МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономика и управление»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Методические рекомендации к лабораторным работам для студентов направления подготовки 27.03.05 «Инноватика» дневной формы обучения

Могилев 2022

УДК 004.45
ББК 32.973-018.2
П78

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Экономика и управление» «21» января 2022 г., протокол № 6

Составитель ст. преподаватель Е. Г. Галкина

Рецензент канд. техн. наук, доц. Т. В. Пузанова

Методические рекомендации к лабораторным работам предназначены для студентов направления подготовки 27.03.05 «Инноватика» дневной формы обучения, изучающих дисциплину «Программное обеспечение для решения задач профессиональной деятельности».

Учебно-методическое издание

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Ответственный за выпуск	И. В. Ивановская
Корректор	А. А. Подошевко
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать . Формат 60×84 /16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. .Уч.-изд. л. . Тираж 36 экз. Заказ

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев

Содержание

Порядок выполнения и защиты лабораторных работ

Перед выполнением лабораторной работы студент должен ознакомиться с соответствующей темой конспекта лекций, а в случае необходимости – с рекомендуемой по дисциплине литературой.

В результате выполнения лабораторной работы студенту необходимо разработать алгоритм решения задачи и набрать текст программы в редакторе кода VBA. Затем программа должна быть проверена на предмет наличия ошибок синтаксиса (этап компиляции), ошибок в использовании библиотечных и пользовательских функций (этап компоновки), а также логических ошибок или ошибок алгоритма, которые выявляются уже на стадии выполнения. После внесения необходимых исправлений в исходный код, он вновь подвергается тестированию до тех пор, пока не будет получен правильный результат.

Если алгоритм программы имеет несколько вычислительных ветвей, то каждая ветвь программы должна быть протестирована отдельно, а результаты тестирования должны быть зафиксированы в отчете.

По окончании процесса тестирования студент должен оформить отчет по лабораторной работе, который включает в себя задание на лабораторную работу, схему алгоритма, распечатку текста программы и результаты ее выполнения.

Текст программы должен содержать необходимый объем комментариев, указывающих назначение программы и ее отдельных частей, метод решения, смысл основных используемых переменных. Идентификаторы переменных должны быть по возможности близки к аббревиатурам названий этих переменных. Схема алгоритма должна быть изображена в соответствии с ГОСТ 19.701–90 (ИСО 5807–85).

К защите лабораторной работы допускаются только студенты, выполнившие работу и оформившие отчет. Защита проходит в форме устного и письменного собеседования, когда студент отвечает на вопросы преподавателя, которые приведены в данных методических рекомендациях в списке контрольных вопросов к каждой лабораторной работе, а также дополняет свои ответы письменно примерами программного кода. В случае успешной защиты преподаватель ставит на отчете свою подпись и дату защиты.

Лабораторная работа № 1. Линейная программа. Организация ввода-вывода

Цель работы: ознакомиться с типами данных, основными операциями языка программирования и функциями ввода-вывода, научиться использовать переменные и константы.

Задание

Создать процедуру *Корзина*.

Объявить четыре переменные типа *Double*: *USD*, *EUR*, *RUR*, *Корзина*.

Присвоить переменным *USD*, *EUR*, *RUR* первоначальные значения с помощью оператора присваивания «=», например:*USD = 11000.*

Ввести формулу для расчета значения корзины валют: *Корзина* = (*USD*EUR*RUR*)^(*1/3*).

Вывести значение корзины на экран в следующем формате: «*Корзина валют: ___*».

Создать на листе *MS Excel* таблицу, представленную на рисунке 1.

	A	B
1	**Валюта**	**Курс**
2	USD	
3	EUR	
4	RUR	
5		
6	**Корзина**	

Рисунок 1 – Пример таблицы для выполнения практического задания

Доработать ранее выполненное задание, реализовав в нем запись значений валют и величины корзины в соответствующие ячейки листа *MS Excel*.

Методические указания

Язык программирования *Visual Basic for Application* (*VBA*) имеет свои правила написания программного кода. Он использует свой алфавит, включающий буквы латинского алфавита и кириллицу; цифры от 0 до 9; символ подчеркивания. Из этих символов состоят имена процедур, переменных, меток переходов, константы и команды. *VBA* нечувствителен к регистру.

Переменная – это область памяти, в которой находятся данные, которыми оперирует программа. При выполнении алгоритма переменная может менять свое значение. Когда переменной присваивается новое значение, старое стирается.

Константа – это элемент данных, не меняющий своего значения.

В упрощенном виде программа линейной структуры представлена в таблице 1.

Таблица 1 – Программа линейной структуры

Описание оператора (команды)	Оператор (команда)
1 Заголовок подпрограммы, включающий тип подпрограммы (процедура или функция), заголовок подпрограммы, список аргументов и пр.	*Public Sub Процедура*()
2 Объявление переменных	*Dim x As Integer, y As Integer*
3 Ввод данных (инициализация переменных, т. е. присваивание им первоначальных значений) каким-либо способом, например:	
с помощью оператора присваивания	*x* = 2
ввод с клавиатуры	*x = InputBox(“Введите x”)*
считывание с листа *Excel*	*x= Cells(1,1).Value*
4 Операторы, выполняющие заданные действия по обработке информации (формулы и пр.)	*y = x ^ 2*
5 Вывод данных на экран	*MsgBox “Результат: ” & y*
6 Завершение подпрограммы	*End Sub*

Тип данных определяет множество допустимых значений, которое может принимать указанная переменная.

В VBA имеется 15 разновидностей типов данных. Основные из них представлены в таблице 2.

Таблица 2 – Типы данных

Тип данных	Размер, байт	Диапазон значений
Byte (короткий целый беззнаковый)	1	Целые числа от 0 до 255
Integer (целый)	2	Целые числа от –32 768 до +32 767
Long (длинный целый)	4	Целые числа от –2 147 483 648 до +2 147 483 647
Single (с плавающей запятой одинарной точности)	4	От –3,402823E+38 до –1,401298E-45 для отрицательных значений и от 1,401298E-45 до 3,4028235E+38 для положительных
Double (с плавающей запятой двойной точности)	8	От –1,797693 до –4,940656E-324 для отрицательных значений и от 4,941E-324 до 1,798E+308 для положительных
Boolean (логический)	2	True, False
String (строковый переменной длины)	10 + 1 байт на символ	Длина строки от 0 до 2,1 млрд символов
Variant (универсальный)	Для чисел 16 байт	Значения соответствуют одному из типов данных, определяемому автоматически
	Для строк 22 + 1 байт на символ	Длина строки от 0 до 2,1 млрд символов

Для объявления переменной используется оператор ***Dim***, который имеет следующий синтаксис:

Dim Имя As ТипДанных

Для объявления нескольких переменных каждую переменную можно объявлять отдельной командой с ключевым словом *Dim*. Также можно все переменные объявить одной инструкцией (одной строкой), но после имени каждой переменной необходимо указывать ее тип. Например:

Dim i As Integer, x As Double, y As Double

Формально при написании своих программ переменные можно не объявлять, и тогда по умолчанию ***VBA*** применит для переменных тип ***Variant***. Но, просмотрев таблицу 1, можно увидеть, что ни один из типов данных не требует 16 байт для сохранения значений переменных. То есть необходимо экономить ресурсы памяти компьютера. Кроме того, не объявляя тип переменных, разработчик рискует совершить грамматические ошибки, связанные с написанием имен переменных буквами на различных языках, а также получить результат выполнения программы, далёкий от ожидаемого по причине некорректного автоматического преобразования типов данных.

Константы, в отличие от переменных, не могут изменять свои значения. Использование констант делает программы легче читаемыми и позволяет проще вносить исправления – отпадает необходимость многократно исправлять значения по тексту программы, т. к. достаточно ввести новое значение при определении константы.

Синтаксис:

Const ИмяКонстанты [As Тип] = Выражение

Например:

Const ПроцентнаяСтавка As Single =0,2

Для обязательного описания всех переменных необходимо поместить в начале модуля инструкцию ***Option Explicit***.

Ввод и вывод информации в визуальных средах, как правило, осуществляется с помощью диалоговых окон. В проектах ***VBA*** наиболее часто встречаются две разновидности диалоговых окон: окна сообщений и окна ввода. Они встроены в VBA, и если их возможностей достаточно, то можно обойтись без проектирования диалоговых окон. Окно сообщений (***MsgBox***) выводит простейшие сообщения для пользователя, а окно ввода (***InputBox***) обеспечивает ввод информации.

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

Синтаксис (в упрощенном варианте):

InputBox (*prompt [, title] [, default]*)

Аргументы:

Prompt – строковое выражение, отображаемое как сообщение в диалоговом окне;

Title – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения;

Default – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым.

Например:

Пароль = InputBox(*«Введите пароль:», «Вход в систему»*)

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

Синтаксис (в упрощенном варианте):

MsgBox prompt

Аргумент ***prompt*** – строковое выражение, отображаемое как сообщение в диалоговом окне. Допускается объединение нескольких строк в одну непосредственно в строке оператора ***MsgBox*** с помощью символа конкатенации (амперсанд) «&». Например:

MsgBox “Строка1” & Переменная & “Строка 2”

Взаимодействие с таблицами MS Excel .

Основное назначение большинства прикладных программ – это обработка данных, вводимых пользователем. При программировании на ***VBA*** данные, введенные пользователем, можно хранить на рабочих листах книги ***MS Excel.*** На них же можно записывать результаты обработки данных.

Для доступа к листам активной рабочей книги применяется коллекция объектов ***Sheets*** (в случае работы с несколькими рабочими книгами в программном коде также необходимо явно указывать соответствующую

рабочую книгу). С помощью данного оператора можно обратиться к нужному рабочему листу:

Sheets(*«Наименование листа»*)

Данный оператор возвращает объект типа ***Worksheet*** – рабочий лист, который, в свою очередь, имеет свойства ***Cells*** (ячейки) и ***Range*** (область) типа ***Range*** (объект область), позволяющие выбрать конкретные ячейки рабочего листа:

Cells(*номер_строки, номер_столбца*)
Range(«*адрес ячейки*»)
Range(«*диапазон ячеек*»)

Например, для ввода числа 34 в ячейку «***A7***» листа «***Лист1***» можно использовать команду

Sheets(«*Лист1*»).*Cells*(*7*, *1*).*Value = 34*;

для считывания значения ячейки «***A7***» листа «***Лист1***» – команду

x = Sheets(«*Лист1*»).*Range*(«*A7*»).*Value.*

С помощью свойства ***Range*** также можно обратиться к диапазону ячеек. Например, для заполнения диапазона ячеек «***A1:E10***» цифрой 9 применяется следующая команда:

Sheets(«*Лист1*»).*Range*(«*A1:E10*»).*Value = 9.*

Контрольные вопросы

1 Опишите типы данных VBA.
2 Что такое константа? Приведите примеры констант различных типов.
3 Покажите способы объявления и инициализации переменных и констант.
4 Какие существуют варианты ввода и вывода данных в VBA?
5 Каким оператором в VBA организуется ввод/вывод данных с листа или на лист *Excel*?
6 Как оформляется оператор ввода данных с экрана?
7 Как оформляется оператор вывода данных на экран?
8 Что можно указывать в качестве элементов списка ввода?
9 Какой символ используется для разделения элементов списка вывода?
10 Что будет выведено на экран, если в списке вывода записано: число, имя величины, текст в кавычках, арифметическое выражение?

