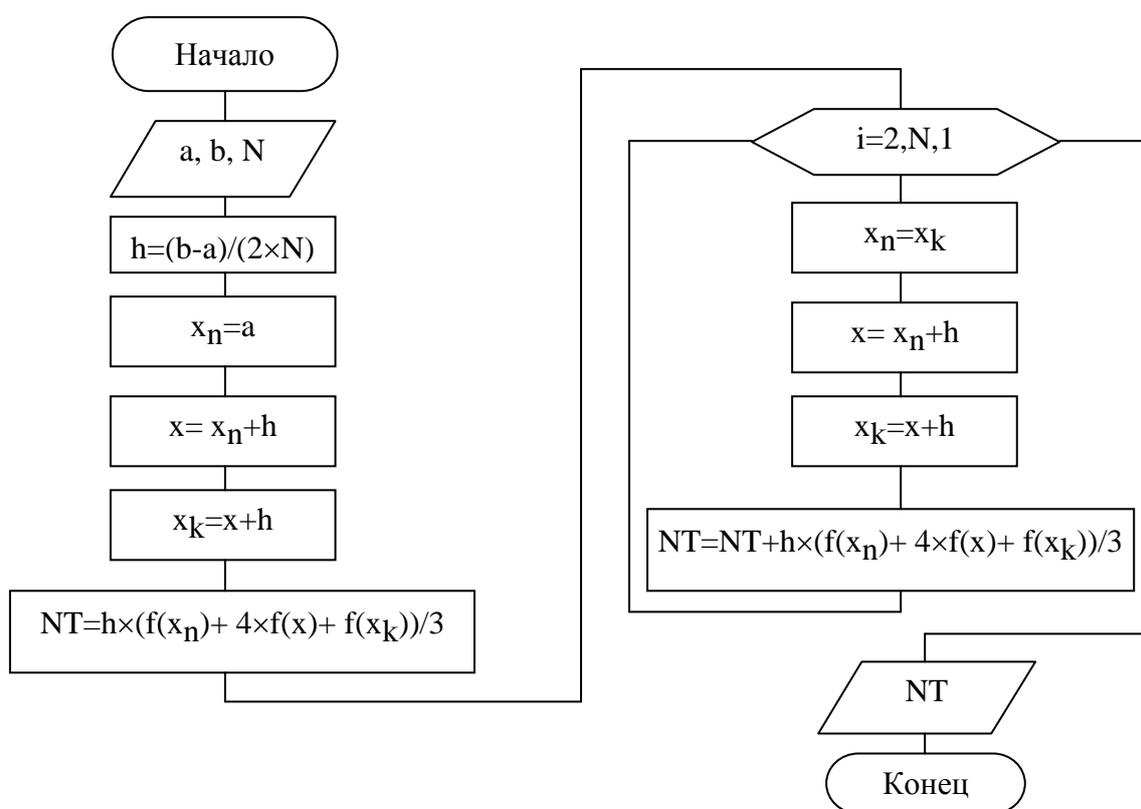
$$\int_{-h}^h f(x)dx = \frac{h}{3} (f(x_{i-1}) + 4f(x_i) + f(x_{i+1})).$$

Погрешность метода парабол

$$T = \frac{h^5}{90} f^{IV}(\xi),$$

где $f^{IV}(\xi)$ – максимальное значение второй производной на интервале $[a, b]$.

Алгоритм расчета определенного интеграла (NT) этим методом для N фрагментов представлен в блок-схеме:



Задание

1. Составить алгоритмы и написать программы для нахождения корня уравнения методом Ньютона и методом деления отрезка пополам. Варианты заданий представлены в табл. 17.

Таблица 17

Варианты задания для численного решения уравнений

№ варианта	Уравнение, интервал поиска корня
------------	----------------------------------

