МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ДГТУ)
Факультет Информатики и вычислительной техники
Кафедра Математики и информатики

Мисюра В.В.

ОСНОВЫ ПРОГРАММИРОВАНИЯ В MS EXCEL

Учебное пособие

Ростов-на-Дону
2019

УДК 618.3.06

Основы программирования на VBA. Методическое пособие
— Ростов-на-Дону, 148 с.

В данном пособии рассматриваются основы процедурного и объектно-ориентированного программирования в среде VBA for Excel. Пособие содержит примеры программ, упражнения для самостоятельного выполнения, лабораторные работы, варианты контрольной работы. Предназначено для студентов экономических специальностей.

Составители: к.ф.-м.н. В.В. Мисюра

Содержание

Введение 3
1. Основные понятия 6
1.1. Объекты 8
1.2. Методы 10
1.3. Свойства. 10
1.4. События 10
2. Работа с объектами Range 11
3. Редактор Visual Basic 11
4. Процедуры и функции 14
5. Допустимые имена 15
6. Типы данных 16
7. Переменные и константы 17
8. Директива Option Explicit 20
9. Область видимости переменной 21
11. Встроенные функции VBA 22
12. Оператор присваивания 25
13. Оператор With – End With 26
14. Комментарий. Перенос строки кода. Расположение нескольких операторв в одной строке . 26
15. Операторы ввода-вывода 27
16. Оператор ветвления 32
17. Оператор выбора Select Case 33
Упражнения 1 34
18. Оператор цикла For . . . Next 34
19. Цикл Do 37
20. Оператор For Each 40
Упражнения 2 41
21. Оператор безусловного перехода GoTo 43
22. Перехват и обработка ошибок 44
23. Введение в процесс разработки приложений 45
Лабораторная работа №1 48
Лабораторная работа №2 55
Лабораторная работа № 3 63
Лабораторная работа №4 70

Лабораторная работа №5 78
Лабораторная работа № 6 83
Лабораторная работа №7 86
Лабораторная работа №8 98
Лабораторная работа №9 102
Лабораторная работа №10 119
Лабораторная работа №11 125
Лабораторная работа №12* 131
Примеры вопросов теста по программированию 139
Контрольная работа 141
Литература 147

Введение

В настоящее время в самых разных областях человеческой деятельности информационные технологии занимают одно из ведущих мест. Применение компьютера значительно повышает производительность труда. Любые современные технологии, в том числе и компьютерные, содержат в своей основе широкий слой теоретических знаний, овладение которыми позволяет в совершенстве пользоваться этими самыми технологиями. Уровень развития программных средств достиг такого уровня, что от пользователя ПК требуется не только умение нажимать на клавиши, но и понимание хотя бы основных процессов, происходящих внутри компьютера и при работе различных программ. Современный компьютер — сложная машина, продукт развитого человеческого интеллекта, поэтому, очевидно, он и рассчитан на взаимодействие с квалифицированным пользователем.

В процессе использования в профессиональной деятельности компьютера перед пользователем возникают проблемы. Первая проблема заключается в том, что пользователю надоедает работать в рамках предлагаемого ему приложения, многократно нажимать на одни и те же клавиши при решении привычной для него задачи и возникает естественное желание автоматизировать работу приложения. Вторая проблема (более общего плана) связана с выбором между посторонней узконаправленной программой и "своим" адаптированным для конкретной задачи приложением. Как правило, пользователи, работающие в организациях, направленных на постоянное развитие, выбирают второй вариант. И, наконец, третья, объективно существующая проблема – это постоянное развитие интеллектуальных и информационных технологий, когда приходится быть в курсе всех новинок предлагаемых в компьютерном мире.

Таким образом, с течением времени, квалифицированный пользователь сталкивается с необходимостью совершенствовать свои знания в области информационных технологий.

Как показывает практика, хорошо разработанными и наиболее популярными на сегодняшний день являются компоненты семейства офисных приложений Microsoft Office. Это связано с тем, что они обладают очень широким диапазоном средств для ввода, анализа и представления данных. Эти средства являются не только простыми и удобными, но и высокопродуктивными, что обеспечивает высокую скорость разработки приложений.

Однако не все задачи, возникающие в практике разработки офисных приложений, можно решить, пользуясь только интерфейсными средствами Microsoft Office. Чтобы приложения, создаваемые в среде компонентов Microsoft Office, сделать удобными для пользователей и функционирующими эффективно, рационально использовать язык Visual Basic for Applications (VBA).

VBA позволяет создавать приложения, выполняемые в среде Microsoft Office. Это могут быть разнообразные аналитические программы, финансовые системы, программы учета кадров, системы автоматического создания официальных писем/документов с помощью библиотеки готовых шаблонов и т.п. При этом интерфейс создаваемой программы может быть совершенно непохожим на интерфейс того приложения, в котором она написана.

Можно выделить основные этапы компьютерного решения задач.

1. **Постановка задачи**. Основное требование к постановке задачи – достаточное количество информации для решения задачи.

2. **Моделирование и формализация задачи**. **Формализация** предполагает замену словесной формулировки решаемой задачи краткими символьными обозначениями, близкими к обозначениям в языках программирования или к математическим. **Моделирование** задачи является важнейшим этапом, целью которого является поиск общей концепции решения. Обычно моделирование выполняется путем выдвижения гипотез решения задачи и их проверке любым рациональным способом (прикидочные расчеты, физическое моделирование и т.д.). Результатом каждой проверки является либо принятие гипотезы, либо отказ от нее и разработка новой. К разработке алгоритма следует приступать только после принятия гипотезы решения задачи. Помимо идеи решения задачи, результатами этого этапа должны быть формализованная постановка задачи типа "дано-найти" и достаточное количество контрольных примеров для последующего тестирования программы. К категории "Дано:" обычно относятся данные, вводимые в начале работы программы и обеспечивающие массовость алгоритма. К категории "Найти:" относятся данные, получаемые в результате работы программы.

3. **Разработка алгоритма**. Этот этап представляет собой реализацию идеи решения задачи.

4. **Тестирование алгоритма**. Этап предполагает проверку алгоритма вручную с использованием подготовленных ранее контрольных примеров. Для сложных задач этот этап может оказаться весьма трудоемким, поэтому опытные программисты пропускают его и тестируют программу.

5. **Программирование алгоритма**. Программирование является формальной записью алгоритма средствами языка программирования.

6. **Тестирование программы**. Тестирование выполняется путем вывода промежуточных результатов работы программы и сравнения их с контрольным примером. Для этого либо используют специальные средства отладки программ, имеющиеся в интегрированной среде языка программирования, либо временно добавляют в программу команды вывода промежуточных значе-

ний. Уменьшить трудоемкость поиска ошибок в программе можно более тщательным проектированием алгоритма и планированием процесса тестирования на ранних стадиях разработки программы.

7. **Эксплуатация программы и интерпретация результатов**. В сложных программах может быть недостаточно тестирования для устранения всех ошибок. Очень часто они обнаруживаются на стадии эксплуатации.

1. Основные понятия

Программа — это детальное и законченное описание алгоритма средствами языка программирования. Исполнителем программы является компьютер. Для выполнения компьютером программа должна быть представлена в машинном коде — последовательности чисел, понимаемых процессором. Написать программу в машинных кодах вручную достаточно сложно. Поэтому сегодня практически все программы создаются с помощью языков программирования, которые по своему синтаксису и семантике приближены к естественному человеческому языку. Это снижает трудоемкость программирования. Однако, текст программы, записанный с помощью языка программирования, должен быть преобразован в машинный код. Эта операция выполняется автоматически с помощью специальной служебной программы, называемой **транслятором**.

Трансляторы делятся на два типа: **интерпретаторы** и **компиляторы**.

Интерпретатор переводит в машинный код и выполняет очередной оператор (команду) программы. Если команда повторяется, то интерпретатор рассматривает ее как встреченную впервые.

Компилятор переводит в машинный код исходный текст программы целиком. Поэтому достоинство компиляторов — быстродействие и автономность получаемых программ. Достоинство интерпретаторов — их компактность, возможность остановить в любой момент выполнение программы, выполнить различные преобразования данных и продолжить работу программы.

В общем случае для создания программ нужно иметь следующие компоненты

- текстовый редактор — для набора исходного текста программы;
- компилятор — для перевода текста программы в машинный код;
- редактор связей — для сборки нескольких откомпилированных модулей в одну программу;
- библиотеки функций — для подключения стандартных функций к программе.

Современные системы программирования включают в себя все указанные компоненты и называются **интегрированными системами**.

Исходный текст программы можно получить без записи его вручную в текстовом редакторе. Существуют системы визуального программирования — **RAD**-среды (Rapid Application Development), которые, не исключая возможности записи программы вручную, позволяют создавать текст программы автоматически, путем манипуляций со стандартными элементами управления, включенными в RAD-среду. Поэтому для RAD-среды понятие «программирование» часто заменяют понятием «проектирование».

По способу разработки программ можно выделить два подхода:

- **процедурное программирование** — это программирование, при котором выполнение команд программы определяется их последовательностью, командами перехода, цикла или обращениями к процедурам;

- **объектно-ориентированное программирование** – программирование, при котором формируются программные объекты, имеющие набор свойств, обладающие набором методов и способные реагировать на события, возникающие как во внешней среде, так и в самом объекте (нажатие мыши, срабатывание таймера, превышение числовой границы и т.д.). Таким образом, выполнение той или иной части программы зависит от событий в программной системе.