Лабораторная работа № 2. Программа с ветвлениями

Цель работы: изучить операторы ветвления, разработать программу с двумя вычислительными ветвями.

Задание

Разработать программу, реализующую следующие действия.

1 Запросить у пользователя ввод числа.

2 Сравнить введенное число с другим числом, заданным константой, например 20.

3 По результатам сравнения вывести соответствующее сообщение:

«25>20» или «15<20»,

где 25, 15 – введенные пользователем числа.

Методические указания

Алгоритм называется разветвляющимся, если последовательность выполнения его шагов изменяется в зависимости от выполнения некоторых условий. Условие – это логическое выражение, которое может принимать одно из двух значений: «*ДА*» – если условие верно (истинно, *TRUE*); «*НЕТ*» – если условие неверно (ложно, *FALSE*).

Схема алгоритма конструкции условного оператора ***If*** представлена на рисунке 2.

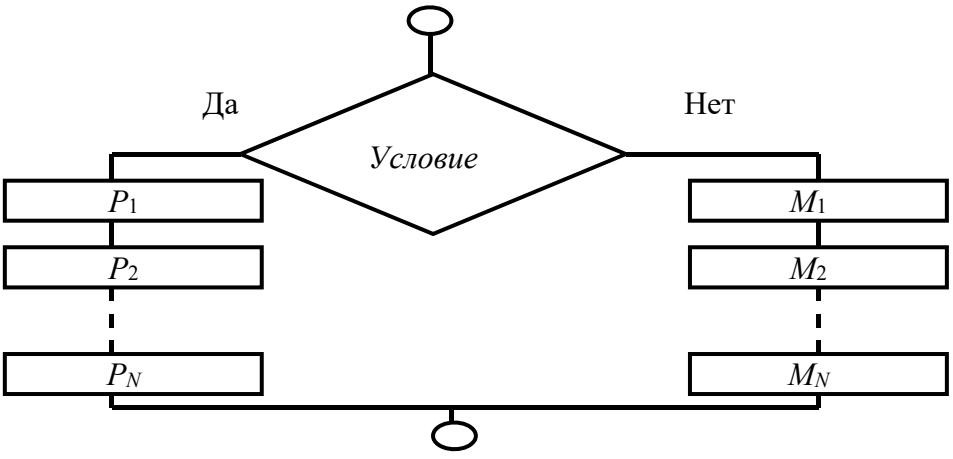


Рисунок 2 – Схема алгоритма конструкции условного оператора ***If***

Оператор ***If*** используется для того, чтобы выполнить какой-либо оператор или несколько операторов, если некоторое условие будет истинным.

Синтаксическая конструкция этого оператора может иметь две формы:

1) безальтернативную: ***If*** <*условие*> ***Then*** $P_1 : P_2 : ... : P_N$**;**

2) альтернативную.

Синтаксис альтернативного условного оператора ***If*** имеет два варианта. Синтаксис условного оператора ***If*** в однострочной форме следующий:

If *<условие> Then* P_1 *:*P_2 *: ... :*P_N ***Else*** M_1 *:* M_2 *: ...* M_N

Возможна и другая синтаксическая форма – блочная (структурная):

If *<условие> Then*
    P_1
    …
    P_N
Else
    M_1
    …
    M_N
End If

где ***If***, ***Then***, ***Else***, ***End If*** – зарезервированные слова;
P_1, P_2, P_N, M_1, M_2, M_N – операторы.

Порядок выполнения: вычисляется значение *<условие>*. Оно может принимать значения **TRUE (Истина)** или **FALSE (Ложь)**. Если *<условие>* принимает значение TRUE, то выполняются P_1, P_2, P_N (операторы ветки ***Then***), в противном случае – M_1, M_2, M_N (операторы ветки ***Else***).

Контрольные вопросы

1 В каких случаях в программе используется неполный условный оператор? Как он оформляется?

2 В каких случаях в программе используется полный условный оператор? Как он оформляется? Как он работает (что происходит при его выполнении)?

3 Перечислите операции отношения, используемые в языке, приведите примеры их использования.

4 Перечислите логические операции, используемые в языке, приведите примеры их использования.

5 Как записать сложное логическое выражение и как определить результат его вычисления?

Лабораторная работа № 3. Вложенные ветвления

Цель работы: изучить оператор вложенного ветвления ***If***; разработать программу, реализующую выбор из нескольких альтернатив (более 2).

Задание

Разработать программу, реализующую следующие действия.

1 Запросить у пользователя ввод целого числа от 0 до 100 включительно – оценка по 100-балльной системе.

2 С применением оператора вложенных ветвлений ***If*** осуществить преобразование введенной оценки по 100-балльной системе в 5-балльную по шкале, предварительно созданной на листе *Excel* (таблица 3).

Таблица 3 – Преобразование оценок

Значение оценки по 100-балльной системе	Значение оценки по 5-балльной системе
От 0 до 20	Оценка 1
От 20 до 40	Оценка 2
От 40 до 60	Оценка 3
От 60 до 80	Оценка 4
От 80 до 100 включительно	Оценка 5
Меньше 0 или больше 100	Ошибка ввода данных

3 По результатам вывести сообщение с введенной оценкой по 100-балльной системе и полученной оценкой по 5-балльной системе либо сообщение об ошибке.

Методические указания

Схема алгоритма конструкции оператора многозначных ветвлений ***If*** представлена на рисунке 3.

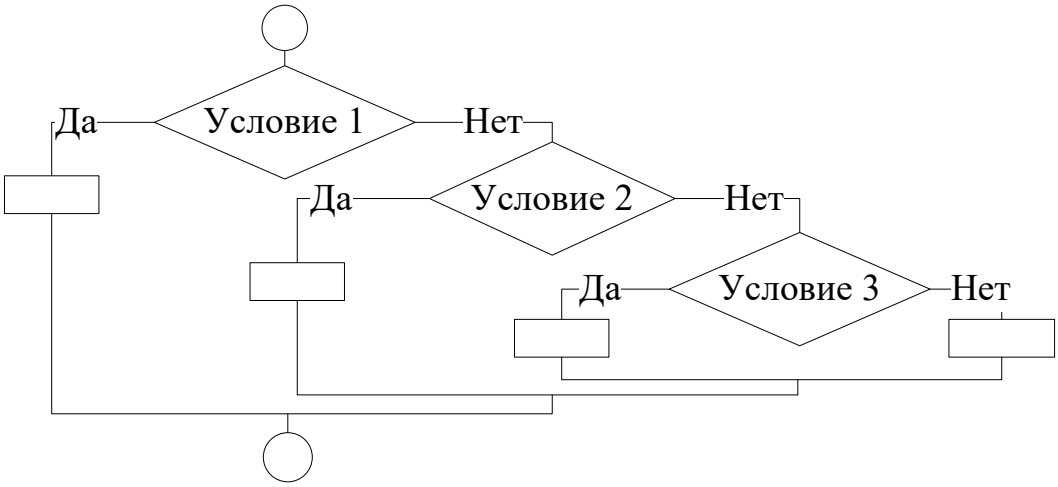


Рисунок 3 – Схема алгоритма конструкции оператора многозначных ветвлений ***If***

Синтаксис оператора многозначных ветвлений ***If*** в блочной (структурной) форме следующий:

If <*Условие 1*>*Then*
    P_1
ElseIf <*Условие 2*>*Then*
    P_2
Else
    P_3
End If

где ***If, Then***, ***ElseIf***, ***Else***, ***End If*** – зарезервированные слова;

P_1, P_2, P_3 – операторы.

Алгоритм работы такой конструкции:

– если условие 1 истинно, то выполняется оператор $\boldsymbol{P_1}$ (или блок операторов), следующий за конструкцией ***Then***, а остальные операторы пропускаются;

– если условие 1 ложно, то оператор $\boldsymbol{P_1}$ пропускается и анализируется условие 2, следующее за ***ElseIf***. Если оно истинно, то выполняется оператор $\boldsymbol{P_2}$ (или блок операторов), следующий за ***Then***, а остальные операторы пропускаются;

– оператор $\boldsymbol{P_3}$ (или блок операторов), следующий за последним ***Else***, выполняется лишь в том случае, если ложны все логические выражения.

Далее представлен пример использования многозначного ветвления, позволяющего определить возможность поступления абитуриента в учреждения образования. На первом шаге происходит ввод пользователем балла, полученного на тестировании. На втором шаге, если полученный балл более 60, приложение выдает сообщение: «Вам можно поступать в ВУЗ». На третьем шаге, если полученный балл более 20, приложение выдает сообщение: «Вам можно поступать в СУЗ». На четвертом шаге, если полученный балл более 5, приложение выдает сообщение: «Вам можно поступать в ПТУ».

```
Sub ball()
Dim ball As Variant
ball = InputBox(«Введите полученный Вами балл на тестировании»)
If ball>=60 Then
        MsgBox(«Вам можно поступать в ВУЗ»)
ElseIf ball>=20 Then
        MsgBox(«Вам можно поступать в СУЗ»)
ElseIf ball>=5 Then
        MsgBox(«Вам можно поступать в ПТУ»)
Else
        MsgBox(«Вам необходимо готовиться к поступлению»)
End If
```

Контрольные вопросы

1 В каких случаях в программе используется оператор вложенного ветвления?

2 Сколько инструкций ***ElseIf*** может быть в блочном варианте оператора ветвления?

3 Нарисуйте алгоритмическую схему выполнения вложенного условного оператора.

4 Опишите синтаксис многозначного ветвления ***If,*** приведите примеры его использования.

5 Приведите примеры использования вложенных операторов ***If*** для организации нескольких вычислительных ветвей.

Лабораторная работа № 4. Оператор выбора

Цель работы: ознакомиться с оператором выбора ***Select Case*** и закрепить полученные знания на практике.