1	$-x^3 - 11x^2 + 3x + 2 = 0, x \in [-2;0]$
2	$1,5x^3 + 3x^2 + x - 1 = 0, x \in [0;1]$
3	$\cos(-3x^2 - 2x + 2) = 0, x \in [1,1;1,33]$
4	$7x^3 + 2x^2 - 2x - 2 = 0, x \in [0,4;0,9]$
5	$\sin(2x^2 - 5x + 7) = 0, x \in [-0,9;-0,67]$
6	$-2x^3 + x^2 + 2x - 1 = 0, x \in [-1,1;-0,5]$
7	$\cos(2x^2 - 3x + 5) = 0, x \in [-2;-1,7]$
8	$-4x^3 + 3x^2 - 2x + 3 = 0, x \in [0,5;1,5]$
9	$\sin(3x^2 + x - 4) = 0, x \in [1,92;2]$
10	$2x^3 - 3x^2 - 2x - 3 = 0, x \in [2;2,5]$
11	$\cos(-3x^2 + 5x - 1) = 0, x \in [-1,88;-1,76]$
12	$-3x^3 + 2x^2 + x - 2 = 0, x \in [-1;-0,5]$
13	$\sin(-2x^2 + 3x + 2) = 0, x \in [1,7;2,1]$
14	$-2x^3 - 3x^2 - 4x - 1 = 0, x \in [-0,8;0]$
15	$\cos(-3x^2 + 5x - 1) = 0, x \in [-1,88;-1,76]$
16	$-x^3 + 3x^2 - 2x - 4 = 0, x \in [-1;0]$
17	$\cos(4x^2 - 3x - 2) = 0, x \in [-1,88;-1,77]$
18	$2x^3 - 3x^2 - x + 3 = 0, x \in [-2;-0,5]$
19	$\sin(3x^2 - 4x - 5) = 0, x \in [1,68;1,7]$
20	$4x^3 - x^2 + 3x + 4 = 0, x \in [-1;-0,5]$
21	$3x^3 + 2x^2 - x + 3 = 0, x \in [-1,5;-1]$
22	$\cos(-3x^2 + x + 3) = 0, x \in [-1,76;-1,62]$

23	$-5x^3 - 3x^2 + 2x - 6 = 0, x \in [-1,5;-1]$
24	$\sin(-2x^2 - x - 6) = 0, x \in [1,45;1,65]$
25	$4x^3 - x^2 + 3x - 3 = 0, x \in [0,5;1]$
26	$\cos(4x^2 + x + 2) = 0, x \in [-1,7;-1,6]$
27	$-3x^3 - 2x^2 - 4x + 1 = 0, x \in [0;0,5]$
28	$\sin(-3x^2 + 2x - 5) = 0, x \in [-1;-0,8]$
29	$3x^3 + 4x^2 + x + 2 = 0, x \in [-2;-1]$
30	$\sin(x^2 + x + 1) = 0, x \in [0,8;1,3]$

2. Составить алгоритм и написать программу для нахождения корней системы уравнений методом Гаусса. Представить в отчете аналитическое решение и результаты программного расчета. Варианты заданий представлены в табл. 18.

Таблица 18

Варианты задания для численного решения систем уравнений

№ варианта	Система уравнений	№ варианта	Система уравнений
1	$\begin{cases} 2x + 3y - 5z = 6 \\ x - y + z = 2 \\ -x + y - 4z = -5 \end{cases}$	2	$\begin{cases} 21x + 13y - 15z = 16 \\ x - y - 3z = -12 \\ -21x - y + 14z = 15 \end{cases}$
3	$\begin{cases} -x + 6y - 4z = 6 \\ 2x + 7y - 4z = 15 \\ 4x - 7y - z = -1 \end{cases}$	4	$\begin{cases} -13x - 2y + 15z = 26 \\ 2x - 5y - z = 12 \\ -3x + y - 2z = -3 \end{cases}$