Объектно-ориентированное программирование (ООП) не исключает, а охватывает технологию процедурного программирования.

Концепция ООП возникла в середине 70-х годов. Главная ее идея в том, что программное приложение, как и окружающий нас мир, должно состоять из объектов, обладающих собственными свойствами и поведением. ООП объединяет исполняемый код программы и ее данные в объекты, что упрощает создание сложных программных приложений. Например, можно организовать коллективную работу над проектом, где каждый участник создает собственный класс объектов, который становится доступным другим участникам проекта.

При объектно-ориентированном подходе программные задачи распределяются между объектами программы. Объекты обладают определенным набором свойств, методов и способностью реагировать на события (нажатие кнопок мыши, интервалы времени и т.д.). В отличие от процедурного программирования, где порядок выполнения операторов программы определяется порядком их следования и командами управления, в ООП порядок выполнения процедур и функций определяется, прежде всего, событиями.

Чтобы проект можно было считать объектно-ориентированным, объекты должны удовлетворять некоторым требованиям. Этими требованиями являются **инкапсуляция**, **наследование** и **полиморфизм**.

- **Инкапсуляция** — означает, что объекты скрывают детали своей работы. Инкапсуляция позволяет разработчику объекта изменять внутренние принципы его функционирования, не оказывая никакого влияния на пользователя объекта.

- **Наследование** — означает, что новый объект можно определить на основе уже существующих объектов, при этом он будет содержать все свойства и методы родительского. Наследование полезно, когда требуется создать новый объект, обладающий дополнительными свойствами по сравнению со старым.

- **Полиморфизм** — многие объекты могут иметь одноименные методы, которые могут выполнять разные действия для разных объектов.
  Например, оператор "+" для числовых величин выполняет сложение, а для текстовых — склеивание.

Visual Basic for Applications (VBA) – интегрированная система программирования для создания прикладных программ в среде Microsoft Office. Следует заметить, что VB не поддерживает наследования в строгом смысле этого понятия. Инкапсуляция реализуется, в основном, за счет применения описаний Private и Public.

С помощью VBA можно создавать *объекты управления* графического интерфейса пользователя, задавать и изменять свойства объектов, подключать к ним соответствующий программный код. Методика программирования с использованием средств VBA сводится к следующему:

- создание объектов управления и контроля (диалоговые окна, пиктограммы, меню);
- разработка процедур, используемых при вызове объектов.

Прикладные программы на языке VBA оперируют со следующими понятиями: объекты, методы, свойства, события.

1.1. Объекты

Объект — набор данных вместе с кодом, предназначенным для их обработки. К объектам относятся экранные формы, графические элементы внутри форм, в том числе текстовые окна, линейки прокрутки, пиктограммы, окна-списки, командные кнопки и др.

Объектная библиотека VBA располагает более 100 различных объектов, находящихся на различных уровнях иерархии. Иерархия определяет связь между объектами и показывает пути доступа к ним.

На вершине **объектной** модели находится объект Application — в данном случае Excel. Но если вы программируете в VBA, запуская Microsoft Word, то объектом Application будет выступать Word.

Приведем несколько примеров объектов, которые находятся в объекте Application:

Windows (коллекция всех объектов Window - окон);
AddIns (коллекция всех объектов AddIn - надстроек);
Workbooks (коллекция всех объектов Workbook - рабочих книг).

Некоторые объекты могут содержать другие объекты. Например, коллекция WoorkBooks состоит из всех открытых объектов Workbook, а объект Workbook включает другие объекты, некоторые из них представлены ниже:

WorkSheets (коллекция объектов рабочих листов);
Charts (коллекция объектов Chart – диаграмм);
Names (коллекция объектов Name – имен).

Каждый из этих объектов, в свою очередь, может содержать другие объекты. Коллекция Worksheets состоит из всех объектов Worksheet рабочей книги Workbook. Объект Worksheet включает другие объекты, среди которых следующие:

ChartObjects (коллекция объектов ChartObject – элементов диаграмм);
Range — диапазон;
PageSetup — параметры страницы;
PivotTables (коллекция объектов PivotTable — сводных таблиц).

Одной из ключевых концепций в программировании на языке VBA являются **коллекции**. Коллекция — это группа объектов одного класса (и сама коллекция тоже является объектом). Как указывалось выше, Workbooks — это коллекция всех открытых в данный момент объектов Workbook. Вы можете одновременно управлять целой коллекцией объектов или отдельным объектом этой коллекции. Чтобы сослаться на один объект из коллекции, введите название или номер объекта в скобках после названия коллекции: WorkSheets(″Лист1″)

Если лист Лист1 — это первый рабочий лист в коллекции, то можно использовать следующую ссылку: WorkSheets(1)

Полная ссылка **на объект** состоит из ряда имен вложенных последовательно друг в друга объектов. Разделителями имен объектов в этом ряду являются точки, ряд начинается с объекта Application и заканчивается именем самого объекта.

Например, полная ссылка на ячейку A1 рабочего листа Лист1 рабочей книги с именем Архив имеет вид:

```
Application.Woorkbooks("Архив").Worksheets("Лист1").Range("A1")
```

Приводить каждый раз полную ссылку на объект совершенно не обязательно. Обычно достаточно ограничиться только неявной ссылкой на объект.

В неявной ссылке, в отличие от полной, объекты, которые активны в данный момент, как правило, можно опускать. В рассмотренном случае, если ссылка на ячейку A1 дана в программе, выполняемой в среде Excel, то ссылка на объект Application может быть опущена, т. е. достаточно привести относительную ссылку:

```
Woorkbooks("Архив").Worksheets("Лист1").Range("A1")
```

Если рабочая книга Архив является активной, то ссылку можно записать еще короче:

```
Worksheets("Лист1").Range("A1")
```

Если и рабочий лист Лист1 активен, то в относительной ссылке вполне достаточно ограничиться упоминанием только диапазона A1:
Range("A1")

1.2. Методы

Объект сам по себе не представляет большого значения. Намного значительнее то, какие действия можно совершать над объектом, и какими свойствами он обладает. **Метод** как раз и представляет собой действие, выполняемое над объектом.

Синтаксис применения метода:

Объект.Метод

Например, при помощи метода Quit: (закрыть) закрывается приложение:

```
Application.Quit
```

Метод можно применять ко всем объектам семейства. Например, к семейству ChartObjects (диаграммы) рабочего листа Лист1 применен метод Delete (удалить), который приводит к удалению всех диаграмм с рабочего листа

```
Worksheets("Лист1").ChartObjects.Delete
```

1.3. Свойства.

Свойство представляет собой атрибут объекта, определяющий его характеристики, такие как размер, цвет, положение на экране и состояние объекта; например, доступность или видимость. Чтобы изменить характеристик объекта, надо просто изменить значения его свойств.

Синтаксис установки значения свойства:

Объект.Свойство = ЗначениеСвойства

В следующем примере изменяется заголовок окна Excel посредством задания свойства Caption объекту Application:

```
Application.Caption = "Пример"
```

Свойство можно изменять сразу у всех объектов коллекции. В приведенном ниже примере с помощью установки свойству Visible (видимость) значения False (ложь) все рабочие листы активной книги (коллекция объектов) скрываются:
```
WorkSheets.Visible=False
```

1.4. События

Событие представляет собой действие, распознаваемое объектом (например, щелчок мышью или нажатие клавиши), для которого можно запрограммировать

отклик. События возникают в результате действий пользователя или программы, или же они могут быть вызваны системой.

Специальный вид процедур, генерирующих отклик на события, называется *процедурами обработки событий.* В целом программирование на VBA состоит в создании кода программ, которые генерируют прямо или косвенно отклики на события.

Программы на языке VBA для приложений, функционирующих в среде Excel, создаются двумя способами:

- в автоматическом режиме как результат построения клавишной макрокоманды *(см. лаб.раб №3*);
- в неавтоматическом режиме путем создания программного кода.

2. Работа с объектами Range

В основном, работа, которая выполняется в VBA, связана с управлением ячейками и диапазонами на рабочих листах, что и является основным предназначением электронных таблиц. Объект Range содержится в объекте Sheets и состоит из одной ячейки или диапазона ячеек на отдельном рабочем листе.

К ссылке на объект Range обращаются с помощью нескольких вариантов синтаксиса. Если в диапазоне указываются только имена столбцов или строк, то объект Range задает диапазон, состоящий из указанных столбцов или строк. Например, Range("A:C") задает диапазон, состоящий из столбцов A, B и C, а Range("2:2") — из второй строки. Другим способом работы со строками и столбцами являются методы Rows (строки) и Columns (столбцы), возвращающие коллекции строк и столбцов. Например, столбцом A является Columns(1), а второй строкой — Rows(2).

Так как ячейка является частным случаем диапазона, состоящим только из единственной ячейки, объект Range также позволяет работать с ней. Объект Cells (ячейки) — это альтернативный способ работы с ячейкой. Например, ячейка A2 как объект описывается Range("A2") или Cells(1,2). В свою очередь объект Cells, вкладываясь в Range, также позволяет записывать диапазон в альтернативном виде, который иногда удобен для работы, а именно, Range("A2:C3") и Range(Cells(1,2),Cells(3,3)) определяют один и тот же диапазон.