Задание

Выполнить задание лабораторной работы № 3 с применением оператора выбора ***Select Case***.

Методические указания

При наличии большого количества ветвлений конструкция вложенного ветвления ***If*** становится тяжёлой для восприятия. В подобных случаях хорошей альтернативой оператору ***If*** служит оператор выбора ***Select Case***, который позволяет выбрать одно из нескольких возможных продолжений программы.

В то время как ***If ... Then ... Else*** для каждой инструкции ***Else If*** оценивает разные выражения, инструкция ***Select Case*** оценивает выражение только один раз, в начале управляющей структуры. ***Select Case*** выполняет одну из нескольких групп инструкций в зависимости от значения выражения.

Синтаксис:

Select Case *<выражение>*
[Case *<списокВыражений-n>*
    [инструкции-n]] ...
[Case Else
    [инструкции_else]]
End Select

<выражение> − обязательный элемент. Любое числовое выражение или строковое выражение.

<списокВыражений-n> – обязательный при наличии предложения ***Case***.

Список с разделителями, состоящий из одной или нескольких форм следующего вида:

<инструкции-n> − необязательный. Одна или несколько инструкций, выполняемых в том случае, если выражение совпадает с любым компонентом списка <**списокВыражений-n**>;

<инструкции_else> − необязательный. Одна или несколько инструкций, выполняемых в том случае, если выражение не совпадает ни с одним из предложений ***Case***.

Синтаксис оператора ***Select Case*** также может содержать следующие элементы:

– **выражение *To* выражение**;

– ***Is*** оператор Сравнения **выражение**.

Ключевое слово ***To*** задает диапазон значений. При использовании ключевого слова ***To*** перед ним должно находиться меньшее значение. Ключевое слово ***Is*** с операторами сравнения (кроме ***Is*** и ***Like***) задает диапазон значений. Если ключевое слово ***Is*** не указано, оно вставляется по умолчанию.

Если выражение совпадает с любым выражением из *спискаВыражений* в предложении ***Case***, выполняются все инструкции, следующие за данным предложением ***Case*** до следующего предложения ***Case***, или, для последнего предложения, до инструкции ***End Select***. Затем управление передается инструкции, следующей за ***End Select***. Если выражение совпадает с выражениями из списка в нескольких предложениях ***Case***, выполняется только первый подходящий набор инструкций.

Предложение ***Case Else*** задает список **инструкции_else**, которые будут выполнены, если не обнаружено ни одно совпадение выражения и компонента **список Выражений** ни в одном из остальных предложений ***Case***. Хотя данное предложение не является обязательным, рекомендуется помещать предложение ***Case Else*** в блок ***Select Case***, чтобы предусмотреть неожиданные значения выражения. Если ни в одном предложении ***Case*** **списокВыражений** не содержит компонента, отвечающего аргументу выражения, и отсутствует инструкция ***Case Else***, выполнение продолжается с инструкции, следующей за инструкцией ***End Select***. Пример использования оператора ***Select Case*** представлен в таблице 4.

Таблица 4 – Синтаксис и пример использования оператора ***Select Case***

Синтаксис оператора ***Select Case***	Пример использования оператора ***Select Case***
Select Case *КлючВыбора* ***Case*** *Is выражение* *оператор* ***Case*** *диапазон значений* *оператор* ***Case*** *диапазон значений* *оператор* ***Case*** *диапазон значений* *оператор* ***Case Else*** *оператор* ***End Select***	***Select Case*** *vozrast* ***Case*** *Is <=7* *Msgbox «Ты дошкольник»* ***Case*** *8 to 16* *Msgbox «Ты учишься в школе»* ***Case*** *17 to 30* *Msgbox «Тебе пора заняться делом»* ***Case*** *31 to 60* *Msgbox «Кто не работает, тот не ест»* ***Case Else*** *Msgbox «Вы заслужили отдых»* ***End Select***

Если значение переменной ***vozrast*** меньше или равно **7**, отображается сообщение «*Ты дошкольник*»; если значение находится в диапазоне от 8 до 16 – сообщение «*Ты учишься в школе*»; если в диапазоне от 17 до 30 – сообщение «*Тебе пора заняться делом*»; если в диапазоне от 31 до 60 – сообщение «*Кто не*

работает, *тот не ест*». Если значение возраста не равно ни одному из предложенных диапазонов значений, выводится сообщение «*Вы заслужили отдых*».

Из представленного в таблице 4 примера видно, что код этой процедуры более прост для восприятия, чем вложенные ветвления ***If***.

Контрольные вопросы

1 В каких случаях в программе используется оператор выбора?

2 В чем преимущество оператора выбора варианта перед многовариантным оператором ветвления?

Лабораторная работа № 5. Оператор цикла с параметром

Цель работы: изучить операторы цикла ***For,*** разработать программу с использованием операторов цикла.

Задание

Разработать программу, реализующую следующие действия.

1 Запросить у пользователя ввод целого числа больше 0.

2 Определить произведение последовательности чисел от 1 до *n* включительно (*n*! – «*n* факториал»).

3 Результат вывести в виде сообщения. Пример:

«Факториал 10! = 1 · 2 · 3 · 4 · 5 ·6 · 7 · 8 · 9 · 10 = 3 628 800».

Методические указания

Часто в программах необходимо реализовать определённые операторы несколько раз. В этих случаях организуют циклические вычисления. Алгоритм называется ***циклическим***, если определенная последовательность шагов выполняется несколько раз в зависимости от заданных условий. Циклические алгоритмы могут быть осуществлены с применением следующих операторов цикла: ***For … Next***, ***While … Wend***, ***Do … Loop,*** которые позволяют повторить группу операторов или один оператор заданное количество раз.

Общий вид алгоритма конструкции оператора цикла ***For … Next*** представлен на рисунке 4.

For i = N_1 To N_2 [Step h]

P_1

…

[Exit For]

P_N

Тело цикла

Next i

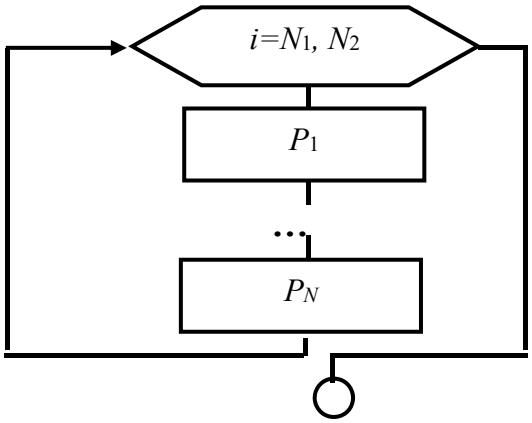


Рисунок 4 – Схема алгоритма конструкции оператора цикла *For* **…** *Next*

Синтаксис конструкции оператора цикла ***For … Next*** следующий:

For (для), ***To*** (до), ***Step*** (шаг), ***Exit For*** (выход из ***For***), ***Next*** (следующий) – служебные слова ***VBA***, а P_1, P_N – операторы. ***Step*** является необязательным параметром. Если он опущен в программе, то значение параметра ***i*** увеличивается на **1**. Параметр ***Step*** может быть любым действительным числом, как целым, так и дробным, как положительным, так и отрицательным. Оператор ***Exit For*** позволяет выйти из цикла ***For … Next*** до его завершения. Тем самым программа сможет среагировать на определённое событие, не выполняя цикл заданное число раз.

Контрольные вопросы

1 Может ли тело оператора цикла с параметром не выполниться ни разу?

2 Как должен быть оформлен оператор цикла с параметром, чтобы тело цикла выполнялось при уменьшающихся значениях параметра цикла?

3 Чему равно количество повторений тела оператора цикла с параметром, если параметр цикла принимает:

а) все целые значения от 1 до 10;

б) все значения от 10 до 100 с шагом 7;

в) все значения от *a* до *b* с шагом *step*.

4 Можно ли в теле оператора цикла использовать условный оператор?

5 Какие Вы знаете операторы для принудительного (преждевременного) выхода из оператора цикла?

Лабораторная работа № 6. Цикл с предусловием или постусловием

Цель работы: научиться применять операторы цикла с предусловием и постусловием.

Задание

Разработать программу, реализующую следующие действия.

1 В цикле реализовать запрос пароля: «Введите пароль:».

2 Сравнить введенный пользователем пароль с заданным.

3 Цикл должен выполняться до тех пор, пока пароль не совпадет.

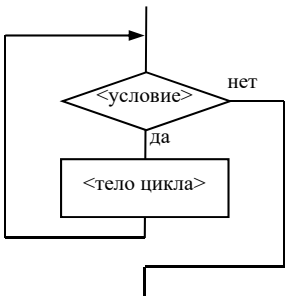
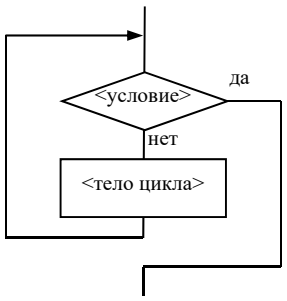
4 Если пароль совпадает, цикл завершается и программа выдает сообщение: «Пароль принят!».

Методические указания

Циклы данного вида используются, когда заранее неизвестно, сколько раз будет выполняться тело цикла.

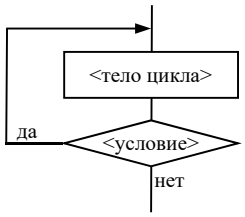
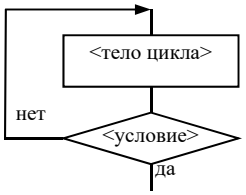
Циклы с предусловием (Do While … Loop, Do Until … Loop) представлены в таблице 5, а операторы циклов с постусловием (Do … Loop While, Do … Loop Until) – в таблице 6.

Таблица 5 – Циклы с предусловием

Вид цикла с предусловием	Do While … Loop	Do Until … Loop
Синтаксис	***Do While*** <*условие*> <*тело цикла*> [***Exit Do***] **…** ***Loop***	***Do Unlil*** <*условие*> <*тело цикла*> [***Exit Do***] **…** ***Loop***
Порядок выполнения	**<***Тело цикла***>** будет выполняться в том случае, когда <*условие*> имеет значение Истина (TRUE) (цикл продолжается при истинном значении <*условия***>**). Если <*условие*> ложно (FALSE), то выполняются операторы, стоящие за циклом. В первом случае есть возможность досрочного выхода из цикла (это реализовано через ***Exit Do*****)**	**<***Тело цикла***>** выполняется до тех пор, пока <*условие*> не примет значение Истина (цикл продолжается при ложном значении <*условия*>). Есть возможность досрочного выхода из цикла (это реализовано через ***Exit Do***)
Вид цикла с предусловием	Do While … Loop	Do Until … Loop
Изображение в схемах		

Отличие циклов с предусловием от циклов с постусловием заключается в том, что тело цикла первых может не выполниться ни разу, в то время как тело цикла с постусловием всегда выполнится хотя бы один раз.