5	$\begin{cases} 21x + 3y + 15z = -4 \\ -4x - 2y + 15z = -2 \\ -x + 7y + 13z = 21 \end{cases}$	6	$\begin{cases} 5x + 13y - 15z = 6 \\ -2x - y - 3z = -12 \\ x - 4y - 4z = -5 \end{cases}$
7	$\begin{cases} 5x - y - z = 1 \\ 4x + 4y - 7z = 12 \\ -3x - 2y - 5z = -15 \end{cases}$	8	$\begin{cases} 22x + 31y + z = 16 \\ 12x - y - 3z = 1 \\ -x + 5y + 4z = -15 \end{cases}$
9	$\begin{cases} -2x - 13y + 15z = -36 \\ -51x - y - 2z = 12 \\ 2x + 5y - 6z = 25 \end{cases}$	10	$\begin{cases} 12x - 4y - 5z = -11 \\ 11x + y + z = -12 \\ -3x - 2y - 14z = 14 \end{cases}$
11	$\begin{cases} -2x - 6y + 5z = 1 \\ 2x + 3y + 4z = 20 \\ 3x - 3y - 5z = -8 \end{cases}$	12	$\begin{cases} -2x - 5y - 7z = -16 \\ x + 2y + 2z = 2 \\ -11x + y + 14z = -8 \end{cases}$
13	$\begin{cases} 12x - 13y + 5z = -16 \\ 35x + 12y - 2z = -22 \\ -x + y - 4z = 5 \end{cases}$	14	$\begin{cases} 8x + 6y + 4z = 6 \\ 2x - y + 10z = 2 \\ -x + 3y - 8z = 15 \end{cases}$
15	$\begin{cases} 8x + 6y + 4z = -6 \\ 2x - y + 10z = -2 \\ -x + 3y - 8z = -15 \end{cases}$	16	$\begin{cases} 12x - 13y + 5z = 16 \\ 35x + 12y - 2z = 22 \\ -x + y - 4z = -5 \end{cases}$
17	$\begin{cases} -2x - 5y - 7z = 16 \\ x + 2y + 2z = -2 \\ -11x + y + 14z = 8 \end{cases}$	18	$\begin{cases} -2x - 6y + 5z = -1 \\ 2x + 3y + 4z = -20 \\ 3x - 3y - 5z = 8 \end{cases}$
19	$\begin{cases} 12x - 4y - 5z = 11 \\ 11x + y + z = 12 \\ -3x - 2y - 14z = -14 \end{cases}$	20	$\begin{cases} -2x - 13y + 15z = 36 \\ -51x - y - 2z = -12 \\ 2x + 5y - 6z = -25 \end{cases}$

21	$\begin{cases} 22x + 31y + z = -16 \\ 12x - y - 3z = -1 \\ -x + 5y + 4z = 15 \end{cases}$	22	$\begin{cases} 5x - y - z = -1 \\ 4x + 4y - 7z = -12 \\ -3x - 2y - 5z = 15 \end{cases}$
23	$\begin{cases} 5x + 13y - 15z = -6 \\ -2x - y - 3z = 12 \\ x - 4y - 4z = 5 \end{cases}$	24	$\begin{cases} 21x + 3y + 15z = 4 \\ -4x - 2y + 15z = 2 \\ -x + 7y + 13z = -21 \end{cases}$
25	$\begin{cases} -13x - 2y + 15z = -26 \\ 2x - 5y - z = -12 \\ -3x + y - 2z = 3 \end{cases}$	26	$\begin{cases} -x + 6y - 4z = -6 \\ 2x + 7y - 4z = -15 \\ 4x - 7y - z = 1 \end{cases}$
27	$\begin{cases} -12x - 2y + 7z = 16 \\ 35x + y - 3z = 12 \\ -4x + y - 14z = -25 \end{cases}$	28	$\begin{cases} 21x + 13y - 15z = -16 \\ x - y - 3z = 12 \\ -21x - y + 14z = -15 \end{cases}$
29	$\begin{cases} -12x + 3y + 5z = -16 \\ x - 3y - z = -2 \\ -x + 6y + 4z = -5 \end{cases}$	30	$\begin{cases} 2x + 3y - 5z = -6 \\ x - y + z = -2 \\ -x + y - 4z = 5 \end{cases}$

3. Составить алгоритмы и написать программы для нахождения

определенного интеграла $I = \int_a^b f(x)dx$ методами прямоугольников, трапеций

и методом Симпсона; функция $f(x)$ – левая часть уравнения из задания 1, а и b – границы интервала аргумента x из задания 1. Определить величину интеграла для каждого из численных методов при равном числе элементарных фрагментов N (н.п., при $N = 10$).

Состав отчета

1. Номер, название и цель работы.
2. Текст задания по варианту.

3. Алгоритмы (блок-схемы) решения задач варианта (для каждого численного метода привести его название).
3. Листинги (тексты) программ по заданиям варианта (для каждого численного метода привести его название).
4. Результаты (информация, выводимая на экран) работы программ (тестовое выполнение).

Контрольные вопросы

1. Поясните основные особенности численных методов при решении задач математики.
2. В чем заключается численный метод решения уравнений методом Ньютона?
3. В чем заключается численный метод решения уравнений методом деления отрезка пополам?
3. В чем заключается численный метод решения систем уравнений методом Гаусса?
4. Поясните идею нахождения численными методами значения определенного интеграла.
5. В чем заключается принципиальное отличие между методами прямоугольников, трапеции и парабол при численном расчете определенных интегралов? Какова погрешность этих методов?