ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономика и управление»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

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

Могилев 2018

УДК 004
ББК 32.973
И 74

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Экономика и управление» «20» сентября 2018 г., протокол № 2

Составитель ст. преподаватель Е. Г. Галкина

Рецензент канд. экон. наук, доц. Т. Г. Нечаева

Приведены задания и рекомендации по их выполнению для освоения курса «Информационные технологии» студентами направления подготовки 27.03.05 «Инноватика» дневной формы обучения.

Учебно-методическое издание

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Ответственный за выпуск	И. В. Ивановская
Технический редактор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилёв.

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.

Отчет должен быть оформлен на листах формата А4 и подшит в папку. Папка должна быть подписана с указанием дисциплины, группы и фамилии студента.

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. Условный оператор: однострочная и блочная формы

Цель работы: изучить оператор ветвления ***If***; создать программу с использованием оператора ветвления ***If*** на языке программирования VBA.

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 1. Синтаксис условного оператора ***If*** в однострочной форме следующий:

If <лог. выраж.> Then $P_1 : P_2 : ... : P_N$ *Else* $M_1 : M_2 : ... : M_N$

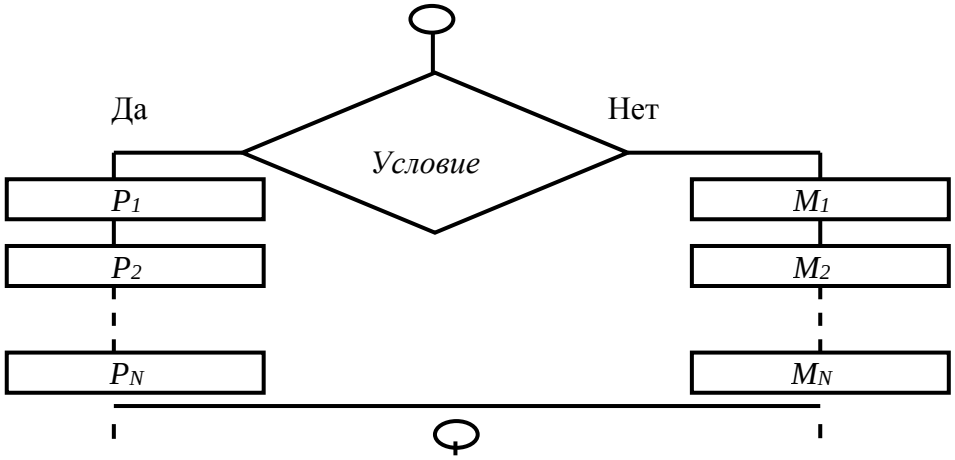


Рисунок 1 – Схема алгоритма конструкции условного оператора *If*

Возможна и другая синтаксическая форма – блочная (структурная):

If <логическое выражение> 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.

Задание

1 Запросить у пользователя ввод числа.

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

3 По результатам сравнения вывести соответствующее сообщение:

«25 > 20» или «15 < 20»,

где 25, 15 – введенные пользователем числа.

Контрольные вопросы

1 Можно ли вставлять инструкцию ***Else*** перед инструкцией ***Else IF*** в блочном варианте оператора ветвления?

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

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

Лабораторная работа № 2. Многозначные ветвления If

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

Методические указания

Схема алгоритма конструкции оператора многозначных ветвлений ***If*** представлена на рисунке 2.

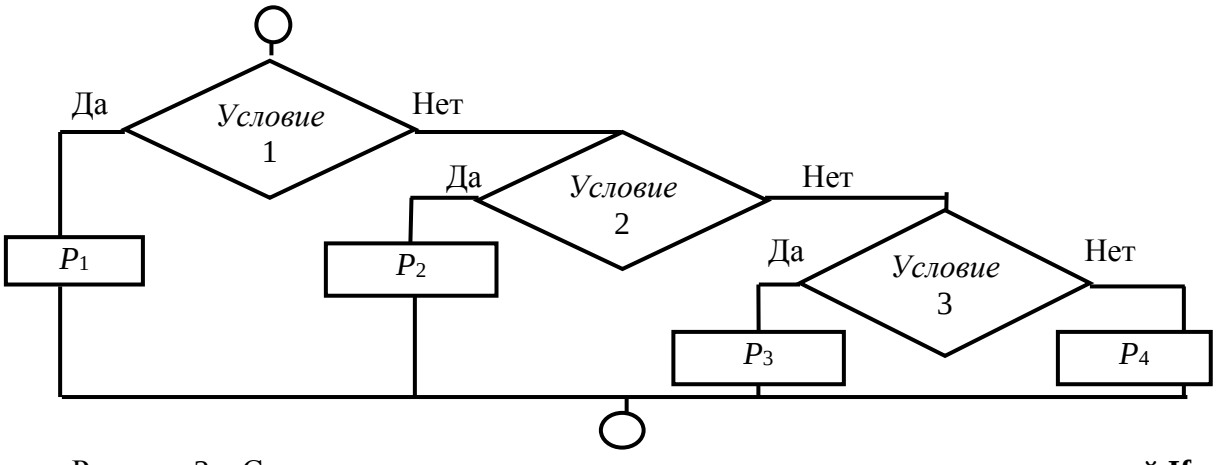


Рисунок 2 – Схема алгоритма конструкции оператора многозначных ветвлений **If**

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

If<лог.выражение1>Then
P_1
Else If<лог.выражение2>Then
P_2
Else
P_3
End If

где *If*, *Then*, *Else*, *End If* – зарезервированные слова;
P_1, P_2, P_3 – операторы.