Таблица 6 – Циклы с постусловием

Вид цикла с постусловием	Do While … Loop	Do Until … Loop
Синтаксис	***Do*** *<тело цикла>* [***Exit Do***] **…** ***Loop While*** *<условие>*	***Do*** *<тело цикла>* [***Exit Do***] **…** ***Loop Until*** **<***условие>*
Порядок выполнения	*<Тело цикла>* будет выполняться в том случае, когда *<условие>* имеет значение Истина (цикл продолжается при истинном значении *<условия>*). Если *<условие>* ложно, то выполняются операторы, стоящие за циклом. Предоставлена возможность досрочного выхода из цикла (это реализовано через ***Exit Do***)	**<***Тело цикла>* выполняется до тех пор, пока *<условие>* не примет значение Истина (цикл продолжается при ложном значении *<условия>*). Есть возможность досрочного выхода из цикла (это реализовано через ***Exit Do***)
Изображение в схемах		

Пример – Организовать ввод последовательности целых чисел, пока их сумма не превысит целого числа *m*. Вывести количество введенных чисел. Текст программы с пояснениями представлен в таблице 7.

Таблица 7 – Программа по вводу последовательности целых чисел

Текст программы	Описание
Public Sub prog1() *Dim x As Integer, m As Integer* *Dim s As Integer, i As Integer* *m=InputBox*(*«Введите число m»*) *i = 1* *s =InputBox*(*«Введите 1 число»*) *Do While s <= m* *i = i + 1* *x=InputBox*(*«Введите» & i_* *& «число»*) *s = s + x* *Loop* *MsgBox* (*«Количество чисел» & i*) *End Sub*	 Ввод предельного числа Номер вводимого числа последовательности Ввод первого числа последовательности Цикл с предусловием: тело цикла выполняется, пока условие s<=m имеет значение Истина (TRUE) Тело цикла: увеличение номера на 1 ввод очередного (*i*-го) значения добавление введенного значения к предыдущему значению суммы Конец тела цикла Вывод значения переменной *i*

Контрольные вопросы

1 В каких случаях используются операторы цикла с условием?

2 В чём основное отличие между циклами с предусловием и с постусловием?

3 В каких случаях целесообразно использовать циклы с предусловием, циклы с постусловием и циклы по счётчику?

4 Может ли тело оператора цикла с предусловием:

а) не выполниться ни разу;

б) выполняться бесконечное число раз (или до тех пор, когда пользователь прервет его выполнение)?

Лабораторная работа № 7. Массивы

Цель работы: изучить одномерные массивы, разработать программу с вводом, выводом и обработкой одномерных массивов.

Задание

Разработать программу, реализующую следующие действия.

1 Объявить одномерный массив размерностью 10 элементов целочисленного типа.

2 Инициализировать все элементы массива произвольными числами: $A(1) = 5$ и т. д.

3 Реализовать расчет среднего арифметического всех чисел, содержащихся в массиве, с помощью оператора цикла.

4 Вывести результат расчета в виде текстового сообщения:

«Среднее арифметическое: ____».

Методические указания

Массив – совокупность однотипных элементов данных (чисел, логических данных, символов), которой при обработке присвоено определенное имя. Совокупность представлена набором переменных с одним именем и с разными индексами. Массивы бывают разной размерности: одномерные, двумерные, трехмерные, …, *n*-мерные.

Массивы бывают статические и динамические. Статическими называются массивы, количество элементов в которых заранее известно и не изменяется в ходе выполнения программы. Динамические массивы – массивы, в которых либо не известно начальное количество элементов, либо размерность массива (количество элементов) изменяется при выполнении программы.

Описание массивов:

1) **одномерный статический массив**

Dim <*имя массива*> **(**<*начальное значение индекса*> **To** <*конечное значение индекса*>**)** [**As** <*тип элементов массива*>]

или

Dim <*имя массива*> **(**<*количество элементов массива*>**)** [**As** <*тип элементов массива*>];

2) **двумерный статический массив**

Dim <*имя массива*> **(**<*начальное значение индекса по строкам*> **To** <*конечное значение индекса по строкам* >**,** < *начальное значение индекса по столбцам*> **To** < *конечное значение индекса по столбцам*>**)** [**As** <*тип элементов массива*>]

или

Dim <имя массива> **(**<количество строк>**,** <количество столбцов>**)** [**As** <тип элементов массива>].

Первый способ отличается от второго тем, что в первом случае указывается индекс первого и последнего элементов, во втором же – только количество элементов, нумерация которых может начинаться как с 0, так и с 1. Это зависит от опции **Base** (задает базовый индекс). Если опция не указана, то нумерация элементов массива начинается с нуля. Для изменения базового индекса в начале листа модуля необходимо написать **Option Base 1**.

Пример*:*

а) Dim A(1 To 10) As Integer – массив А состоит из 10 элементов целого типа, индексы которых 1, 2, **…**, 10;

б) Dim A(10) As Integer – массив состоит из 10 значений целого типа. Индексация зависит от опции **Base**. Если опция не указана, то номера элементов − от 0 до 9, если же указана (т. е. вначале модуля записано **Option Base 1**), то номера элементов изменяются от 1 до 10;

3) **динамический массив**

Dim <имя массива> **( )** [**As**<тип элементов массива>].

После определения количества элементов массива выполняется его переопределение:

ReDim <имя массива> **(**<задается размерность массива (одномерного/двумерного >**)**.

Пример

Dim A( ) *As Single* – динамический массив А вещественных элементов

n=7

ReDim A(1 *Ton*) – переопределение одномерного массива из *n* значений

ReDim A(5,*n*) – переопределение двумерного динамического массива, состоящего из пяти сток и *n* столбцов (начало индексации элементов определяется по опции **Base**).

Обращение к элементу массива осуществляется следующим образом: указывается имя массива, а затем в круглых скобках − номер элемента в массиве. Если массив двумерный – указывается вначале номер строки, затем через запятую номер столбца.

Примеры*:*

Dim A(*12*) *As Integer* ‘ одномерный массив из 12 элементов типа *Integer*

Dim A(*1 To 12*) *As Integer*

Dim B(*3, 3*) *As Single* ‘ двумерный массив элементов 3 х 3 (матрица) типа Single

Dim B(*1 To 3, 1 To 3*) *As Single*

Причем по умолчанию первый элемент массива *A* будет *A*(0), а последний − *A*(11). В этом случае говорят, что 0 – базовый индекс. Можно изменить базовый индекс, написав в начале листа модуля инструкцию ***Option Base 1***. После этого индексы массивов *A* и *B* будут начинаться с единицы.

Массив в программе определяется поэлементно.

Удобным способом определения одномерных массивов является функция ***Array***, преобразующая список элементов, разделенных запятыми, в вектор из этих значений и присваивающая их переменной тип ***Variant***.

Контрольные вопросы

1 Дайте определение и приведите примеры объявления и инициализации одномерных массивов.

2 Какое значение нижнего индекса элемента массива принято в VBA по умолчанию? Каким образом можно для него задать значение 1?

3 Как установить или изменить размерность многомерного динамического массива?

4 Как изменить размерность динамического массива без потери имеющихся значений его элементов?

5 Сколько индексов характеризуют конкретный элемент двумерного массива?

6 Приведите алгоритм и пример программы накопления суммы и произведения элементов массива.

7 Приведите алгоритм и пример программы линейного поиска элемента в массиве.

8 Приведите алгоритм и пример программы поиска минимального элемента массива.

Лабораторная работа № 8. Подпрограммы и их применение

Цель работы: изучить механизмы работы функций и процедур, разработать программу с использованием функций.

Задание

1 Создать новую процедуру и задать в ней вызов любой из процедур, разработанных в рамках лабораторных работ № 1–7.

2 Разработать функцию, осуществляющую расчет стоимости товара с НДС. В качестве аргументов данная функция должна принимать стоимость товара без НДС и ставку НДС. Возвращаемое функцией значение – стоимость товара с НДС.

Создать новую процедуру и задать в ней вызов разработанной функции с целью расчета стоимости товара с НДС, указав при вызове необходимые параметры (стоимость товара без НДС и ставку НДС). Полученную стоимость вывести на экран с помощью сообщения: «*Стоимость товара с НДС: ____*».

Методические указания

В программировании сложный код программы разбивают на подпрограммы. *Подпрограмма* – это группа операторов, выполняющих законченное действие. *Основная программа* – программа, реализующая основной алгоритм решения задачи и содержащая в себе обращения к подпрограммам (вызов подпрограмм). В точке вызова функции выполнение программы переходит к подпрограмме и, выполнив все действия подпрограммы, возвращается в основную программу. В подпрограмме могут быть объявлены собственные переменные, а также параметры подпрограммы для обмена значений с основной программой. В VBA существуют два типа подпрограмм: подпрограммы-функции и подпрограммы-процедуры. Функция, в отличие от процедуры, возвращает значение и может входить в состав выражений. Сравнительная характеристика процедур и функций представлена в таблице 8.

<*Список параметров*> отличается от <*списка аргументов*> тем, что первый указывается при описании подпрограммы, второй – при ее вызове в основной программе.

Таблица 8 – Сравнительная характеристика процедур и функций

	Подпрограмма-процедура	Подпрограмма-функция
Описание	**Sub** <имя процедуры> **(**[<список параметров>]**)** <операторы> **[Exit Sub]** <операторы> **End Sub**	**Function <**имя функции>[(<список параметров>]) [As<тип функции>] <операторы> **[Exit Function]** <операторы> **<**имя функции> = **<**выражение> **End Function**
Вызов в основной программе	1) <имя процедуры> <список аргументов>; 2) **Call** <имя процедуры> [**(**<список аргументов>**)**]	<имя функции> (<список аргументов>)

<*Тип*> позволяет явно задать тип передаваемых значений. Если тип опущен, то по умолчанию принимает значение Variant.

Ключевое слово ***ByVal*** (передача по значению) используется в тех случаях, когда желают, чтобы изменение параметров внутри процедуры не приводило к изменению соответствующих аргументов процедуры в основной программе. Использование ***ByRef*** (передача по ссылке) или, если ключевое слово опущено,

означает, что при изменении параметров внутри процедуры происходит изменение соответствующих параметров в основной программе.