3. Редактор Visual Basic

Редактор VBA **активизируется** из Excel одним из следующих способов:

- нажать <Alt+F11> ;

- выбрать вкладку Разработчик/кнопка Visual Basic (в случае отсутствия вкладки Разработчик см. Приложение 1).

Интерфейс VBA состоит из следующих основных компонентов: окна проекта (Project Explorer), окна свойств, окна редактирования кода, окна форм, панели инструментов, строки меню (см. рис.1, рис.2).

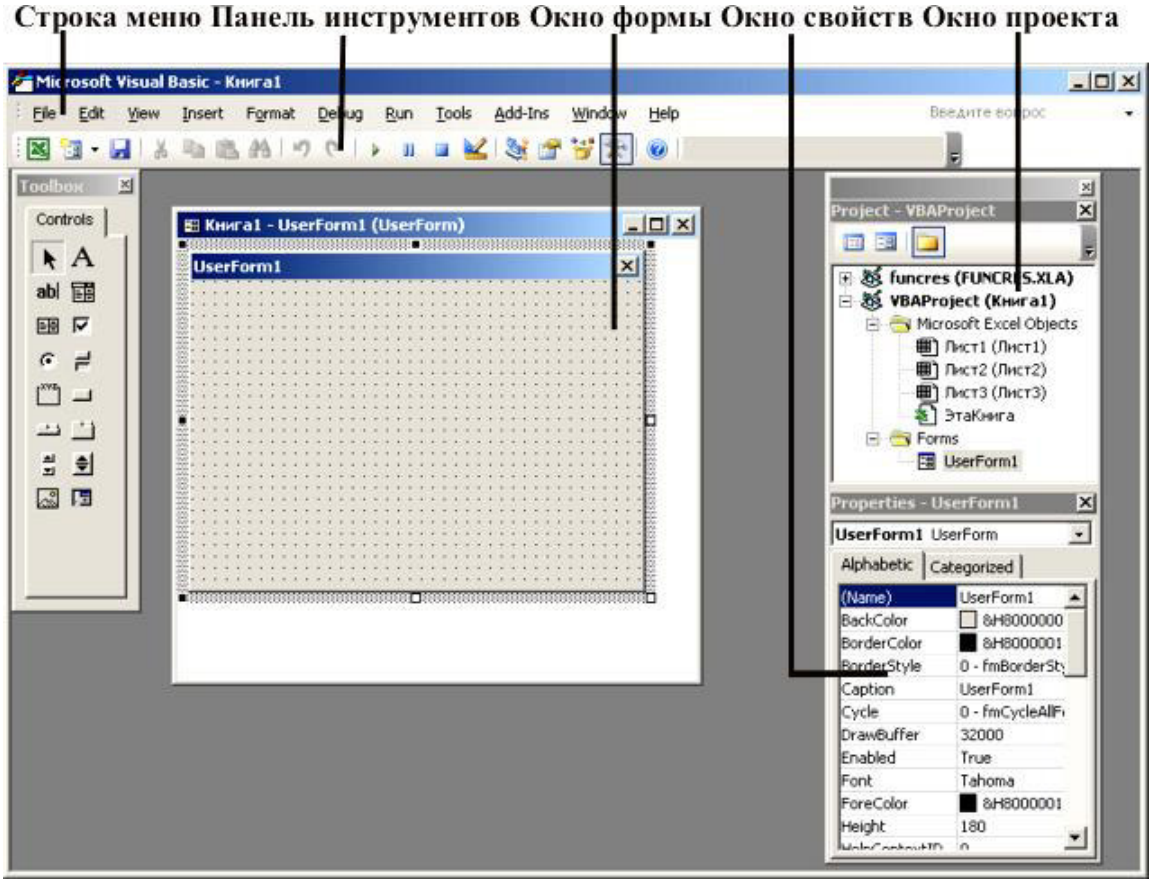


Рисунок 1. Редактор VBA

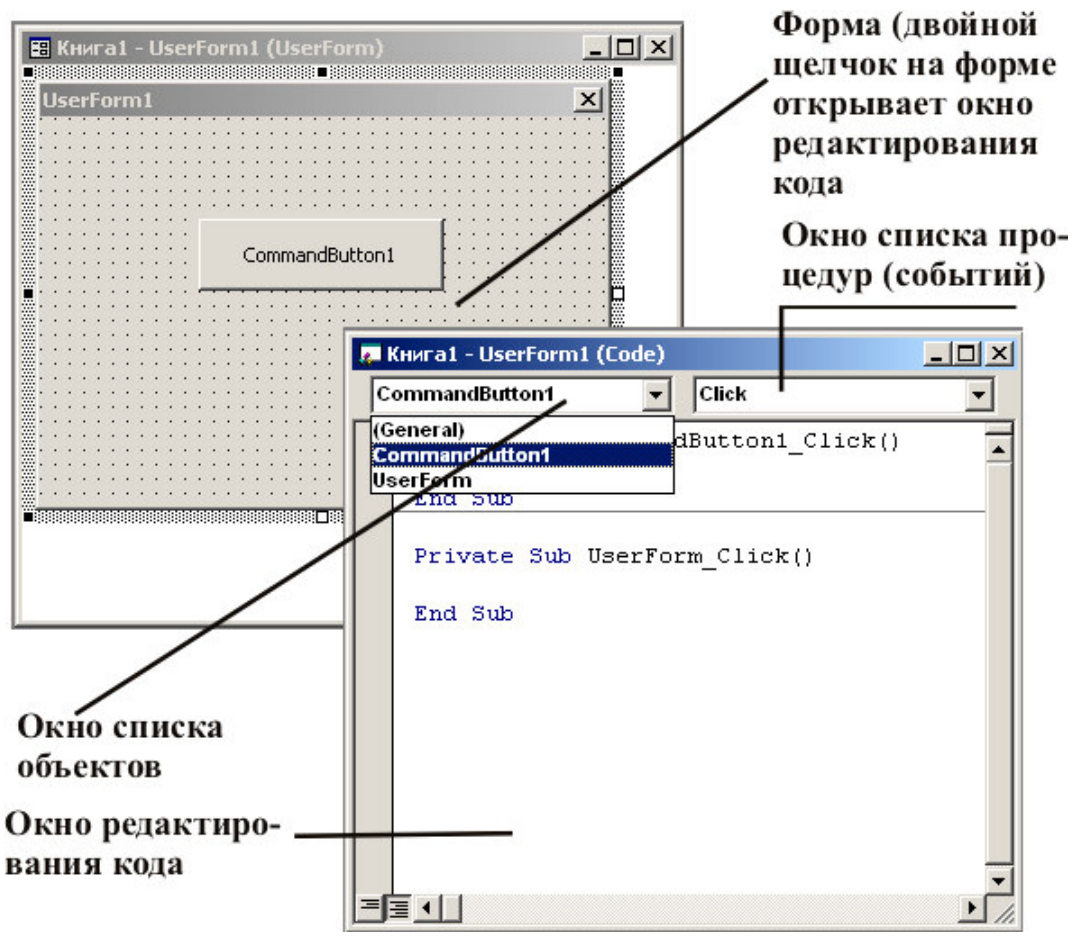


Рисунок 2. Компоненты редактора VBA

Два раскрывающихся списка в верхней части окна редактора кода облегчает ориентацию в процедурах. В списке объектов перечислены элементы управления, помещенные в форму, сама форма. Список удобно использовать для быстрого доступа к коду, связанному с соответствующим объектом. В списке процедур перечислены события, которые связываются с данным объектом. Список процедур используется для быстрого доступа к нужным событиям и процедурам.

Основой программ на VBA являются процедуры и функции, состоящие из инструкций. Каждая процедура имеет имя, по которому она вызывается на выполнение.

Программы на языке Visual Basic в среде Excel хранятся в модулях. **Модуль** является структурой, сохраняющей некоторый набор описаний и процедур, или способом организации процедур.

В проекте автоматически создается модуль для каждого рабочего листа и для всей книги, кроме того, модули создаются для каждой пользовательской формы, макросов и классов.

По своему предназначению модули делятся на два типа: модули объектов и стандартные модули.

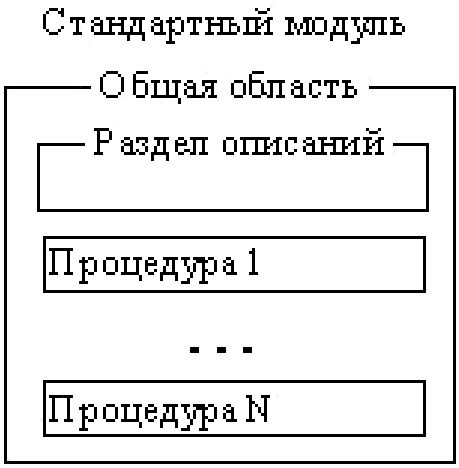


Рисунок 3. Структура стандартного модуля

Стандартные модули являются отдельными объектами Excel. В них хранятся процедуры, доступные из любых других объектов проекта Excel (см. рис. 3). Вызов этих процедур может осуществляться из процедур обработки событий, процедур других стандартных модулей, макросов и просто из выражений.