Алгоритм работы такой конструкции:

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

– если логическое выражение **1** ложно, то оператор $\boldsymbol{P_1}$ пропускается и анализируется логическое выражение **2**, следующее за ***Else If***. 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 представлены примеры использования условного оператора ***If*** и 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*»)
ElseIfball>=20 Then
MsgBox («*Вам можно поступать в СУЗ*»)
Else If ball>=5 Then
MsgBox («*Вам можно поступать в ПТУ*»)
End If

Задание

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

2 С применением оператора многозначных ветвлений ***If*** осуществить преобразование введенной оценки по 100-балльной системе в 5-балльную по

шкале, предварительно созданной на листе Excel (таблица 1).

Таблица 1 – 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 включ.	Оценка 5
< 0 или > 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.

Контрольные вопросы

1 В каких случаях в программе используется вложенный условный оператор?
2 Сколько инструкций ***Else IF*** может быть в блочном варианте оператора ветвления?
3 Нарисуйте алгоритмическую схему выполнения вложенного условного оператора.

Лабораторная работа № 3. Оператор выбора Select Case

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

Методические указания

При наличии большого количества ветвлений конструкция многозначных ветвлений **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 **ElseIf** 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. 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 **списка Выражений** 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 список Выражений** 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 2.

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 «*Заслуженный отдых*».

Из представленного в таблице 2 примера видно, что код этой процедуры более прост для восприятия, чем многозначные ветвления ***If***.

Таблица 2 – 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*

Задание

Выполнить задание предыдущей лабораторной работы с применением оператора выбора ***Select Case***.

Контрольные вопросы

1 В каких случаях в программе используется оператор выбора?

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

3 Когда применение оператора Select Case эффективнее оператора If Then Else End If?

Лабораторная работа № 4. Оператор цикла с параметром For … Next

Цель работы: изучить оператор цикла ***For***.

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 3.

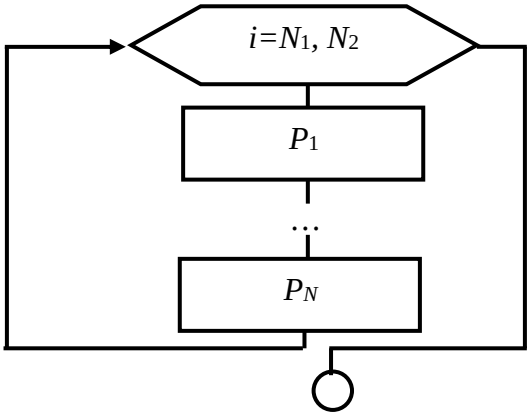


Рисунок 3 – Схема алгоритма конструкции оператора цикла *For … Next*

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

For i = N$_1$ *To N*$_2$ [*Step h*]
P$_1$
…
[*Exit For*] } *Тело цикла*
P$_N$
Next i

For (для), ***To*** (до), ***Step*** (шаг), ***Exit For*** (выход из ***For***), ***Next*** (следующий) – служебные слова ***VBA***, а ***P*1**, ***PN*** – 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.

Задание

1 Запросить у пользователя ввод целого числа больше 0.

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

3 Результат вычисления вывести в виде сообщения.

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

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 № 5. Вложенные операторы цикла For … Next

Цель работы: изучить вложенные операторы цикла ***For***.

Методические указания

Если телом цикла является циклическая структура, то такие циклы называются вложенными. Цикл, содержащий в себе другой цикл, – внешний, а цикл, содержащийся в теле другого цикла, – внутренний. Синтаксис конструкции вложенных операторов цикла ***For … Next*** следующий:

For $i = N_1$ *To* N_2
For $j = M_1$ *To* M_2
P_1
…
P_N
Next j
Next i

тело внутреннего цикла

тело внешнего цикла

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

При первом вхождении в цикл параметр внешнего цикла ***i*** принимает значение, равное N_1, и управление передаётся во внутренний цикл, в котором параметр цикла ***j*** принимает значение, равное M_1, и выполняется оператор (операторы), который записан во внутреннем цикле. Затем параметр внутреннего цикла ***j*** увеличивается на **1**, и вновь повторяется тело цикла.

Операторы P_1, P_N будут реализовываться до тех пор, пока параметр цикла ***j*** не станет больше величины M_2. Затем параметр внешнего цикла ***i*** увеличивается на **1**, и вновь начинает свою работу внутренний цикл, в котором параметр цикла ***j*** будет изменяться от M_1 до M_2, и при каждом прохождении цикла будут выполняться операторы P_1 и P_N. Внешний цикл закончит свою работу, когда параметр цикла ***i*** станет больше величины N_2.