<*Список аргументов*> перечисляется через запятую. Количество и типы параметров и аргументов должны соответствовать. Аргументы, соответствующие параметрам с ключевым словом ***ByRef*** (или по умолчанию), должны быть переменными.

Для выхода из подпрограммы и возврата в основную программу, опуская оставшиеся операторы, используется ***Exit Sub*** (в процедурах) и ***Exit Function*** (в функциях).

Пример 1 – Вычислить сумму членов ряда $\sum_{i=1}^{n} \frac{1}{i!}$, где $i!$ – факториал числа i (произведение натуральных чисел от 1 до i). Схема алгоритма вычисления приведена на рисунке 5.

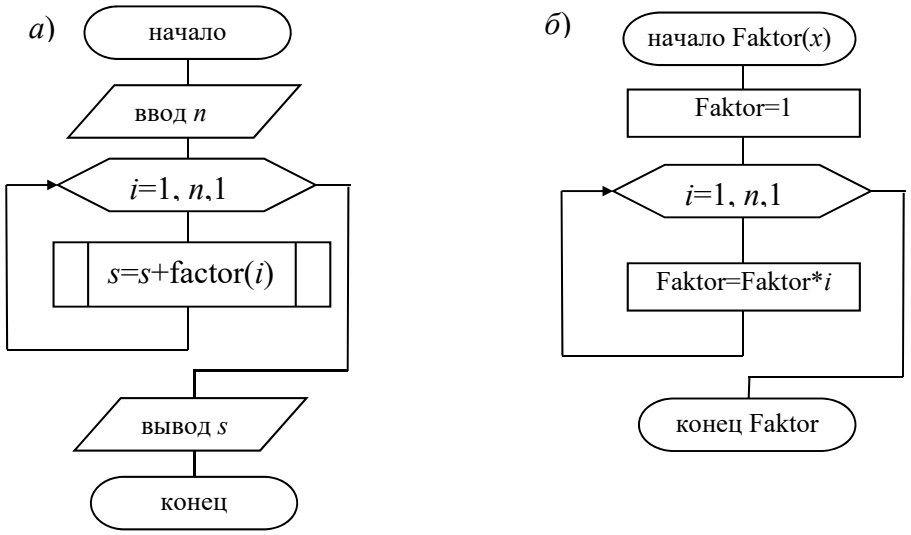


Рисунок 5 – Блок-схема программы prog3 (*а*) и подпрограммы Faktor (*б*)

Текст процедуры вычисления суммы членов ряда с пояснениями представлен в таблице 9, текст функции в таблице 10.

Таблица 9 – Процедура вычисления суммы членов ряда

Текст процедуры	Описание
Public Sub prog3() *Dim i As Integer, n As Integer* *Dim s As Double* *n = CInt*(*InputBox*(*«Введите n»*)) *For i = 1 To n* *s = s + 1 / faktor*(*i*) *Next i* *MsgBox s* *End Sub*	При вычислении искомой суммы производится вызов функции factor и передается ее аргумент *i*

Таблица 10 – Функция вычисления факториала

Текст функции	Описание
Public Function faktor(*x As Integer*) *As Long* *faktor = 1* *For i = 1 To x* *faktor = faktor * i* *Next i* *EndFunction*	При описании функции типу ее результата присваивается тип длинный целый (Long), т. к., например, 10!=40320, что выходит за диапазон типа Integer. При выполнении функции результат присваивается ее имени

Пример 2 – Построить график функции $y = \sin(x)$ на отрезке $[a;b]$ с шагом 0,1. Построение графика организовать процедурой. Схема алгоритма вычисления приведена на рисунке 6. Схема макроса не изображается.

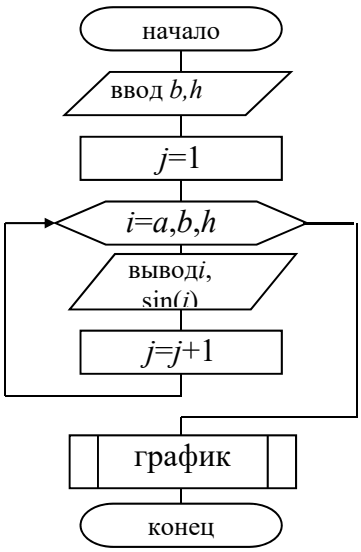


Рисунок 6 − Схема программы prog4

Текст процедуры построения графика функции с пояснениями представлен в таблице 11. Текст процедуры вычисления функции и вызова процедуры построения графика с пояснениями представлен в таблице 12.

Таблица 11 – Программа процедуры построения графика функции

Текст процедуры	Описание
Sub график() *n=Application.CountA*(*Worksheets*(*1*).*Range*(*«A:A»*)) *Range*(*«A1:B»&CStr*(*n*)).*Select* *Charts.Add* *ActiveChart.ChartType = xlXYScatterSmoothNo_* *Markers* *ActiveChart.SetSourceData* *End Sub*	При создании процедуры графика вначале был создан макрос в *Excel*, а затем макрос отредактирован. В частности, первая команда в данной процедуре определяет количество заполненных строк на листе в столбце А и это значение присваивается переменной *n*

Таблица 12 – Программа вычисления функции и вызова процедуры построения графика

Текст процедуры	Описание
Public Sub prog4() *Dim a As Double b As Double, h As Double* *Worksheets*(*1*).*Range*(*«A:B»*).*Select* *Selection.Clear* *a = CDbl* (*InputBox*(*«Введите b»*)) *b = CDbl* (*InputBox*(*«Введите b»*)) *j = 1* *For i = 1 To b Step 0.1* *Worksheets*(*1*).*Range*(*«A»*& *j*) = *i* *Worksheets*(*1*).*Range*(*«B»*& *j*) = *Sin*(*i*) *j = j + 1* *Next i* *график* *End Sub*	 Выделение двух столбцов первого листа Очистка от данных выделенного диапазона *j* используется для задания номера строки при выводе данных В цикле For … Next выводятся на лист *Excel* данные для построения диаграммы Вызывается процедура − график

Контрольные вопросы

1 В чем различие между функцией и процедурой?

2 Понятие и синтаксис процедур и функций (понятие, объявление, определение, вызов).

3 Чем отличаются локальные переменные от глобальных переменных?

4 В какой последовательности должны располагаться имена параметров в операторах вызова функции?

5 Как объявить тип данных для аргументов функции?

6 Могут ли в одной программе процедура и функция иметь одно и то же имя?

7 Как организовать преждевременный выход из функции?

8 Что такое функция? Приведите синтаксис функций (объявление, определение, вызов).

9 Для глобальных, локальных и статических локальных переменных перечислите: место их объявления в программе, место хранения в оперативной памяти, время появления и исчезновения, а также область видимости в программе.

10 Опишите и приведите пример передачи параметров функциям.

11 Приведите пример использования рекурсивной функции.

12 Приведите пример использования аргументов функции по умолчанию.

Лабораторная работа № 9. Линейная программа с использованием пользовательской формы

Цель работы: получить навыки разработки приложений с графическим интерфейсом, включающим окна ввода, командные кнопки и надписи.

Задание

Разработать программу (пользовательский интерфейс ***UserForm***) в соответствии с вариантом (таблица 13).

Таблица 13 – Варианты задания

Вариант	Задание
1	Расчет площади круга
2	Расчет площади квадрата
3	Расчет площади ромба
4	Расчет площади треугольника
5	Расчет площади прямоугольника
6	Расчет площади трапеции
7	Расчет площади параллелепипеда
8	Расчет объема куба
9	Расчет объема цилиндра
10	Расчет объема шара
11	Расчет объема конуса

Ввод и вывод данных обеспечьте путем использования элементов управления ***TextBox*** (текстовое поле) с соответствующими надписями ***Label***.

Поместите на форму 2 элемента управления ***CommandButton*** **(**кнопки) «Расчет» и «Отмена».

Все расчеты производятся при нажатии кнопки «Расчет». Данная кнопка должна срабатывать при нажатии на клавишу <*Enter*> (независимо от того, какой элемент управления является активным в момент нажатия данной клавиши).

Кнопка «Отмена» должна закрывать текущую форму и срабатывать при нажатии на клавишу <*Esc*>.

Перед проведением расчета программа должна выполнить проверку корректности ввода данных. Результат расчета выводится в соответствующее текстовое поле ***TextBox*** и на страницу *Excel*.

В заголовке формы должно выводиться Ваше задание, например, «Расчет площади круга».

Методические указания

По своей сути форма (или пользовательская форма) представляет собой диалоговое окно, в котором можно размещать различные элементы управления. В приложении может быть как одна, так и несколько форм. Новая форма добавляется в проект выбором команды **Вставка (Insert)** →***UserForm***.

В VBA имеется обширный набор встроенных элементов управления. Используя этот набор и редактор форм, нетрудно создать любой пользовательский интерфейс, который будет удовлетворять всем требованиям, предъявляемым к интерфейсу в среде Windows. Элементы управления являются объектами. Как любые объекты, они обладают свойствами, методами и событиями. Элементы управления создаются при помощи Панели элементов, которая отображается на экране, либо выбором команды **Вид (*View*) →Панель элементов (*Toolbox*)**, либо нажатием кнопки  панели инструментов **Standard.** На этой панели представлены кнопки, позволяющие конструировать элементы управления. Для создания элементов управления служат все кнопки панели инструментов, за исключением кнопки **Выбор объекта**  **.** Щелкнув по кнопке **Выбор объекта**, можно выбрать уже созданный в форме элемент управления для последующего его редактирования (изменения размеров или редактирования).

В таблице 14 представлен список основных элементов управления и соответствующих кнопок панели элементов.

Таблица 14 – Список основных элементов управления

Элемент управления	Имя	Кнопка, его создающая	Элемент управления	Имя	Кнопка, его создающая
Поле	TextBox		Переключатель	OptionButton	
Надпись	Label		Флажок	CheckBox	
Кнопка	CommandButton		Выключатель	ToggleButton	
Список	ListBox		Рамка	Frame	
Поле со списком	ComboBox		Рисунок	Image	
Полоса прокрутки	ScrolBar		Набор страниц	MultiPage	
Счетчик	SpinButton		Набор вкладок	TabStrip	

Для размещения элемента управления на листе или в форме необходимо нажать соответствующую кнопку на панели элементов и с помощью мыши перетащить рамку элемента управления в нужное место. После этого элемент управления можно перемещать, изменять его размеры, копировать в буфер обмена, вставлять из буфера обмена и удалять с формы.