К модулям объектов относятся модули, связанные с рабочей книгой, рабочими листами, формами и модулями класса.

Двойной щелчок на значке объекта в окне проекта (рабочей книге, рабочем листе, модуле, форме и т.д.) в редакторе открывает окно редактора кода для соответствующего модуля.

4. Процедуры и функции

Процедуры имеют следующий синтаксис:

[Private | Public] Sub <Имя>([Формальные аргументы])
<Тело процедуры>
End Sub

Здесь и далее угловые скобки (< >) содержат пояснения, на место которых должны быть подставлены реальные текстовые конструкции, соответствующие синтаксическим правилам языка. Квадратные скобки означают необязательность применения записанных в них служебных слов. Вертикальная черта означает возможность выбора одного из служебных слов.

Вызов процедуры общего назначения выполняется по имени:

<Имя>([Фактические аргументы])

При вызове процедуры фактические аргументы подставляются на место формальных и управление выполнением передается процедуре. Аргументы могут

быть входными, выходными или модифицируемыми. Через входные аргументы процедура получает данные при обращении к ней. Выходные аргументы возвращают результаты выполнения процедуры. Модифицируемые аргументы являются одновременно входными и выходными.

Функция общего назначения построена так же, как процедура, однако, результат работы функции передается (возвращается) через ее имя. Поэтому, обращения к функциям можно использовать внутри арифметических и логических выражений. Синтаксис функции:

[Private | Public] Function <Имя>(<Аргументы>)[As Тип]
<Тело функции>
End Function

Для того чтобы функция возвращала результат через имя, в теле функции должна присутствовать хотя бы одна команда присваивания типа

<Имя>=<выражение>

Служебные слова **Private** и **Public** задают область видимости процедур и функций. **Private** делает объект доступным только внутри данного модуля. **Public** делает объект доступным из других модулей.

5. Допустимые имена

В VBA пользователь определяет имена переменных, функций, процедур, типов, констант и других объектов. Вводимые пользователем имена должны отражать суть обозначаемого объекта так, чтобы делать программу легко читаемой. В VBA имеются следующие ограничения на имена:

- длина имени не должна превышать 255 символов;
- имя не может содержать точек, пробелов и следующих символов: %, &,#,@, $;
- имя может содержать любую комбинацию букв, цифр и символов, начинающуюся с буквы;
- имена должны быть уникальны внутри области, в которой они определены;
- не следует использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

Хотя верхний или нижний регистр букв в имени не имеет значения, умелое использование его может существенно облегчить понимание содержательной стороны переменной.

Например, вместо плоских и невыразительных имен

```
Firstname, x_initialvalue
```

предпочтительнее использовать имена, которые легко читаются с одного взгляда, благодаря акцентированию входящих в них компонентов разумным использованием верхнего регистра букв: **`FirstName, X_InitialValue.`**

6. Типы данных

Типы данных относятся к самим фундаментальным понятиям любого языка. Тип данных определяет множество допустимых значений, которое может принимать указанная переменная. В табл. 1 перечислены типы данных, используемых в VBA.

Таблица 1. *Типы данных, используемые в VBA*

Тип данных	Размер (байт)	Диапазон значений
Byte(байт)	1	От 0 до 255
Boolean (логический)	2	True или False
Integer(целое)	2	От -32 768 до 32 767
Long (длинное целое)	4	От -2 147 483 648 до 2 147 483 647
Single (с плавающей точкой обычной точности)	4	От –3,402823E38 до –1,401298E-45 для отрицательных значении; от 1,401298E-45 до 3,402823E38 для положительных значений;
Double (с плавающей точкой двойной точности)	8	От -1,79769313486232E308 до -4,94065645841247E-324 для отрицательных значений; от 4,94065645841247E-324 до 1,79769313486232E308 для положительных значений
Currency (денежный)	8	От -922 337 203 685 477,5808 до 922 337 203 685 477,5807
Date (даты и время)	8	От 1 января 100 г. до 31 декабря 9999 г.
Object (объект)	4	Любой указатель обьекта
String (строка переменной длины)	10 + длина строки	От 0 до приблизителыно 2 миллиардов
String (строка постоянной длины)	Длина строки	От 1 до приблизительно 65 400
Variant	16 + 1 байт на каждый символ	Значение любого из выше перечисленных типов данных
Тип данных, определяемый пользова-	Объем определяется элемен-	Диапазон каждого элемента определяется его типом данных

Тип данных	Размер (байт)	Диапазон значений
телем	тами	

Данные типа Variant могут иметь особое значение Null, которое означает, что данные отсутствуют, неизвестны или неприменимы. Например, по умолчанию данные в полях таблицы базы данных имеют тип Variant. Поэтому, если оставить поле пустым, ему будет присвоено значение Null.

7. Переменные и константы

В VBA переменная (variable) используется для временного хранения данных в оперативной памяти, т. е. она идентифицирует область в памяти, где хранится некоторая информация. После объявления переменная указывает на одну и ту же область памяти до тех пор, пока не будет уничтожена. Программист не должен указывать явное место хранение данных, ему для ссылки на область памяти достаточно лишь указать на имя.

Уведомить VBA о существовании переменной, или объявить переменную можно двумя способами: явным и неявным.

Явное объявление означает, что переменная должна быть объявлена прежде, чем ее можно использовать, и оно производится при помощи операторов **Dim**, **Private**, **Static**, **Public**, которые также определяют и область видимости переменной:

Dim|Static|Private|Public ***varname1*** **[As type1][,** ***varname2*** **[As type2]]…**

varname — имя переменной, удовлетворяющее стандартным правилам именования переменных;
type— тип данных переменной. Для каждой описываемой переменной следует использовать отдельное предложение *As type* .

Явное объявление каждой переменной делает программу надежнее и, кроме того, убыстряет ее работу, т. к. VBA не требуется тратить время на распознавание типа неописанной переменной при каждом обращении к ней.

Например, следующая инструкция объявляет переменную типа integer

```
Dim M As Integer
```

Если тип данных при объявлении переменной опущен, то по умолчанию переменная получает тип Variant.

В инструкции

```
Dim A, B As Single
```

переменная A объявлена как Variant, так как в VBA нельзя объявить тип данных для группы переменных, разделив переменные запятыми.

По умолчанию строковая или текстовая переменная (String) является массивом переменной длины, который содержит символы. Однако текстовая переменная может быть определена и фиксированной длины. В следующем примере объявляется символьный массив размером в 25 символов:

```
Dim A As String*25
```

В этом случае, если вы присвоите переменной A строку длиной более 25 символов, то она будет усечена.

Как и в других языках программирования, в VBA вы можете использовать ***массивы***. Под структурой данных типа ***массив*** понимают однородную структуру однотипных данных, одновременно хранящихся в последовательных ячейках оперативной памяти.

Одномерный массив — это однородная структура однотипных данных, для получения доступа к его элементам достаточно одной индексной переменной.

Двумерный массив — это структура однотипных элементов, расположенных в виде таблицы значений. Такое представление значений соответствует математическому понятию двумерный массив. Каждый элемент в двумерном массиве идентифицируется номером строки и номером столбца, на пересечении которых он расположен.

Для двумерного массива A(N,M) (состоящего из N строк и M столбцов) в обозначении элемента A(i,j) первое значение i соответствует номеру строки и изменяется от 1 до N, а j - номеру столбца и изменяется от 1 до M.

Примеры объявления массивов:

```
Dim B(3,3) As Single
Dim A(12) As Integer
```

Первая строка объявляет двухмерный массив 3x3 (матрицу), состоящий из действительных чисел (матрица имеет нулевой столбец и нулевую строку, состоит из 16 элементов). Вторая строка объявляет одномерный массив (вектор), состоящий из 13 целых чисел, причем по умолчанию первый элемент массива будет A(0), а последний — A(11). В этом случае говорят, что 0 — базовый индекс. Можно изменить базовый индекс, написав в начале модуля директиву Option Base 1. После этого индексы массивов A и B будут начинаться с единицы. Другим способом изменения базового индекса является использование ключевого слова ***To*** при объявлении массива, например

```
Dim B(1 To 3, 1 To 3) As Single
Dim A(1 To 12) As Integer
```

Массив в программе определяется поэлементно. Например:

```
Dim B(1 To 2, 1 To 2) As Integer
B(1,1)=3 : B(1,2)=-4
```

```
B(2,1)=0 : B(2,2)=7
```

Удобным способом определения одномерных массивов является функция Array, преобразующая список элементов, разделенных запятыми, в вектор из этих значений и присваивающая ему тип Variant, например

- инициализация одномерного массива

```
Dim Num As Variant
Num=Array(10,20)
```

- инициализация многомерного массива

```
Dim Num As Variant
Num=Array(Array(10,20), Array(30,40))
```

Иногда в процессе выполнения программы требуется изменять размер массива. В этом случае его объявляют как ***динамический***. Для этого при объявлении массива не нужно указывать размерность. Например:

```
Dim R( ) As Single
```

Затем в программе следует вычислить необходимый размер массива, присвоив его некоторой переменной, например **n**, и указать размер динамического массива с помощью оператора **ReDim**.