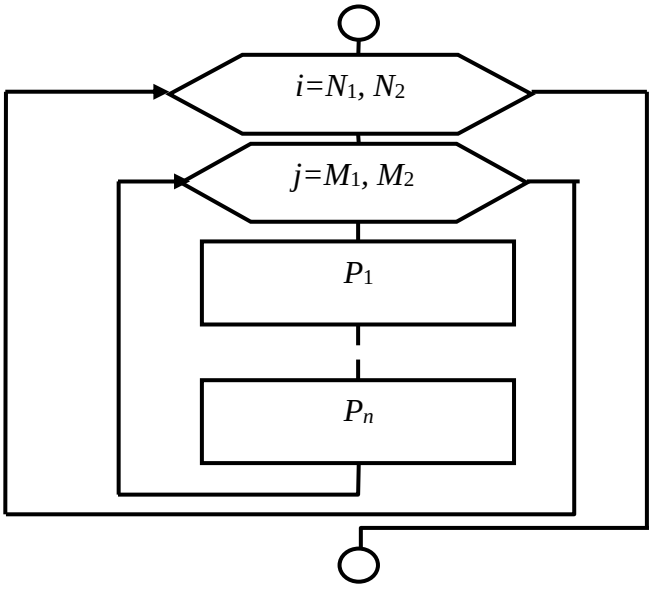


Рисунок 4 – Схема алгоритма конструкции вложенных операторов цикла *For … Next*

Задание

С помощью вложенных операторов цикла ***For … Next*** реализовать запись на рабочий лист ***Excel*** матрицы размерностью ***n* × *n*** (значение числа ***n*** запросить у пользователя), элементы которой должны являться произведением номера строки на номер колонки.

Контрольные вопросы

1 В каких случаях используются вложенные операторы цикла?

2 Как оформляются вложенные операторы цикла?

3 Нарисуйте алгоритмическую схему выполнения вложенных операторов цикла.

4 Вложенный цикл образован двумя операторами цикла с параметром. Что является телом?

5 Может ли внешний оператор вложенного цикла: не выполниться ни разу; выполняться бесконечное число раз (или до того момента, когда пользователь прервет его выполнение)?

Лабораторная работа № 6. Цикл While … Wend (цикл с предусловием)

Цель работы: изучить оператор цикла ***While …Wend***.

Методические указания

Для выполнения оператора цикла ***For … Next*** необходимо задать параметры, определяющие, сколько раз должен повториться оператор(ы) цикла. Альтернативой циклу ***For … Next*** являются циклы ***While*** и ***Do***, в которых

группа операторов выполняется до тех пор, пока определённое логическое выражение имеет значение ***True*** (истина) или ***False*** (ложь). Такие циклы нужно применять в тех задачах, где мы не можем знать точно, сколько раз будет повторен цикл.

Наиболее простой конструкцией построения цикла является конструкция ***While … Wend***. Также существует несколько разновидностей альтернативной конструкции – цикла ***Do***, в зависимости от условий его выполнения. Конструкции циклов ***Do*** являются более универсальными и, соответственно, сложными.

Общий вид алгоритма конструкции цикла ***While … Wend*** представлен на рисунке 5.

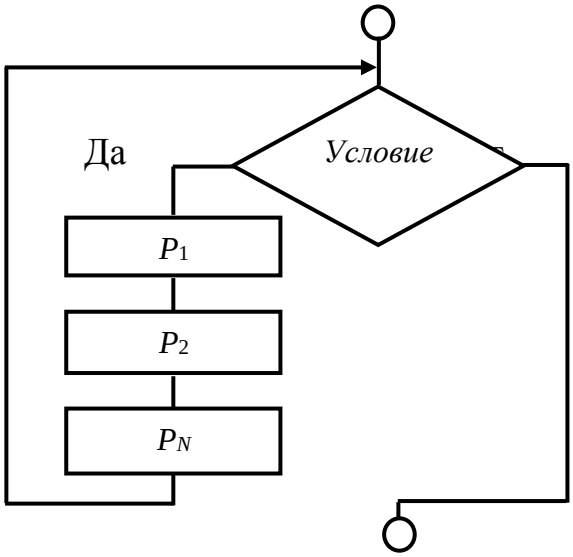


Рисунок 5 – Общий вид алгоритма конструкции цикла *While … Wend*

Синтаксис операторов данного цикла:

While<лог.выражение>
P_1
…
P_N
Wend

тело цикла

В данном цикле сначала производится проверка логического выражения, при его истинности – выполняется тело цикла. Затем – повторно проверка условия. Таким образом, цикл повторяется до тех пор, пока заданное логическое выражение истинно. Если условие в заголовке цикла не является истинным с самого начала, цикл не выполняется ни разу.