В таблице 15 представлены основные общие свойства элементов управления.

Таблица 15 – Основные общие свойства элементов управления

Свойство	Описание
Caption	Надпись, отображаемая при элементе управления
AutoSize	Допустимые значения: True (устанавливает режим автоматического изменения размеров элемента управления так, чтобы на нем полностью помещался текст, присвоенный свойству Caption) и False (в противном случае)
Visible	Допустимые значения: True (элемент управления отображается во время выполнения программы) и False (в противном случае)
Enabled	Допустимые значения: True (пользователь вручную может управлять элементом управления) и False (в противном случае)
Height и Width	Устанавливают геометрические размеры объекта (высоту и ширину)
Left и Top	Устанавливают координаты верхнего левого угла элемента управления, определяющие его местоположение в форме
BackStyle	Устанавливает тип заднего фона
ControlTipText	Устанавливает текст в окне всплывающей подсказки, связанной с элементом управления. В следующем примере элементу управления CommandButton назначен текст всплывающей подсказки «Это кнопка» CommandButton1.ControlTipText = «Это кнопка»
BackColor, ForeColor и BorderColor	Устанавливают цвет заднего и переднего плана элемента управления, также его границы
BorderStyle	Устанавливает тип границы. Допустимые значения: fmBorderStyleSingle (граница в виде контура); fmBorderStyleNone (граница невидима)
SpecialEffect	Устанавливает тип границы. Отличается от свойства BorderStyle тем, что позволяет установить несколько типов, но одного цвета. BorderStyle позволяет установить только один тип, но различных цветов
Picture (создание картинки)	Внедряет картинку на элемент управления

В таблице 16 представлены наиболее часто используемые свойства элементов управления.

Таблица 16 – Свойства элементов управления

Свойство	Описание
TextBox (поле) используется для ввода текста пользователем или для вывода из него результатов расчетов программ	
Text	Возвращает текст, содержащийся в поле
Multiline	Допустимые значения: True (устанавливает многострочный режим ввода текста в поле) и False (однострочный режим)
WordWrap	Допустимые значения: True (устанавливает режим автоматического переноса) и False (в противном случае)

Окончание таблицы 16

Свойство	Описание
Label (надпись) используется для отображения надписей, например, заголовков элементов управления, не имеющих свойства Caption	
Caption	Возвращает текст, отображаемый в надписи
Multiline	Допустимые значения: True (устанавливает многострочный режим ввода) и False (однострочный режим)
WordWrap	Допустимые значения: True (устанавливает режим автоматического переноса) и False (в противном случае)
CommandButton (кнопка) используется для инициирования выполнения некоторых действий, вызываемых нажатием кнопки, например, запуск программы, печать и т. д.	
Caption	Возвращает текст, отображаемый на кнопке
Default	Задаёт кнопку по умолчанию, т. е. устанавливает ту кнопку, для которой действия, связанные с ней, будут выполняться при нажатии клавиши <Enter>
Cancel	Допустимые значения: True (устанавливаются отменяющие функции для кнопки, т. е. нажатие клавиши <Esc> приводит к тем же результатам, что и нажатие кнопки) и False (в противном случае)
Accelerator	Назначает клавишу, при нажатии на которую одновременно с клавишей <Alt> происходит запуск действий, связанных с кнопкой. Например, CommandButton1.Accelerator=«C»

Пример – В качестве примера работы с формой сконструируем простое приложение, вычисляющее значение функции, например, Cos(x).

В таблице 17 приведено описание создаваемой формы.

Таблица 17 – Описание создаваемой формы

Элемент управления	Предназначение
CommandButton1 (кнопка)	При нажатии на кнопку запускается процедура обработки события (Private Sub CommandButton1_Click()), которая считывает значение аргумента из поля TextBox1. Проверяется, введено ли в это поле число. Если введено не число, то на экране отображается соответствующее сообщение, прерывается выполнение процедуры, и фокус (курсор) устанавливается на поле TextBox1, предлагая исправить вводимые данные. Если введено число, то находится значение функции при введенном значении аргумента, результат выводится в TextBox2
Label1 (надпись)	Пояснительная надпись для поля ввода
TextBox1 (поле)	Поле для ввода пользователем значения аргумента
TextBox2 (поле)	В это поле будет выводиться значение функции. Поле сделаем недоступным для пользователя, т. е. пользователь не сможет ни ввести, ни скорректировать данные в этом поле

Перейдем в VBA и, выполнив команду ***Insert*** **(Вставка)** →***UserForm***, добавим в проект форму. Расположим на форме следующие элементы управления: ***Label***, ***TextBox***, ***CommandButton*** (рисунок 7).

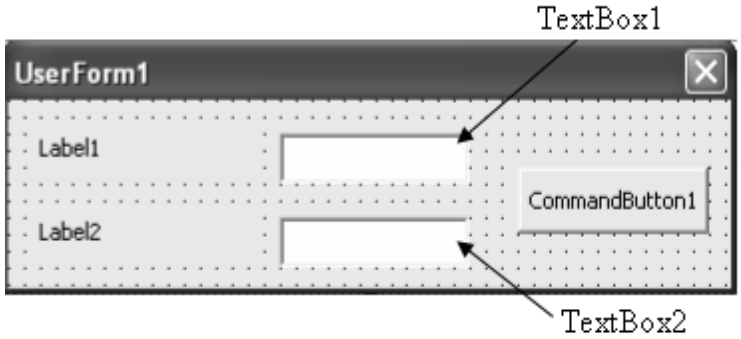


Рисунок 7 – Вид формы в режиме конструктора

Форма создана, функция каждого элемента управления известна. Для написания кода программы, связанного с пользовательской формой, достаточно дважды щелкнуть, например, кнопку ***CommandButton*1**. Откроется редактор кода на листе модуля UserForm1. Более того, он откроется на том месте, где программируются действия, связанные с элементом управления, который был дважды нажат. Если код еще не набран, то при открытии редактора кода появятся инструкции заголовка и окончания процедуры, которая будет связана с элементом управления. Редактор кода представлен в таблице 18.

Таблица 18 – Пример программы с применением элементов управления

Текст программы	Описание
Private Sub CommandButton1_Click() *If Not IsNumeric*(*TextBox1.Text*) *Then* *MsgBox «Аргумент должен быть числом», _vbExclamation* *TextBox1.SetFocus* *Exit Sub* *End If* *x = CDbl*(*TextBox1.Text*) *y = Cos*(*x*) *TextBox2.Text = CStr*(*y*) *End Sub*	 Проверка является ли введённое значение числом Вывод окна сообщения Фокус (курсор) устанавливается на поле *TextBox*1 Досрочный выход из процедуры При считывании числа из поля ввода при помощи функции CDbl строковый тип, возвращаемый свойством Text, преобразуется в числовой. Чтобы вывести результат в поле, переводим число в строковый формат при помощи функции CStr
Private Sub UserForm_Initialize() *UserForm1.Caption = «Значение функции Cos*(*x*)*»* *Label1.Caption = «Аргумент»* *Label2.Caption = «Значение функции»* *CommandButton1.Caption = «OK»* *TextBox2.Enabled = False* *End Sub*	Процедура UserForm_Initialize конструирует форму до ее загрузки Инструкция устанавливает текст, отображаемый в строке заголовка формы Инструкции задают видимые надписи для объектов Инструкция делает *TextBox*2 недоступным для пользователя

После конструирования формы и написания кода в модуле формы выберем команду **Run→RunSub/UserForm**, либо нажмем клавишу **<F5>**, либо кнопку панели инструментов Standard, и форма отобразится поверх активного рабочего листа *Excel*. Введем значение аргумента и нажмем кнопку OK. Вид полученной пользовательской формы представлен на рисунке 8.

Рисунок 8 – Вид пользовательской формы

Контрольные вопросы

1 Как создать графический интерфейс своего приложения с помощью VBA?
2 Назовите этапы создания форм.
3 Что такое элемент управления?
4 Какое назначение элементов управления ***Label***, ***TextBox*** и ***CommandButton***?

Лабораторная работа № 10. Программа с элементами управления: зависимый и независимый переключатель

Цель работы: изучить свойства, события и методы элементов управления ***OptionButton***, ***CheckBox***; разработать программу с выбором действий пользователем при помощи переключателей.

Задание

Разработать программу выполнения одной из четырех арифметических операций над двумя числами по выбору пользователя. Исполняемая операция устанавливается и выполняется за счет выбора соответствующего переключателя.

Методические указания

Элемент управления ***OptionButton*** (переключатель) создается с помощью соответствующей кнопки на панели элементов ***Toolbox***. Он позволяет выбрать один из нескольких взаимоисключающих параметров или действий. Переключатели обычно отображаются группами, обеспечивая возможность выбора альтернативного варианта.

Приведем наиболее часто используемые свойства элемента управления ***OptionButton*** (таблица 19).

Таблица 19 – Свойства элемента управления ***OptionButton***

Свойство	Описание
Value	Возвращает True, если переключатель выбран, и False – в противном случае
Enabled	Допустимые значения: True (пользователь может выбрать переключатель) и False (в противном случае)
Visible	Допустимые значения: True (переключатель отображается во время выполнения программы) и False (в противном случае)
Caption	Надпись, отображаемая рядом с переключателем

Элемент управления ***CheckBox*** (флажок) обладает таким же набором свойств, но, в отличие от ***OptionButton***, позволяет выбрать несколько вариантов.

Контрольные вопросы

1 Какое назначение элементов управления ***OptionButton***, ***CheckBox***?

2 Назовите основные свойства элементов управления ***OptionButton***, ***CheckBox***.

3 Что такое «событие»?

4 Как происходит обработка событий ***OptionButton***, ***CheckBox***?

Лабораторная работа № 11. Программа с элементами управления: списки

Цель работы: изучить свойства, события и методы элементов управления ***ListBox***, ***ComboBox***; научиться использовать списки при решении задач.

Задание

Создать простое приложение, состоящее из формы, представленной на рисунке 9. Это приложение позволяет на основе введенных данных (выбора фигуры и ввода соответствующих параметров) произвести расчет площади (варианты 1–10) или объема (варианты 11–20) выбранной фигуры.