ReDim [Preserve] ***varname*****(indexes) [As type] [, varname(indexes) [As type]]…**

Preserve — ключевое слово, используемое для сохранения данных в существующем массиве при изменении значения последней размерности. Если ключевое слово Preserve не используется, то данные в массиве при изменении размерности не сохраняются;
varname — имя переменной, удовлетворяющее стандартным правилам именования переменных;
indexes— размерности переменной массива; допускается описание до 60 размерностей;
type — тип данных массива.

В следующем примере сначала объявляется динамический массив, а затем устанавливаются границы его индекса.

```
Dim R( ) As Integer
N=20
ReDim R(1 To N)
```

Допустимо повторное использование инструкции **ReDim** для изменения числа элементов и размерностей массива.

Хотя переменные обычно объявляются в начале процедуры или функции, они могут быть объявлены в любом месте до того, как в коде встречается первая ссылка на них.

Неявное объявление переменной осуществляется включением в конец его имени специального символа, устанавливающего тип переменной. В этом случае переменную не надо объявлять перед тем, как ее применять.

Для установки типов допустимо использование специальных символов, приведенных в табл. 2.

Таблица 2. *Специальные символы для установки типов*

Специальный символ	Тип	Специальный символ	Тип
%	Integer	#	Double
&	Long	@	Currency
!	Single	$	String

Константы, в отличие от переменных, не могут изменять свои значения. Использование констант делает программы легче читаемыми и проще исправляемыми, т. к. отпадает необходимость заменять многократно значения по тексту программы —достаточно ввести новое значение при объявлении константы.

[Public | Private] Const *ИмяКонстанты* [As Тип] = *Выражение*

Public — ключевое слово, используемое на уровне модуля для объявления констант, доступных всем процедурам во всех модулях. Не допускается в процедурах;
Private — ключевое слово, используемое на уровне модуля для объявления констант, доступных только внутри модуля, в котором выполняется описание. Не допускается в процедурах;
ИмяКонстанты — имя константы, удовлетворяющее стандартным правилам именования переменных;
Тип — один из поддерживаемых типов данных (см. табл.1). Для каждой объявляемой константы следует использовать отдельное предложение *As Тип;*
Выражение — другая константа или любое выражение, которое включает все арифметические или логические операторы за исключением *Is*.

В данном коде определяются числовая и строковая константы:

```
Const Индекс As Single = 0.2
Const Специальность = "Экономист"
```

8. Директива Option Explicit

Для обязательного объявления всех переменных, в так называемой области модуля *General Declarations* (разделе описаний, см. рис.3), надо поместить директиву *Option Explicit*. Использование этой директивы не допускает возможности неправильного ввода имени переменной, которая применяется в одной или нескольких процедурах модуля.

Например, если переменная была объявлена как Ставка, а в коде при наборе вместо русской буквы с была использована латинская буква c, то это приведет к ошибке. В отсутствие директивы *Option Explicit* подобную ошибку было бы трудно отследить.

9. Область видимости переменной

Область видимости переменной определяет, в каких модулях и процедурах она может использоваться. Типы областей действия переменных приведены в таблице 3.

Таблица3. *Типы областей действия переменных*

Область действия	Способ объявления переменной
Отдельная процедура	В процедуру включается оператор Dim или Static
Отдельный модуль	Перед первой процедурой в модуле вводится оператор Dim или Private
Все модули	Перед первой процедурой в модуле вводится оператор Public

Локальная переменная — это переменная объявленная в процедуре. Локальные переменные могут использоваться только в процедуре, в которой они объявлены. После выполнения процедуры переменная становится невостребованной, поэтому освобождается соответствующая область памяти.

В случае если необходимо, чтобы переменная была доступна во всех процедурах модуля, нужно объявить ее перед первой процедурой модуля.

Например:

```
Dim CurrentValue As Integer
Sub MySub()
    [Текст процедуры]
End Sub
Sub YourSub()
    [Текст процедуры]
End Sub
```

Значение переменной модуля не изменяемся при окончании выполнения процедуры.

Чтобы сделать переменную доступной во всех процедурах всех модулей (переменная называется глобальной), необходимо объявить переменную на уровне модуля с помощью ключевого слова **Public**, а не **Dim**. Такое объявление переменной может вводится только в стандартном модуле VBA, а не в коде модуля листа или формы.

Переменные **Static** объявляются на уровне процедуры и сохраняют свое значение после окончания процедуры.

10. Операции VBA

В программах на VBA можно использовать стандартный набор операций над данными. Имеются три основные типа операций:

- математические, выполняются над числами, и их результатом являются числа;
- отношения, применяются не только к числам, и их результатом являются логические значения, например x>y;
- логические, используются в логических выражениях и их результатом являются логические значения, например Not x And y.

Операции имеют определенный приоритет выполнения (табл. 4).

Таблица 4. *Основные операции языка VBA*

Приоритет	Операция	Название операции
1	Вызов функции и скобки	
2	^	Возведение в степень
3	-	Смена знака
4	*	Умножение
4	/	Деление
5	\	Целочисленное деление
6	Mod	Остаток от деления по модулю
7	+	Сложение
7	-	Вычитание
8	>, <, >=, <=, <>, =	Операции отношения
9	Not	Логическое отрицание
10	And	Логическое умножение
11	Or	Логическое сложение
12	Xor	Исключающее Or
13	Equ	Логическая эквивалентность
14	Imp	Логическая импликация

11. Встроенные функции VBA

В VBA имеется большой набор встроенных функций и процедур, использование которых существенно упрощает программирование. Эти функции можно разделить на следующие основные категории:

- математические функции;
- функции проверки типов;
- функции преобразования форматов;
- функции обработки строк;
- функции времени и даты.

Приведем описание некоторых процедур и функций, которое поможет отыскать необходимую процедуру или функцию, а полное описание и правильное использование процедуры или функции можно найти в справке VBA или воспользоваться интеллектуальными возможностями редактора кода.

Таблица 5. *Краткое описание некоторых процедур и функций*

Название	Описание
Abs (функция)	возвращает абсолютное значение числа
Asc (функция)	возвращает числовой код первого символа строки аргумента
Atn (функция)	возвращает арктангенс числа в радианах
CBool (функция)	приводит выражение к типу Boolean
CByte (функция)	преобразует выражение к типу Byte
CCur (функция)	преобразование выражения к типу Currency
CDate (функция)	преобразование выражения к типу Date
CDbl (функция)	преобразование к типу Double
Chr (функция)	возвращает символ, связанный с определенным числовым кодом
CInt (функция)	преобразование выражения к типу Integer
CLng (функция)	преобразование выражения к типу Long
Cos (функция)	возвращает косинус числа
CSng (функция)	преобразование выражения к типу Single
CStr (функция)	преобразование выражения к типу String
CVar (функция)	преобразование выражения к типу Variant
Date (оператор)	устанавливает значение системной даты
Date (функция)	возвращает значение системной даты
DateAdd (функция)	возвращает переменную типа Variant, содержащую дату, отличающуюся от заданной на определенный интервал времени
DateDiff (функция)	возвращает число временных интервалов между двумя датами
DatePart (функция)	возвращает определенную часть заданной даты
DateSerial (функция)	возвращает дату для заданного года, месяца и дня
DateValue (функция)	возвращает дату
Day (функция)	возвращает число от 1 до 31, соответствующее текущему дню месяца
Exp (функция)	возвращает экспоненту числа
Fix (функция)	возвращает целую часть числа
Format (функция)	форматирует выражение в соответствии с заданным форматом
Iff (функция)	возвращает одно из двух значении в зависимости от значения выражения
InStr (функция)	возвращает позицию первой найденной подстроки в

Название	Описание
	строке
Int (функция)	возвращает целую часть числа
Is (операция)	сравнение двух ссылок на объекты
IsArray (функция)	возвращает булево значение, указывающее, является ли данная переменная массивом
IsDate (функция)	возвращает булево значение, указывающее, может ли выражение быть преобразовано к типу Date
IsEmpty (функция)	возвращает булево значение, указывающее, инициализировано ли значение данной переменной
IsError (функция)	возвращает булево значение, указывающее, является ли выражение значением кода ошибки
IsMissing (функция)	возвращает булево значение, указывающее, был ли передан данный необязательный параметр в подпрограмму
IsNull (функция)	возвращает булево значение, указывающее, не содержит ли выражение недопустимое (Null) значение
IsNumeric (функция)	возвращает булево значение, указывающее, может ли данное выражение рассматриваться как число
LBound (функция)	возвращает значение нижней границы индекса массива
Left (функция)	возвращает определенное число символов с начала строки
Len (функция)	возвращает число символов строки или число байт, необходимых для хранения переменной
Like (операция)	сравнение двух строк
Log (функция)	возвращает натуральный логарифм числа
LTrim (функция)	возвращает копию строки без лидирующих пробелов
Mid (оператор)	замещает определенное число символов в строке на символы из другой строки
Mid (функция)	возвращает определенное число символов с определенной позиции строки
Minute (функция)	возвращает целое число в диапазоне 0 59, представляющее минуту часа
Month (функция)	возвращает целое число в диапазоне 1 12, представляющее номер месяца
Now (функция)	возвращает текущие значения даты и времени
Randomize (оператор)	инициализирует генератор случайных чисел
Right (функция)	возвращает определенное число символов с правой стороны строки
Rnd (функция)	возвращает случайное число
RSet (оператор)	копирует правую часть строки в строковую переменную
RTrim (функция)	возвращает копию строки без конечных пробелов
Sgn (функция)	возвращает знак числа