Задание

Смоделировать работу оператора сотовой связи, который проверяет наличие денег на счете абонента (предлагается начальная величина в 500 р.),

при наличии денег – позволяет совершить звонок с последующим уменьшением количества денег на счете.

Стоимость звонка условно принимается постоянной величиной, например, 150 р.

В теле цикла должно выдаваться сообщение:

«*Баланс:* «<*СуммаДенег*>«. *Звонок разрешается!*».

При отсутствии денег на счете выдается соответствующее сообщение:

«*Задолженность*:»<*СуммаДенег*>.

Контрольные вопросы

1 Нарисуйте алгоритмическую схему выполнения цикла с предусловием.

2 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)?

Лабораторные работы № 7–8. Цикл Do While … Loop (цикл с предусловием). Цикл Do … While Loop (цикл с постусловием)

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

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 3, а операторы циклов с постусловием (Do … Loop While, Do … Loop Until) − в таблице 4.

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.

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 5.

Таблица 3 – Циклы с предусловием

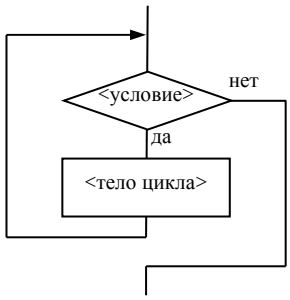
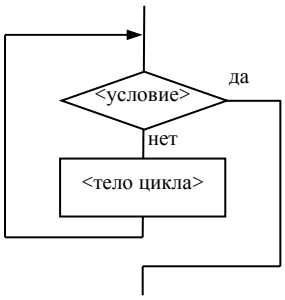
Синтаксис	**DoWhile**<*условие*> <*тело цикла*> [**ExitDo**] … **Loop**	**DoUnlil**<*условие*> <*тело цикла*> [**ExitDo**] … **Loop**
Порядок выполнения	<*Тело цикла*> будет выполняться в том случае, когда <*условие*> имеет значение Истина (TRUE) (цикл продолжается при истинном значении <*условия*>). 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**)
Изображение в блок-схемах		

Таблица 4 – Циклы с предусловием

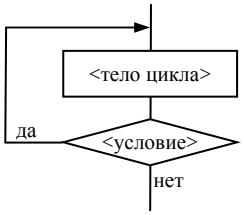
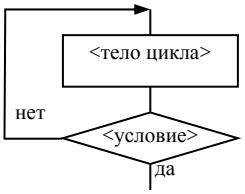
Синтаксис	**Do** <*тело цикла*> [**Exit Do**] … **Loop While**<*условие*>	**Do** <*тело цикла*> [**Exit Do**] … **Loop Until** <*условие*>
Порядок выполнения	<*Тело цикла*> будет выполняться в том случае, когда <*условие*> имеет значение Истина (цикл продолжается при истинном значении <*условия*>). Если <*условие*> ложно, то выполняются операторы, стоящие за циклом. Предоставлена возможность досрочного выхода из цикла (это реализовано через **ExitDo**)	<*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 блок-схемах		

Таблица 5 – 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*) *EndSub*	 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*

Задание

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!».

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: не выполниться ни разу; выполняться бесконечное число раз (или до тех пор, когда пользователь прервет его выполнение)?

Лабораторная работа № 9. Массивы

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

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 **OptionBase 1**.

Пример 1:

а) DimA(1 To 10) AsInteger – массив А состоит из 10 элементов целого типа, индексы которых 1, 2, …, 10;

б) Dim A(10) AsInteger – 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 **OptionBase 1**), то номера элементов изменяются от 1 до 10;

в) **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:

DimA( ) 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 5 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 объявления массивов:

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 А будет А(0), а последний − А(11). В этом случае говорят, что 0 – базовый индекс. Можно изменить базовый индекс, написав в начале листа модуля инструкцию OptionBase 1. После этого индексы массивов А и В 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.

Задание

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:* ____».

Контрольные вопросы

1 Опишите синтаксис объявления массива с использованием оператора Dim.

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?

Лабораторная работа № 10. Процедуры и функции

Цель работы: изучить процедуры и функции и их применение.

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 6.

Таблица 6 – Сравнительная характеристика процедур и функций

Описание	Подпрограмма-процедура	Подпрограмма-функция
	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>)

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

<*Список параметров*>позволяет передать в подпрограмму требуемые значения из вызывающей программы и имеет следующий синтаксис:

[***ByRef***|***ByVal***] <*имя параметра1*> [**As**<*Тип*>], [***ByRef***|***ByVal***] <*имя параметра2*> [**As**<*Тип*>], [***ByRef***|***ByVal***] <*имя параметра3*> [**As**<*Тип*>], …

<*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 изме-

нение 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, используются **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). Текст программы с пояснениями представлен в таблице 7. Блок-схема реализации вычисления приведена на рисунке 7.