Как видно из рисунка, форма содержит:

– текстовую информацию пояснительного характера;

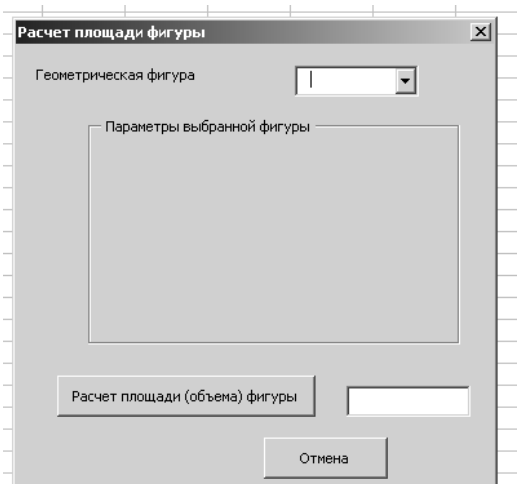
– раскрывающийся список, позволяющий выбрать одну из геометрических фигур (не менее 3);

– текстовые поля для ввода параметров выбранной фигуры;

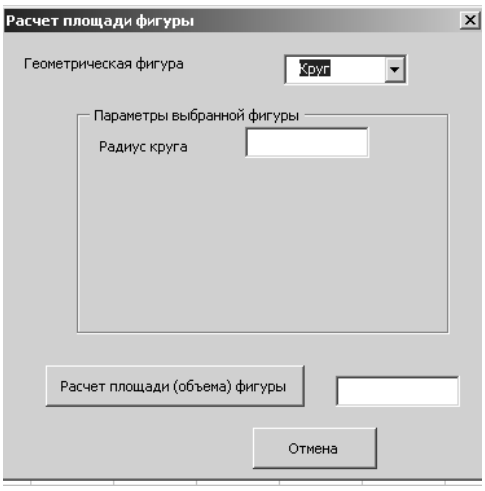
– кнопка, при нажатии на которую будет производиться расчет площади фигуры или объема фигуры;

– поле, размещенное в нижней части формы справа от кнопки, предназначенное для отображения полученного результата.

а)



б)



а – открытие формы; *б* – форма после выбора фигуры

Рисунок 9 – Пример приложения, позволяющего производить вычисления на основе введенных данных (вариант 1)

Методические указания

Элемент управления ***ListBox*** (список) применяется для хранения списка значений. Из списка пользователь может выбрать одно или несколько значений, которые в последующем будут использоваться в тексте программы.

В таблице 20 представлены часто используемые свойства элемента управления ***ListBox***.

Таблица 20 – Свойства элемента управления ***ListBox***

Свойство	Описание
ListIndex	Возвращает номер текущего элемента списка. Нумерация элементов списка начинается с нуля
ListCount	Возвращает число элементов списка
TopIndex	Возвращает элемент списка с наибольшим номером
ColumnCount	Устанавливает число столбцов в списке
TextColumn	Устанавливает столбец в списке, элемент которого возвращается свойством Text
Enabled	Допустимые значения: True (запрещен выбор значения из списка пользователем) и False (в противном случае)
Text	Возвращает выбранный в списке элемент
List	Возвращает элемент списка, стоящий на пересечении указанной строки и столбца. Синтаксис: List(row, column)
RowSource	Устанавливает диапазон, содержащий элементы списка
MultiSelect	Устанавливает способ выбора элементов списка. Допустимые значения: fmMultiSelectSingle (выбор только одного элемента); fmMultiSelectMulti (разрешен выбор нескольких элементов посредством либо щелчка, либо нажатием клавиши <Пробел>); fmMultiSelectExtended (разрешено использование клавиши <Shift> при выборе ряда последовательных элементов списка)

Окончание таблицы 20

Свойство	Описание
ColumnHeads	Допустимые значения: True (выводятся заголовки столбцов раскрывающегося списка) и False (в противном случае)
Selected	Допустимые значения: True (если элемент списка выбран) и False (в противном случае). Используется для определения выделенного текста, когда свойство MultiSelect имеет значение fmMultiSelectMulti или fmMultiSelectExtended
ControlSource	Устанавливает диапазон (ячейку), куда возвращается выбранный элемент из списка
MatchEntry	Выводит первый подходящий элемент из списка при наборе его имени на клавиатуре. Допустимые значения: fmMatchEntryNone (режим вывода подходящего элемента в списке отключен); fmMatchEntryFirstLetter (выводит подходящий элемент по набранной первой букве. В этом случае предпочтительно, чтобы элементы списка были бы упорядочены в алфавитном порядке); fmMatchEntryComplete (выводит подходящий элемент по полному набранному имени)

В таблице 21 представлены наиболее часто используемые методы элемента управления ***ListBox***.

Таблица 21 – Методы элемента управления ***ListBox***

Метод	Описание
Clear	Удаляет все элементы из списка
RemoveItem	Удаляет из списка элемент с указанным номером. Синтаксис: RemoveItem (index), где index – номер удаляемого из списка элемента
AddItem	Добавляет элемент в список. Синтаксис: AddItem ([Item [, varIndex]]) Item – элемент (строковое выражение), добавляемый в список varIndex – номер добавляемого элемента

Заполнить список можно одним из следующих способов (таблица 22).

Таблица 22 – Способы заполнения списка ***ListBox***

Способ заполнения списка	Программный код
Массивом, если список состоит из нескольких колонок, например, двух	*Dim A (2, 1) As String* *A(0, 0) = «Июнь»* *A(0, 1) = «Сессия»* *A(1, 0) = «Июль»* *A(1, 1) = «Каникулы»* *A(2, 0) = «Август»* *A(2, 1) = «Каникулы»* *With ListBox1* *.ColumnCount = 2* *.List = A* *End With*

Окончание таблицы 22

Поэлементно, если список состоит из нескольких колонок, например, двух	*With ListBoxl* *.ColumnCount = 2* *.Addltem «Июнь»* *.List(0, 1) = «Сессия»* *.Addltem «Июль»* *.List(l, 1) = «Каникулы»* *.Addltem «Август»* *.List (2, 1) = «Каникулы»* *End With*
Поэлементно, если список состоит из одной колонки	*With ListBox1* *.AddItem «Июнь»* *.AddItem «Июль»* *.AddItem «Август»* *End With*
Из диапазона A1:B4, в который предварительно введены элементы списка. Результат выбора (индекс выбранной строки) выводится в ячейку C1	*With ListBoxl* *.ColumnCount = 2* *.RowSource = «A1:B4»* *.ControlSource = «C1»* *End With*
Массивом, если список состоит из одной колонки	*With ListBoxl* *.List = Array(«Июнь»,_ «Июль», Август»)* *.Listlndex= 1* *End With*

Пример – Создадим приложение, которое позволит подсчитать сумму или произведение выбранных в списке чисел.

Перейдем в VBA и, выполнив команду **Insert (Вставка) →UserForm**, добавим в проект форму. Расположим на форме следующие элементы управления (рисунок 10). Описание создаваемой формы представлено в таблице 23.

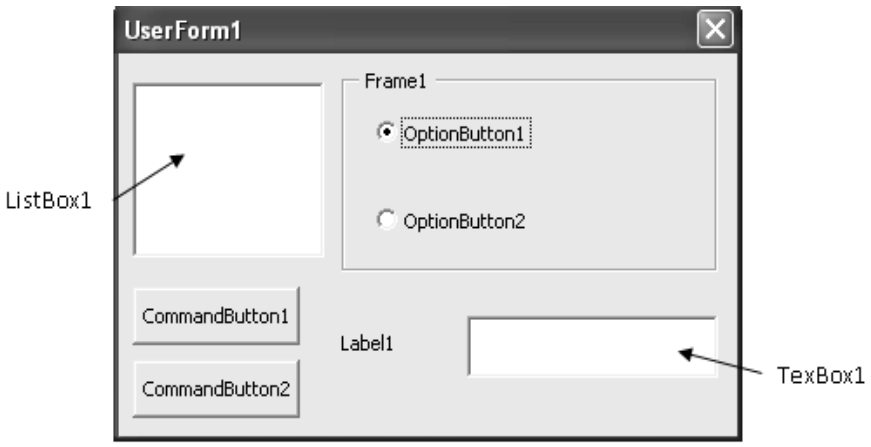


Рисунок 10 – Проектируемая пользовательская форма

Таблица 23 – Описание создаваемой формы

Элемент управления	Предназначение
CommandButton1 (кнопка)	Нажатие на кнопку запускает процедуру обработки события (Private Sub CommandButton1_Click()), которая определяет, какой переключатель выбран. В зависимости от выбранного переключателя производится действие над выбранными в списке числами. Найденное значение выводится в поле TextBox1
CommandButton2 (кнопка)	Нажатие на кнопку запускает процедуру обработки события (Private Sub CommandButton2_Click()), которая закрывает диалоговое окно
TextBox1 (поле)	В это поле будет выводиться результат. Поле сделаем недоступным для пользователя, т. е. пользователь не сможет ни ввести, ни скорректировать данные в этом поле
Label1 (надпись)	Пояснительная надпись для поля вывода
Frame1 (рамка)	Используется для группировки переключателей
OptionButton1 (переключатель)	Выбор переключателя указывает, какая операция будет выполняться над выбранными числами

Форма создана, осталось только в модуле формы набрать код (таблица 24).

Таблица 24 – Программа подсчета суммы или произведения выбранных в списке чисел

Текст программы	Описание
Private Sub CommandButton1_Click() *Dim i As Integer* *Dim n As Integer* *Dim Сумма As Double* *Dim Произведение As Double* *Dim Результат As Double* *If OptionButton1.Value = True Then* *Сумма = 0* *With ListBox1* *For i = 0 To .ListCount - 1* *If .Selected*(*i*) *= True Then* *Сумма = Сумма + .List*(*i*) *End If* *Next i* *End With* *Результат = Сумма* *End If* *If OptionButton2.Value = True Then* *Произведение = 1* *With UserForm1.ListBox1* *For i = 0 To .ListCount - 1* *If .Selected*(*i*) *= True Then* *Произведение = Произведение* .List*(*i*) *End If* *Next i* *End With* *Результат = Произведение* *End If* *TextBox1.Text = CStr*(*Результат*) *End Sub*	При выборе первого переключателя вычисляется сумма выбранных элементов При выборе второго переключателя вычисляется произведение выбранных элементов Результат выводится в поле *TextBox*1

Окончание таблицы 24

Текст программы	Описание
Private Sub CommandButton2_Click() *UserForm1.Hide* *End Sub*	Процедура закрытия диалогового окна
Private Sub UserForm_Initialize() *With ListBox1* *.List = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)* *.ListIndex = 0* *.MultiSelect = fmMultiSelectMulti* *End With* *With UserForm1.OptionButton1* *.Value = True* *.Caption = «Сумма»* *.ControlTipText = «Сумма выбранных_ элементов»* *End With*	Процедура инициализации диалогового окна Заполнение списка Установка режима выбора При загрузке формы первоначально будет выбран переключатель «Сумма»
OptionButton2.ControlTipText =_ «Произведение выбранных_ элементов» *CommandButton2.ControlTipText = «Выход из программы»* *CommandButton1.ControlTipText = «Нахождение результата»* *UserForm1.Caption = «Операции над элементами списка»* *OptionButton2.Caption = «Произведение»* *Label1.Caption = «Результат»* *CommandButton1.Caption = «Вычислить»* *CommandButton2.Caption = «Отмена»* *Frame1.Caption = «Операция»* *TextBox1.Enabled = False* *End Sub*	Задание текста всплывающих подсказок у элементов управления Задание заголовка пользовательской формы Инструкции задают видимые надписи для объектов Инструкция делает *TextBox*1 недоступным для пользователя

После конструирования формы и написания кода в модуле формы выберем команду ***Run*** и на экране появится форма, представленная на рисунке 11.