Название	Описание
Sin (функция)	возвращает значение синуса угла
Space (функция)	возвращает строку, содержащую определенное число пробелов
Sqr (функция)	подсчет значения квадратного корня числа
Stop (оператор)	приостанавливает выполнение программы
Str (функция)	возвращает строковое представление числа
StrComp (функция)	возвращает результат сравнения строк
StrConv (функция)	возвращает преобразованную строку
String (функция)	возвращает строку заданной длины из одинаковых символов
Tan (функция)	возвращает значение тангенса угла
Time (оператор)	устанавливает значение системных часов
Time (функция)	возвращает значение типа Date, указывающее текущее системное время
Timer (функция)	возвращает число секунд, прошедших после полуночи
TimeSerial (функция)	возвращает значение типа Date, содержащее время для заданного часа, минуты и секунды
Time Value (функция)	возвращает значение типа Date, содержащее время суток
Trim (функция)	возвращает копию строки без начальных и конечных пробелов
Val (функция)	возвращает числовое представление строки
VarType (функция)	возвращает значение, указывающее тип переменной
Weekday (функция)	возвращает целое число, представляющее день недели
Year (функция)	возвращает целое число, представляющее год

12. Оператор присваивания

Данный тип оператора служит для присвоения начальных значений, записи результата вычисления в переменную, изменения значений.

Оператор присваивания имеет следующую структуру построения:

<Var>= <Formula>,

Var — имя переменной;

Formula — формула.

Переменная слева от знака "=" может быть простой переменной, элементом массива или свойством объекта. Формула состоит из переменных, констант, операций и функций.

Например:

```
X=200*0.8/70
Y="Петров Семен Иванович"
Z=X/80+30
```

13. Оператор With – End With

Оператор **With – End With** позволяет программисту выполнить несколько операций над одним и тем же объектом без повторений этого объекта при работе с его свойствами и методами.

Следующая процедура изменяет три свойства диапазона "A1:B2".

```
Sub ChageFont1()
   Range("A1:B2").Font.Name = "Times New Roman"
   Range("A1:B2").Font.FontStyle = "Bold Italic"
   Range("A1:B2").Font.Size = 12
End Sub
```

Эту процедуру можно переписать с помощью конструкции **With – End With**. Процедура, показанная ниже, работает точно так же, как и предыдущая.

```
Sub ChageFont2()
   With Range("A1:B2")
      .Font.Name = "Times New Roman"
      .Font.FontStyle = "Bold Italic"
      .Font.Size = 12
   End With
End Sub
```

Первый вариант более понятный, но вторая процедура помогает повысить эффективность выполнения кода по сравнению с эквивалентной ей процедурой, которая явно ссылается на объект в каждом операторе.

14. Комментарий. Перенос строки кода. Расположение нескольких операторв в одной строке

Комментарий - это пояснение к программе. Любой комментарий начинается в строке с произвольного места программы символом '. Комментарии предназначены для сопровождения и поддержки программ. Как правило, разработка и сопровождение программ осуществляются различными специалистами. Текст комментариев должен обеспечить понимание логики программы, отражать ее специфику.

Расположение символов <Пробел>+<Знак подчеркивания> в конце строки обеспечивает то, что последующая строка является продолжением предыдущей. При этом надо помнить, что:

- ✓ нельзя разбивать переносом строковые константы;
- ✓ допустимо не более семи продолжений одной и той же строки;
- ✓ сама строка не может состоять более чем из 1024 символов.

В следующем примере первая из конструкций является разбиением второй на две строки:

```
Dim s As Single, _
     r As Integer, t As Integer
```

и

```
Dim s As Single, r As Integer, t As Integer
```

Для переноса строковой константы ее надо представить как результат конкатенции нескольких строковых констант, а перенос строки производить по операции конкатенции.

Рассмотрим корректный и некорректный перенос строки "Visual Basic for Applications ".

Некорректный перенос	Корректный перенос
`S="Visual Basic for _` `Applications "`	`S=" Visual Basic for " _` `& "Applications "`

Обратите внимание, что для правильного переноса строковой константы применяется операция — конкатенция. Эта операция применяется для объединения нескольких строк в одну. Кроме символа амперсанда “&” для обозначения операции конкатенция применяется символ сложения “+”. При объединении двух строк вторая строка добавляется непосредственно в конец первой. Результатом является строка большего размера, содержащая целиком обе исходные строки.

В приведенном примере эта операция используется для соединения строк при переносе строки кода.

Использование знака двоеточия позволяет разместить несколько операторов на одной строке. Т.о., следующие две конструкции эквивалентны.

`x=1` `x=x+2`	`x=1:x=x+2`

15. Операторы ввода-вывода

Одним из важнейших свойств программы является **массовость**. Она обеспечивается возможностью ввода различных исходных данных в программу и получения различных результатов без изменения кода самой программы. Кроме того, пользователю необходимо наблюдать результаты работы программы, что обеспечивается операторами вывода данных.

В Visual Basic возможны несколько способов ввода данных в программу. Одним из самых простых является использование функции InputBox, имеющей следующий синтаксис:

InputBox(<строка_сообщение> [,<заголовок_окна>]
[,<текст_по_умолчанию>] [,Хпоз][,Упоз][, <файл-справка>,
<контекст>]),

текст_по_умолчанию — строка символов, выводимая в текстовом блоке;
Хпоз, Упоз — позиция левого верхнего угла окна (в твипах ≈ 1/1440 дюйма);
файл-справка — строковая переменная, задающая путь справочного файла, в котором находится дополнительная информация, относящаяся к теме окна сообщения;
контекст — константа, определяющая ссылку на раздел в справочном файле.

Функция **InputBox()** обеспечивает формирование окна для вывода строки сообщения и ожидания ввода строки символов или нажатия кнопки. Возвращает содержание текстового блока.

Пример 15.1. Пример формирования окна ввода.

```
Sub Msg_Inp()
   Message = "Введите Фамилию, Имя, Отчество студента"
   Title = " Пример окна для ввода"
   Def = " Иванов Иван Иванович"
   Response = InputBox(Message, Title, Def, 100, 100)
End Sub
```

В результате действия приведенной выше процедуры появится диалоговое окно, представленное на рисунке 4.

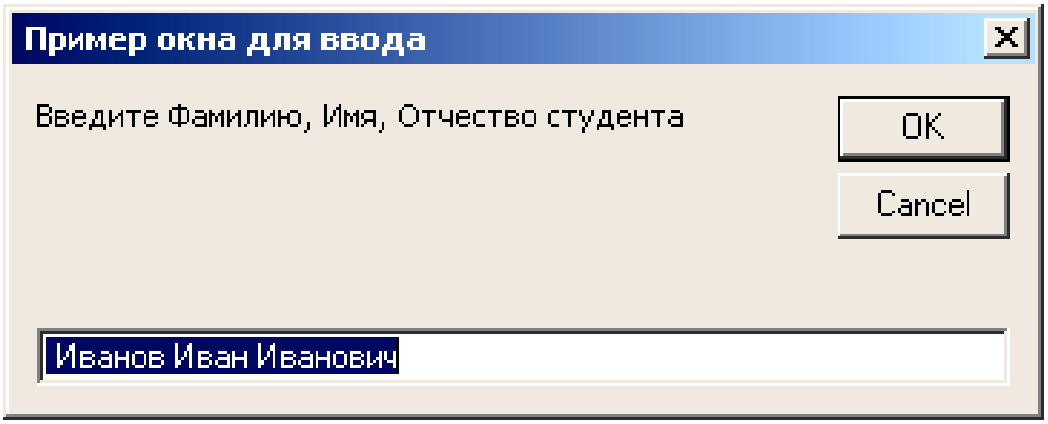


Рисунок 4. Диалоговое окно ввода

После ввода значения в текстовое поле и нажатия кнопки OK, переменной Response будет присвоен введенный текст.

Следует иметь в виду, что функция InputBox() возвращает текстовое значение. Поэтому при необходимости ввести число, следует преобразовать возвращаемое значение в требуемый числовой тип. Для преобразования в основные числовые типы используются функции:

- **CInt()** – для преобразования в целый тип Integer;
- **CSng()** – для преобразования в вещественный тип Single.

Таким образом, для ввода целого значения N следует записать:

```
N = CInt(InputBox("Введите N"))
```

При преобразовании текстовых значений в вещественные с помощью функции **CSng()**, в окне InputBox в качестве разделителя целой и десятичной частей числа должна быть введена запятая. Если введен текст, не соответствующий образу числа, то функции **CInt()** и **CSng()** выдают сообщение об ошибке.

Для вывода результатов вычислений можно использовать процедуру **MsgBox()**.

Встроенная функция **MsgBox()** обеспечивает создание диалоговых окон различных типов.