Таблица 7 – Программа вычисления суммы членов ряда

Текст программы	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)* *Nexti* *MsgBox s* *End Sub*	При вычислении искомой суммы производится вызов функции factor и передается ее аргумент *i*
Public Function faktor(x As Integer) As Long *faktor = 1* *For i = 1 To x* *faktor = faktor * i* *Next i* *End Function*	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

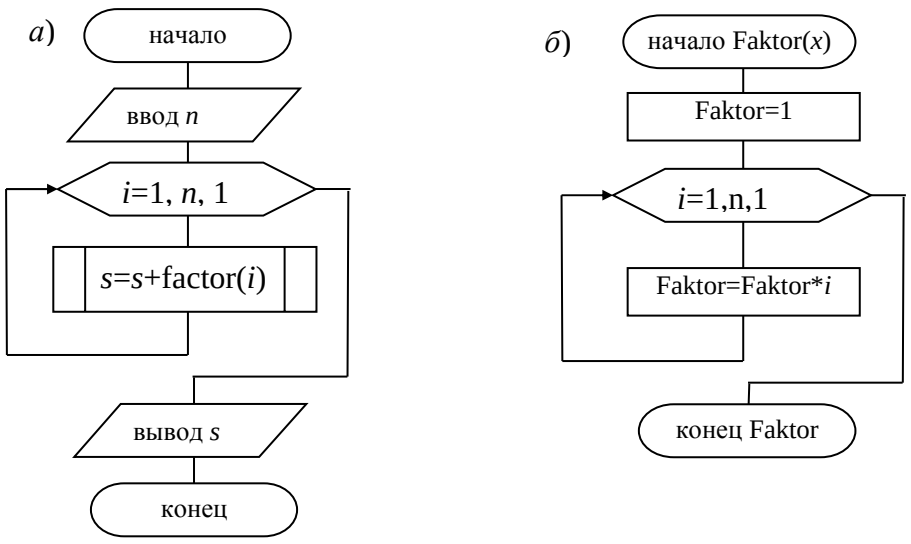


Рисунок 7 – Блок-схема программы prog3 (*а*) и подпрограммы Faktor (*б*)

Пример 2 − Построить график функции $y = \frac{\sin x}{x^2 + 1}$ на отрезке [a; b] с шагом 0,1. Построение графика организовать процедурой. Текст программы с пояснениями представлен в таблице 8. Блок-схема реализации вычисления приведена на рисунке 8.

Таблица 8 – Программа построения графика функции

Текст программы	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) / (i ^ 2 + 1)* *j = j + 1* *Next i* *график* *End Sub*	Выделение двух столбцов (A и B) 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»
Sub график() *n=Application.CountA*(*Worksheets* (*1*).*Range*(«*A:A*»)) *Range*(«*A1:B*»*&CStr*(*n*)).*Select* *Charts.Add* *ActiveChart.ChartType = xlXYScatterSmoothNoMarkers* *ActiveChart.SetSourceData* *Source:=Sheets*(«*Лист1*»).*Range*(«*A1:B*»*&CStr*(*n*)), *PlotBy:= xlColumns* *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 A и это значение присваивается переменной n

Блок-схема макроса не изображается.

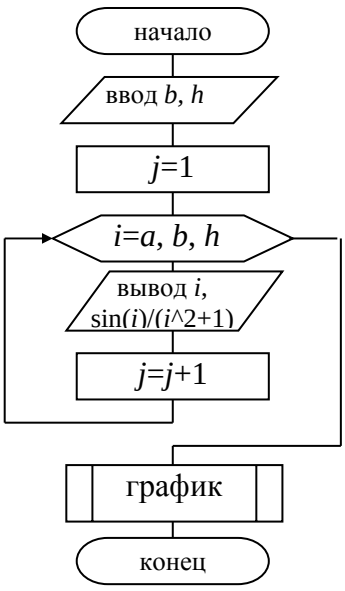


Рисунок 8 − Блок-схема программы prog4

Задание

1 Создать новую процедуру и задать в ней вызов любой из процедур, разработанных в рамках лабораторных работ № 1–9.

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:* ____».

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?

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?

Лабораторная работа № 11. Ознакомление с пользовательской формой. Использование элементов управления TextBox, Label, CommandButton

Цель работы: ознакомиться с возможностями создания интерфейса в VBA, некоторыми его элементами, их свойствами и методами.

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 9 представлен список основных элементов управления и соответствующих кнопок панели элементов.

Таблица 9 – 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 лист или в форму 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 из формы.

В таблице 10 представлены основные общие свойства элементов управления.

Таблица 10 – 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)

Окончание таблицы 10

Свойство	Описание
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
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
BackStyle	Устанавливает тип заднего фона
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 11 представлены наиболее часто используемые свойства элементов управления.

Таблица 11 – 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)
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 или остановка ее выполнения, 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>

Окончание таблицы 11

Свойство	Описание
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).

Перейдем в VBA и, выполнив команду **Insert (Вставка) →UserForm**, добавим в проект форму. Расположим на форме следующие элементы управления: Label, TextBox, CommandButton (рисунок 9).