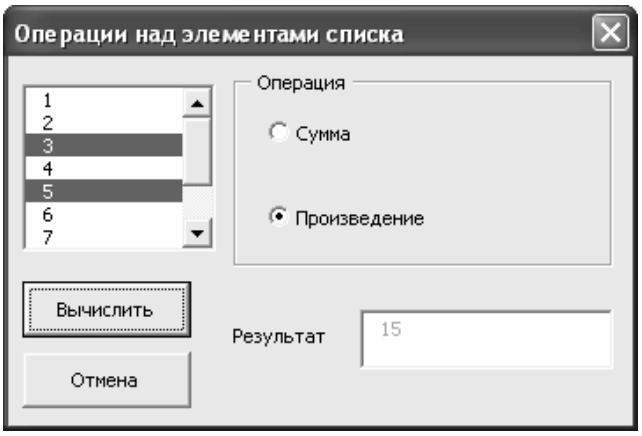


Рисунок 11 – Спроектированная форма

Элемент управления ***ComboBox*** (поле со списком) применяется для хранения списка значений. Он сочетает в себе функциональные возможности списка ***ListBox*** и поля ***TextBox***. В отличие от ***ListBox*** в элементе управления ***ComboBox*** отображается только один элемент списка. Кроме того, у него отсутствует режим выделения нескольких элементов списка, но он позволяет вводить значение, используя поле ввода, как это делает элемент управления ***TextBox***.

Свойства объекта ***ComboBox***, такие как Listlndex, ListCount, Enabled, List, и методы Clear, RemoveItem и AddItem аналогичны соответствующим свойствам и методам списка ***ListBox***. Кроме того, у ***ComboBox*** есть ряд уникальных свойств. Далее приведены наиболее употребляемые из уникальных свойств элемента управления ***ComboBox*** (таблица 25).

Таблица 25 – Свойства элемента управления ***ComboBox***

Свойство	Описание
MatchRequired	Допустимые значения: True (в поле ввода раскрывающегося списка нельзя ввести значения, отличные от перечисленных в списке, т. е. в поле со списком отключается функция поля ввода) и False (в противном случае)
DropButtonStyle	Устанавливает вид раскрывающегося списка. Допустимые значения: FmDropButtonStylePlain (кнопка без символов); FmDropButtonStyleArrowDisplays (кнопка со стрелкой); FmDropButtonStyleEllipsis (кнопка с эллипсом); FmDropButtonStyleReduce (кнопка с линией)
ListRows	Устанавливает число элементов, отображаемых в раскрывающемся списке
MatchFound	Допустимые значения: True (среди элементов раскрывающегося списка имеется элемент, совпадающий с вводимым в поле ввода раскрывающегося списка) и False (в противном случае)

Контрольные вопросы

1 Какое назначение элементов управления *ComboBox* и *ListBox*?

2 Назовите основные свойства элементов управления *ComboBox* и *ListBox*.

3 Понятие события. Как происходит обработка событий формы *ComboBox* и *ListBox*?

Лабораторная работа № 12. Программа с элементами управления: счетчик, полоса прокрутки

Цель работы: изучить свойства, события и методы элементов управления ***SpinButton***, ***ScrollBar***; разработать программу с вводом пользователем значений при помощи счетчика или полосы прокрутки.

Задание

На основе созданного в лабораторной работе № 11 приложения необходимо обеспечить ввод соответствующих параметров фигур через ***SpinButton*** (четные варианты) и ***ScrollBar*** (нечетные варианты), произвести расчет площади (варианты 1–10) или объема (варианты 11–20) фигуры.

Методические указания

Элемент управления ***ScrollBar*** (полоса прокрутки) обладает рядом свойств. Приведем наиболее часто используемые свойства элемента управления ***ScrollBar*** (таблица 26).

Таблица 26 – Свойства элемента управления ***ScrollBar***

Свойство	Описание
Value	Возвращает текущее значение полосы прокрутки (только целые неотрицательные числа)
Min	Минимальное значение полосы прокрутки (только целые неотрицательные числа)
Max	Максимальное значение полосы прокрутки (только целые неотрицательные числа)
SrnallChange	Устанавливает шаг изменения значения при щелчке по одной из стрелок полосы прокрутки
Enabled	Допустимые значения: True (пользователь может изменить значение полосы прокрутки) и False (в противном случае)
Visible	Допустимые значения: True (полоса прокрутки отображается во время выполнения программы) и False (в противном случае)

Элемент управления ***SpinButton*** по своим функциональным возможностям аналогичен полосе прокрутки. Счетчик – это полоса прокрутки без ползунка. Счетчик имеет те же свойства Value, Min, Max, Enabled, Visible и SmallChange, что и полоса прокрутки.

Контрольные вопросы

1 Какое назначение элементов управления ***ScrollBar*** и ***SpinButton***?

2 Назовите основные свойства элементов управления ***ScrollBar***, ***SpinButton***.

3 Понятие события. Как происходит обработка событий элементов формы ***ScrollBar*** и ***SpinButton***?

Лабораторная работа № 13. Создание пользовательских диалоговых окон

Цель работы: создать пользовательские диалоговые окна с применением различных элементов управления.

Задание

Необходимо создать форму (диалоговое окно пользователя) для расчета параметра (по вариантам). Все расчеты производятся при нажатии кнопки «Расчет». Данная кнопка должна срабатывать при нажатии на клавишу <*Enter*> (независимо от того, какой элемент управления является активным в момент нажатия данной клавиши). Перед проведением расчета программа должна выполнить проверку корректности ввода данных. Результат расчета выводится в соответствующее текстовое поле ***TextBox*** и на страницу *Excel*. Кнопка <Отмена> должна закрывать текущую форму и срабатывать при нажатии на клавишу <*Esc*>.

Варианты индивидуальных заданий для выполнения лабораторной работы выдаются преподавателем.

Методические указания

Инициализировать и отобразить диалоговое окно на экране очень просто. Инициализация производится при помощи процедуры обработки события *Initilize* формы *UserForm.* Отображение диалогового окна на экране осуществляется методом *Show*. Инструкцию с методом *Show* обычно помещают в процедуру, которая связана с командой пользовательского меню, кнопкой панели инструментов или элементом управления, как правило, кнопкой диалогового окна.

Простой инициализации или обычного отображения диалогового окна часто бывает недостаточно, т. к. это приводит к появлению на экране функционально ненастроенного диалогового окна. Такое диалоговое окно можно сравнить с каркасом дома. В таком доме жить неприятно и в него совсем не хочется въезжать. Для того чтобы жить в доме было приятно и удобно, прежде чем в него вселяться, надо сделать много отделочных работ. Также и при инициализации диалогового окна необходимо предусмотреть огромное количество на первый

взгляд мелочей, но без которых работать с диалоговым окном неудобно. В частности, при отображении диалогового окна на экране нужно установить значения полей, применяемые по умолчанию, задать функции кнопок, назначить им комбинации клавиш, связать с элементами управления всплывающие подсказки, вывести в списках первоначально выводимые элементы списков, задать первоначальную установку флажков, вывести в элементы управления формы требуемые рисунки и т. д.

В VBA диалоговые окна работают в режиме модального диалога. Это означает, что пользователь, прежде чем перейти к выполнению действий, не связанных с текущим активным диалоговым окном, должен его закрыть. Закрытие диалогового окна производится методом *Hide*. Следующая процедура является примером процедуры закрытия диалогового окна. Эта процедура активизируется при нажатии кнопки *CommandButton*2 диалогового окна UserForm1 и выполняет только одну инструкцию, осуществляющую закрытие этого диалогового окна.

```
Private Sub CommandButton2_Click()
        UserForm1.Hide ‘ Процедура закрытия диалогового окна
End Sub
```

Закрыть диалоговое окно также, конечно, можно, нажав системную кнопку, расположенную в правом верхнем углу любого диалогового окна. Если при закрытии диалогового окна необходимо произвести какие-то действия, например, считать информацию из окна в файл на диске и т. д., во избежание потери информации, действия, производимые программой при закрытии окна, разумно также продублировать в процедуре обработки события *Terminate* (закрытие) пользовательской формы

Контрольные вопросы

1 Как создать графический интерфейс своего приложения с помощью VBA?

2 Опишите самые важные свойства и методы форм.

3 Опишите основные свойства элементов управления.

4 Как происходит обработка событий формы?

5 Как обеспечивается связывание помещенных на форму элементов управления?

6 Как назначить горячие клавиши элементам управления?

7 Как назначить последовательность перехода фокуса по элементам управления?

8 Для чего применяется свойство *Visible* элементов управления?

9 Для чего применяется свойство *Enable* элементов управления?

10 Для чего применяется свойство *Cancel* элементов управления?

11 Как расширить набор стандартных элементов управления?

Список литературы

1 **Головин, И. Г.** Языки и методы программирования: учебник / И. Г. Головин, И. А. Волкова. – 2-е изд., стер. – Москва: Академия, 2016. – 304 с.

2 Информационные системы в экономике: учебное пособие / Под общ. ред. М. Н. Садовской. – Минск: БГЭУ, 2018. – 316 с.

3 **Стариченко, Б. Е.** Теоретические основы информатики: учебник / Б. Е. Стариченко. – 3-е изд., перераб. и доп. – Москва: Горячая линия – Телеком, 2017. – 400 с.

4 **Скитер, Н. Н.** Информационные технологии: учебное пособие / Н. Н. Скитер, А. В. Костикова. – Волгоград: ВолгГТУ, 2019. – 96 с.

5 **Щеглов, А. Ю.** Защита информации. Основы теории: учебник / А. Ю. Щеглов, К. А. Щеглов. – Москва: Юрайт, 2019. – 309 с.