1. Простое окно-сообщение

MsgBox ("строка_сообщения")

Если в сообщении должно присутствовать значение переменной или элемента массива переменных, элемент структуры пользовательского типа данных и т.п., следует преобразовать значения в *строковый* и обеспечить конкатенацию строк. Для преобразования числа любого типа в текст используется функция **CStr().**

Пример 15.2. Пример вывода сообщения о значении переменной.

```
Sub Msg_Priim()
   Randomize
   a=rnd()
   MsgBox "Значение случайного числа  " & CStr(a)
End Sub
```

В результате действия приведенной выше процедуры может появиться диалоговое окно показанное на рисунке 5.

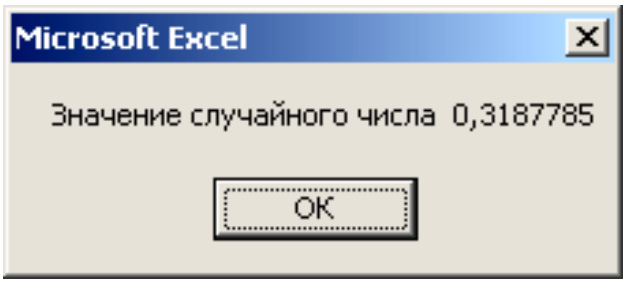


Рисунок 5. Окно вывода сообщения о значении переменной

Если в одном окне MsgBox требуется вывести несколько чисел, каждое из них следует преобразовать в текстовый тип и "склеить" оператором конкатенции (+ или &).

2. Окно-сообщение с командными кнопками

Общий формат оператора:

MsgBox("строка_сообщение"[, <кнопки>] [, "заголовок_окна" [, <файл-справка>, <контекст>]),

строка_сообщение — строка символов с максимальной длиной 1024 символа;
кнопки — число, являющееся суммой кодов выбранных типов кнопок и пиктограммы, или имена кнопок;
заголовок_окна — строка символов;
файл-справка — строковая переменная, задающая путь справочного файла, в котором находится дополнительная информация, относящаяся к теме сообщения;
контекст — константа контекстной ссылки, которая дает ссылку на тему в справочном файле.

Пример 15.3. Пример формирования диалогового окна с кнопками

```
Sub Priim()
   Msg = "Вы хотите продолжить?"
   Style = 66
   Title = " Пример окна-сообщения"
   Response = MsgBox(Msg, Style, Title)
End Sub
```

В результате действия приведенной выше процедуры появится диалоговое окно показанное на рисунке 6.

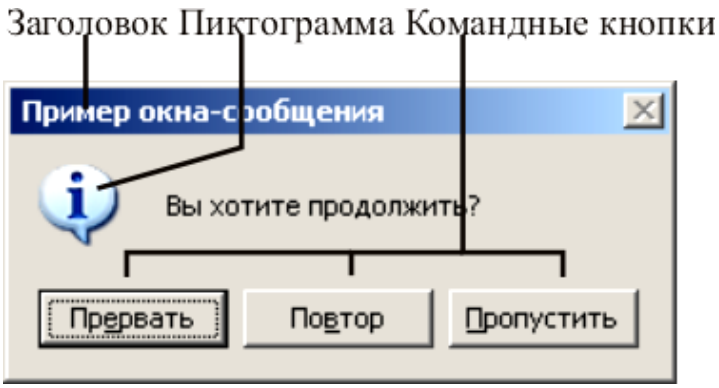


Рисунок 6. Диалоговое окно с кнопками

Коды задания командных кнопок и пиктограмм в функции **MsgBox()** приведены в табл. 6 и табл.7. Функция **MsgBox()** возвращает код (число), соответствующий нажатой кнопке.

Таблица 6. *Список видов командных кнопок и их кодов*

Код	Константа	Отображаемые кнопки
0	vbOKOnly	ОК
1	vbOKCancel	ОК, Отмена
2	vbAbortRetrylgnore	Прервать, Повторить, Игнорировать
3	vbYesNoCansel	Да, Нет, Отмена
4	vbYesNo	Да, Нет
5	vbRetryCansel	Повторить, Отмена

Таблица 7*. Пиктограммы и соответствующие им коды*

Код	Константа	Описание
16	vbCritical	Важное сообщение
32	vbQuestion	Предупредительный запрос
48	vbExclamation	Предупредительное сообщение
64	vbInformation	Информационное сообщение

В приведенной выше процедуре Priim переменная Style получила значение 66, как результат сложения двух констант кода командных кнопок (2) и кода пиктограммы (64).

После щелчка на кнопке диалогового окна возвращается значение соответствующее этой кнопки. При написании программ с откликом, в зависимости от того, какая кнопка диалогового окна нажата, вместо возвращаемых значений удобнее использовать приведенные в табл.8 константы VBA, которые делают код программы более удобочитаемым и, к тому же, их легко запомнить.

Таблица 8. *Константы, возвращаемые оператором MsgBox*

Константа	Значение	Нажатая кнопка
vbOK	1	OK
vbCancel	2	Отмена (Cancel)

Константа	Значение	Нажатая кнопка
vbAbort	3	Прервать (Abort)
vbRetry	4	Повторить (Retry)
vbIgnore	5	Пропустить (Ignore)
vbYes	6	Да (Yes)
vbNo	7	Нет (No)

16. Оператор ветвления

Оператор ветвления (условный оператор) предназначен для изменения порядка выполнения программы по некоторому условию. Условный оператор соответствует блоку ветвления в блок-схеме.

Синтаксис оператора:

If <условие1> Then
<Блок 1>
[ElseIf <условие2> Then
<Блок 2>
· · ·]
[Else
<Блок N>]
End If

Оператор работает следующим образом: если (**If**) выполняется **<условие 1>**, то (**Then**) будет выполнена последовательность операторов **<Блок 1>**, иначе, если (**ElseIf**) выполняется **<условие 2>**, то будет выполнен **<Блок 2>** и т.д., иначе (**Else**) - **<Блок N>** .

Допустима форма записи оператора ветвления в одну строку:

If <условие1> Then <Оператор 1>: <Оператор 2>: . . . : <Оператор N>

В этом случае не указывается конец оператора **End If** как в первом варианте. В случае, если условие ложно, управление передается оператору, следующему за условным.

Пример 16.1. Использования условного оператора

```
Sub Pr1_If()
   Dim ball As String, st As String
   st = "Вы хотите поступить в РГСУ?"
   ball=MsgBox(st, vbYesNo)
   If ball = vbYes Then
      MsgBox "Надо сдать экзамены."
   Else
     MsgBox "Идите в ПТУ."
   End If
```

```
End Sub
```

В результате работы процедуры пользователь в зависимости от ответа на вопрос «Вы хотите поступить в РГСУ?» получает различные сообщения.

17. Оператор выбора Select Case

Данный оператор применяется в том случае, если во всех логических условиях участвует одна и та же величина (переменная):

Select Case <величина>
Case <сравнение 1>
<блок операторов 1>
Case <сравнение2>
<блок операторов2>
[Case Else
<блок операторов Else>]
End Select

Выражение для *сравнения* может быть записано в виде: Case 45 или Case 3, 4, 5, или Case 5 To 12.

Если ни одно из сравнений не является истинным, выполняется блок операторов **Else,** если блок **Else** отсутствует, управление передается оператору, следующему за **End Select**.

Пример 17.1. Использования оператора выбора

```
Sub Pr_Case()
  Vozrast=InputBox("Сколько Вам лет?")
  Select Case Vozrast
   Case Is  <=7
        Msgbox "Вы дошкольник."
   Case 8 to 16
       Msgbox "Вы учитесь в школе."
    Case 17 to 30
       Msgbox "Вам пора заняться делом."
    Case 31 to 60
       Msgbox "Кто не работает, тот не ест."
    Case Else
       Msgbox "Вы заслужили отдых."
  End Select
End Sub
```

Если значение переменной Vozrast меньше или равно 7, отображается сообщение ”Ты дошкольник”. Если значение переменной Vozrast находится в диапазоне от 8 до 16, отображается сообщение ”Ты учишься в школе”. Если значение

переменной Vozrast находится в диапазоне от 17 до 30, отображается сообщение ”Тебе пора заняться делом”. Если значение переменной Vozrast находится в диапазоне от 31 до 60, отображается сообщение ” Кто не работает, тот не ест”. Если значение возраста не равно ни одному из предложенных диапазонов значений, выводится сообщение ”Вы заслужили отдых”.

Код этой процедуры более прост для восприятия, чем код программы записанный с помощью условного оператора. В случае трёх и более возможных разветвлений в программе лучше использовать оператор **Select Case.**

Упражнения 1.

1. X и Y могут принимать лишь два значения A и B исключающим образом: если X=A, то Y=B; если X=B, то Y=A. Записать приведенное высказывание с помощью условного оператора и без него.

2. Задается дата в виде числа, месяца и года. Требуется определить завтрашнее число.

3. Считывается пять первых цифр номера карточки социального страхования в виде двух целых переменных ПХ — код пола (1 – М, 2 – Ж), ГР — год рождения. Надо вывести возраст рассматриваемого лица в виде такого сообщения: «Госпожа, Вам … лет» или «Господин, Вам … лет».

4. Дан номер некоторого года (положительное целое число). Вывести число дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

5. Дан номер некоторого года (положительное целое число). Вывести соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год.

18. Оператор цикла For . . . Next

Данный оператор повторяет выполнение группы инструкций указанное число раз и к категории *арифметических* циклов. Оператор For . . . Next соответствует циклическому алгоритму с параметром (см. лаб. раб. №7).