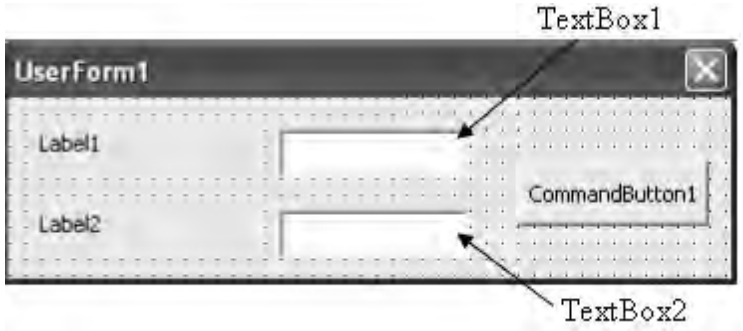


Рисунок 9 – Вид формы в режиме конструктора

В таблице 12 приведено описание создаваемой формы.

Таблица 12 – 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

Форма создана, функция каждого элемента управления известна. Для написания кода программы, связанного с пользовательской формой, достаточно

дважды щелкнуть, например, кнопку CommandButton1. 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 13.

Таблица 13 – 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 TextBox1 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*)»	Процедура UserForm_Initialize конструирует форму до ее загрузки Инструкция устанавливает текст, отображаемый в строке заголовка формы
Label1.Caption = «*Аргумент*» *Label2.Caption =* «*Значение функции*» *CommandButton1.Caption =* «*OK*» *TextBox2.Enabled = False* *EndSub*	Инструкции задают видимые надписи для объектов Инструкция делает TextBox2 недоступным для пользователя

После конструирования формы и написания кода в модуле формы выберем команду **Run→Run Sub/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 10.

Рисунок 10 – Вид пользовательской формы

Задание

Выполнить примеры из методических указаний.

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 № 12. Использование элементов управления CheckBox, OptionButton

Цель работы: изучить свойства элементов управления CheckBox, OptionButton, использовать их для решения задач.

Методические указания

Элемент управления OptionButton (переключатель) создается с помощью кнопки **Переключатель** (OptionButton). 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 14).

Таблица 14 – 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.

Задание

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

Контрольные вопросы

1 Какое назначение элементов управления OptionButton, CheckBox и Frame?

2 Назовите основные свойства элементов управления OptionButton, CheckBox и Frame.

3 Понятие события. Как происходит обработка событий OptionButton, CheckBox?

Лабораторная работа № 13. Использование элементов управления ListBox, ComboBox, Frame

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

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 15 представлены часто используемые свойства элемента управления ListBox.

Таблица 15 – 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 15

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
ListStyle	Устанавливает способ выделения выбранных элементов. Допустимые значения: fmListStylePlain (выбранный элемент из списка выделяется цветом); fmListStyleOption (перед каждым элементом в списке располагается флажок и выбор элемента из списка соответствует установке этого флажка)
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 упорядочены в алфавитном порядке); fmMatchEntryComplete (выводит подходящий элемент по полному набранному имени)

В таблице 16 представлены наиболее часто используемые методы элемента управления ListBox.

Таблица 16 – 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 17).

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 11). Описание создаваемой формы представлено в таблице 18.

Таблица 17 – Способы заполнения списка ListBox

Способ заполнения списка	Программный код
Поэлементно, если список состоит из одной колонки	*With ListBox1* *.AddItem "Июнь»* *.AddItem "Июль»* *.AddItem "Август»* *End With*
Массивом, если список состоит из одной колонки	*With ListBoxl* *.List = Array(«Июнь», «Июль», «Август»)* *.Listlndex= 1* *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*
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*
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*

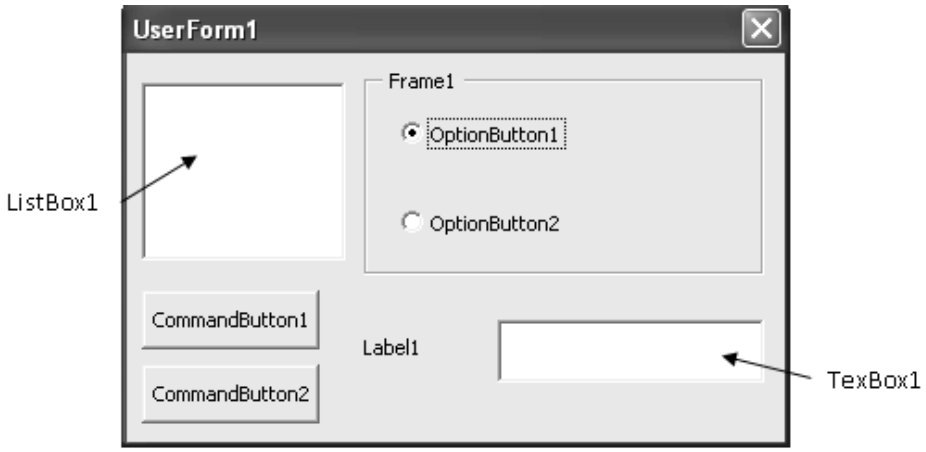


Рисунок 11 – Проектируемая пользовательская форма

Таблица 18 – 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 19).

После конструирования формы и написания кода в модуле формы выберем команду **Run** и на экране появится форма, представленная на рисунке 12.

Таблица 19 – 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*	При выборе первого переключателя вычисляется сумма выбранных элементов При выборе второго переключателя вычисляется произведение выбранных элементов

Окончание таблицы 19

Текст программы	Описание
Nexti *End With* *Результат = Произведение* *EndIf* *TextBox1.Text = CStr(Результат)* *End Sub*	Результат выводится в поле TextBox1
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* *OptionButton2.ControlTipText = «Произведение выбранных элементов»* *CommandButton2.ControlTipText = «Выход из программы»* *CommandButton1.ControlTipText = «Нахождение результата»*	Процедура инициализации диалогового окна Заполнение списка Установка режима выбора При загрузке формы первоначально будет выбран переключатель «Сумма» Задание текста всплывающих подсказок у элементов управления
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 заголовка пользовательской формы Инструкции задают видимые надписи для объектов Инструкция делает TextBox1 недоступным для пользователя

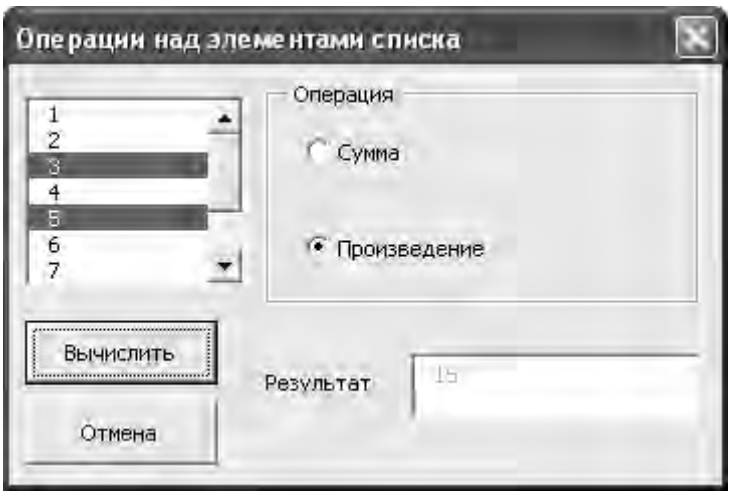


Рисунок 12 – Спроектированная форма

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 20).

Таблица 20 – 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)

Задание

Создать простое приложение, состоящее из формы, представленной на рисунке 13. 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;

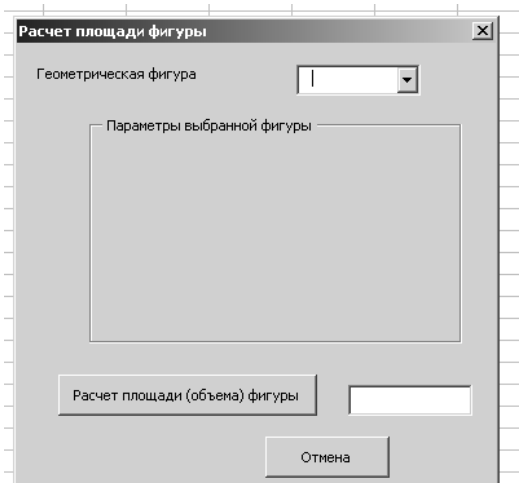
– раскрывающийся список, позволяющий выбрать геометрическую фигуру (не менее трех);

– текстовые поля для ввода параметров выбранной фигуры;

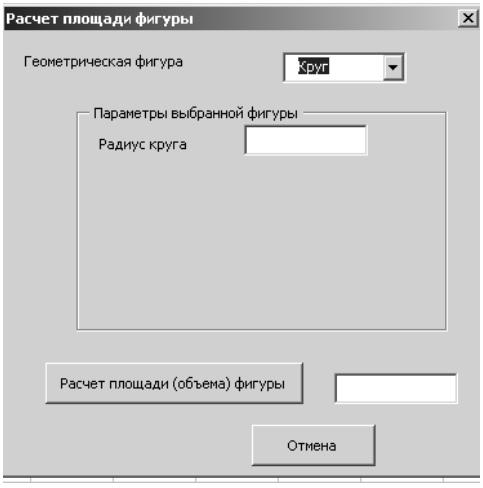
– кнопку, при нажатии на которую будет производиться расчет площади фигуры или объема фигуры;

– поле, размещенное в нижней части формы с правой стороны кнопки, предназначенное для отображения полученного результата.

а)



б)



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

Рисунок 13 – Пример приложения, позволяющего производить вычисления на основе введенных данных (вариант 1)

Контрольные вопросы

1 Какое назначение элементов управления ComboBox и ListBox?
2 Назовите основные свойства элементов управленияComboBox и ListBox.
3 Понятие события. Как происходит обработка событий формы ComboBox и ListBox?

Лабораторная работа № 14. Использование элементов управления ScrolBar, SpinButton

Цель работы: изучить свойства элементов управления ScrolBar, SpinButton.

Методические указания

Элемент управления ScrollBar (полоса прокрутки) обладает рядом свойств. Приведем наиболее часто используемые свойства элемента управления ScrollBar (таблица 21).

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.

Таблица 21 – Свойства элемента управления ComboBox

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)
SrnallCha nge	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)

Задание

На основе созданного в лабораторной работе № 13 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.

Контрольные вопросы

1 Какое назначение элементов управления ScrolBar и SpinButton?
2 Назовите основные свойства элементов управления ScrolBar, SpinButton.
3 Понятие события. Как происходит обработка событий элементов формы ScrolBar, SpinButton?

Лабораторная работа № 15. Использование элементов управления MultiPage, TabStrip

Цель работы: изучить свойства MultiPage, TabStrip.

Методические указания

Элемент управления MultiPage (набор страниц) реализует многостраничные диалоговые окна. Заголовки страниц отображаются на вкладках. Переход от страницы к странице осуществляется выбором вкладки посредством щелчка кнопкой мыши.

Создать, переименовать, удалить или переместить страницу элемента управления MultiPage можно вручную, выбрав ярлык соответствующего листа и вызвав щелчком правой кнопки мыши контекстное меню. Используя это контекстное меню, можно произвести одно из перечисленных действий.

Объект MultiPage содержит в себе семейство Pages, являющееся набором всех страниц, входящих в этот объект. Далее приведены свойства объекта MultiPage (таблица 22).

Таблица 22 – Свойства элемента управления MultiPage

Свойство	Описание
Value и BoundValue	Возвращают номер активной страницы. Нумерация производится с нуля
MultiRow	Допустимые значения: True (если ярлыки не помещаются в одну строку, то они выводятся в несколько строк) и False (если ярлыки не помещаются в одну строку, то появляется полоса прокрутки, позволяющая переходить от страницы к странице)
Selectedltem	Возвращает выбранную страницу

Семейство Pages, содержащее все страницы, входящие в объект MultiPage, имеет единственное свойство Count, возвращающее число элементов семейства. Кроме того, у семейства Pages имеются следующие методы (таблица 23).

Элемент управления TabStrip (набор вкладок) позволяет создать несколько вкладок в диалоговом окне. Объект TabStrip содержит в себе семейство Tabs, представляющее собой набор всех вкладок. Объект TabStrip и семейство Tabs обладают теми же свойствами и методами, что и объект MultiPage и семейство Pages.

Таблица 23 – Свойства элемента управления MultiPage

Свойство	Описание
Add	Создает новую страницу. Синтаксис: Set Object = object.Add([Name [, Caption [, index]]]) object – семейство Pages Name – имя страницы Caption – текст, отображаемый на ярлыке страницы index – номер страницы, нумерация страниц производится с 0
Clear	Удаляет все страницы из семейства Pages
Remove	Удаляет страницу из семейства Pages
Item	Возвращает страницу со специфицированным индексом. Синтаксис: Set Object = object.Item(collectionindex)

Задание

На основе созданного в лабораторной работе № 13 приложения необходимо обеспечить выбор одной из трех фигур через MultiPage (четные варианты) и TabStrip (нечетные варианты), произвести расчет площади (варианты 1–10) или объема (варианты 11–20) выбранной фигуры.

Контрольные вопросы

1 Какое назначение элементов управления MultiPage, TabStrip?
2 Назовите основные свойства элементов управления MultiPage, TabStrip.

3 Понятие события. Как происходит обработка событий элементов формы MultiPage, TabStrip?

Лабораторная работа № 16. Создание диалоговых окон пользователя. Инициализация и отображение диалогового окна

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

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 CommandButton2 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

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.

Контрольные вопросы

1 Как создать графический интерфейс своего приложения с помощью VBA?
2 Назовите самые важные свойства и методы форм.
3 Назовите основные свойства элементов управления.
4 Понятие события. 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 **Богданова, С. В.** Информационные технологии / С. В. Богданова. – Ставрополь : Сервисшкола, 2014. – 211 с.

2 **Голицына, О. Л.** Информационные системы : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2-е изд. – Москва : ИНФРА-М, 2014. – 448 с. : ил.

3 **Гагарина, Л. Г.** Информационные технологии : учебное пособие / Л. Г. Гагарина. – Москва : ИНФРА-М, 2015. – 320 с.

4 **Федотова, Е. Л.** Информационные технологии и системы : учебное пособие / Е. Л. Федотова. – Москва : ИНФРА-М, 2014. – 352 с.

5 **Федотова, Е. Л.** Информационные технологии в науке и образовании : учебное пособие / Е. Л. Федотова. – Москва : ИНФРА-М, 2015. – 336 с.

6 **Черников, Б. В.** Информационные технологии управления : учебник / Б. В. Черников. – Москва : ИНФРА-М, 2017. – 368 с.