Синтаксис:

For <i>=<i0> To <iN> [Step <ih>]
  [инструкции]
  [Exit For]
  [инструкции]

Next [<i>]
Значения <i>, <i0>, <iN>, <ih> соответственно означают:

- <i> - параметр цикла (счетчик);
- <i0>, <iN> - начальное и конечное значения параметра цикла;
- <ih> - значение шага изменения параметра цикла.
  Перечисленные величины являются целыми числовыми.

Оператор For. . . Next повторяет выполнение группы инструкций, пока параметр цикла i изменяется от начального значения i0 до конечного значения iN с указанным шагом ih. Если шаг не указан, то он по умолчанию принимается равным 1. Для досрочного выхода из цикла используется команда **Exit For**.

Для каждого значения параметра цикла выполняется блок операторов, расположенных до ключевого слова **Next**. Затем происходит изменение переменной цикла (увеличение значения на шаг), проверяется полученное значение (не более указанного конечного значения) и повторяется выполнение блока операторов.

Если значение параметра цикла превысило значение iN, управление передается оператору, следующему за **Next.**

Примеры использования оператора For . . . Next

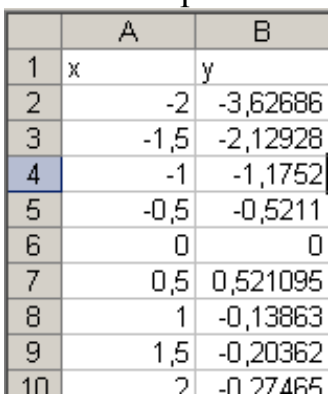
Пример 18.1. Необходимо построить таблицу значений функции

$$y=\begin{cases}\dfrac{\ln(1+x)}{x-6}, \text{если } x\ge 1\\ \dfrac{e^{x}+e^{-x}}{2}, \text{если } x<1\end{cases}, \; x\in[-2;5], \text{ шаг } 0{,}5.$$

Будем выводить значения функции на отрезке в ячейки рабочего листа. Для этого создадим счетчик i для вычисления номера строки.

```
Sub PR_For2()
   Cells(1,1)="x" : Cells(1,2)= "y"
   i = 1
   For x = -2 To 5 Step 0.5
     If x >= 1 Then
       y = Log(1 + x) / (x - 6)
     Else
       y = (Exp(x) - Exp(-x)) / 2
     End If
     i = i + 1
  'Вывод результата в ячейки рабочего листа
     Cells(i, 1) = x: Cells(i, 2)=y
   Next
End Sub
```

В результате работы программы на рабочем листе Excel будет получена таблица значений функции показанная на рис.7.



	A	B
1	x	y
2	-2	-3,62686
3	-1,5	-2,12928
4	-1	-1,1752
5	-0,5	-0,5211
6	0	0
7	0,5	0,521095
8	1	-0,13863
9	1,5	-0,20362
10	2	-0,27465

Рисунок 7.

Пример 18.2. Найти количество положительных чисел среди элементов массива A(20), имеющих четные индексы.

```
Sub Pr_For3()
   Dim A(20) as Single
  'Заполнение массива
   For k=1 To 20
     A(k)=-1+2*Rnd()
   Next
   p=0
  'Поиск положительного элемента в массиве
   For i=2 to 20 Step 2
  'Если положительный элемент найден
  'счетчик p увеличить на1
      If A(i)>0 then p=p+1
   Next
  'Отображение результата
   MsgBox "p=" & Cstr(p)
End Sub
```

Пример 18.3. Дана строка длиной до 254 символов, содержащая слова, разделенные пробелами. Преобразовать строку в массив слов (лексем).

Будем считать, что разделителем слов в строке является единственный пробел. Просмотр строки осуществим в цикле с известным числом повторений: количество символов определяем при помощи функции Len. При помощи функции Mid будем в цикле по одному выделять символы строки и анализировать их. Перед началом формирования очередного слова увеличиваем счетчик на 1 и присваиваем элементу массива лексем (каждая лексема состоит из одного символа) стартовое значение "", чтобы было к чему прибавлять очередные символы.

```
Sub Lexema()
   Dim w() As String
```

```
    st = TextBox1.Text
  'Длину строки делим нацело на 2
  n = Len(st) \ 2
  'w — массив лексем, размерность массива n
  'Лексема состоит из одного символа
  ReDim w(1 To n) As String
  'Добавляем к началу строки пробел
  'к — счетчик массива лексем
  st = st + " ": k = 1: w(1) = ""
    For i = 1 To Len(st)
     c = Mid(st, i, 1)
     If c = " " Then
      If Mid(st, i + 1,1) <> " " Then k=k+1:w(k)=""
     Else
        w(k) = w(k) + c
     End If
    Next
  'Вывод результатов
    For i = 1 To k - 1
       MsgBox "Лексема №" & i & ": " & w(i)
    Next
End Sub
```

19. Цикл Do

Для выполнения оператора For необходимо задать параметры, которые будут определять, сколько раз должен выполниться оператор(ы) цикла. Альтернативой циклу с For…Next является цикл Do , в котором 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. Например, Вы хотели бы, чтобы пользователь вводил пароль в вашей программе до тех пор, пока он не совпадёт с Вашим паролем. Существует несколько разновидностей цикла Do, в зависимости от условий его выполнения.

Цикл Do с предусловием.

Синтаксис:

Do [While|Until] <условие>
  [инструкции]
  [Exit Do]
  [инструкции]
Loop

Если выбрано служебное слово **While**, то цикл продолжается, пока выполняется условие; если **Until** – то прекращается, когда условие выполняется.

Цикл Do с постусловием
Синтаксис:
Do
  [инструкции]
  [Exit Do]
  [инструкции]
Loop [{While|Until}] <условие>

Значение служебных слов цикла аналогично циклу с предусловием, но цикл выполняется хотя бы один раз.

В обоих циклах Do для досрочного выхода из цикла используется команда **Exit Do**.

Рассмотрим примеры.

Пример 19.1. Вычислить сумму членов ряда $z = 1 + \sum_{n=1}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$ с точностью до члена ряда, меньшего ε.

```
Sub Pr_Do()
   Dim x as Single
   x = InputBox("Введите x .")
   eps = 0.01
   y = 1: z = 1: i = 1
   Do While Abs(y) > eps
     y = y * (-1) * x ^ 2 / (2 * i * (2 * i - 1))
     z = z + y
     i = i + 1
   Loop
   MsgBox "z=" & CStr(z)
End Sub
```

Цикл Do будет выполняться до тех пор, пока член ряда по модулю будет больше eps. При выполнении каждого последующего шага цикла следующий член ряда получаем из предыдущего (y(n)=y(n-1)*c). Чтобы найти множитель c необходимо y(n) разделить на y(n-1).

Пример 19.2. Некоторая сумма денег помещается в банк под r процентов годовых. Найти через сколько лет сумма удвоится.

Пусть s0 начальная сумма денег, помещенная в банк. Поместим значение s0 в ячейку s. Так как процент, наращенный суммой s равен s*r/100, то сумма через год станет равна s=s+ s*r/100 или s= s * (1 + r / 100). Таким образом нужно продолжать счет, пока s<2*s.

Запишем решение поставленной задачи с помощью операторов цикла Do с предусловием и постусловием.

Решение с использованием цикла с предусловием.

```
Sub Vclad1()
   Dim s As Single, r As Integer, t As Integer
   Dim st As Single
   s0 = InputBox("Введите начальную сумму.")
   st="Введите проц. ставку. "
   st=st & "Целое число от 1 до 100"
   r=InputBox(st)
   t = 0: s = s0: ss = 2 * s0
   Do While s < ss
      s = s * (1 + r / 100)
      t = t + 1
   Loop
 MsgBox "Сумма " & s0 & " у.е., помещенная под " _
& r & " процентов годовых, удвоится через " _
& t & " лет."
End Sub
```

Поскольку наращенную сумму нужно вычислить хотя бы за один год, можно применить для решения поставленной задачи цикл с постусловием.

```
Sub Vclad2()
   Dim s As Single, r As Integer, t As Integer
   Dim st As Single
   s0 = InputBox("Введите начальную сумму.")
   st="Введите проц. ставку. "
   st=st & "Целое число от 1 до 100"
   r=InputBox(st)
   t = 0: s = s0: ss = 2 * s0
   Do
     s = s * (1 + r / 100)
     t = t + 1
   Loop Until s >= ss
   MsgBox "Сумма " & s0 & " у.е., помещенная под " _
& r & " процентов годовых, удвоится через " _
& t & " лет."
End Sub
```

В первой процедуре цикл повторяется пока условие истинно, во второй — пока условие ложно.

20. Оператор For Each

В случае, когда необходимо выполнить операции над всеми объектами некоторой коллекции или массива очень часто удобно воспользоваться оператором For Each, особенно когда неизвестно, сколько элементов насчитывает коллекция или массив.
Синтаксис оператора For Each:

```
For Each элемент In группа
    [инструкции]
    Exit For
    [инструкции]
Next [элемент]
```

Пример 20.1. Суммирование элементов с помощью оператора For Each.
Создадим процедуру, которая суммирует элементы массива или коллекции.

```
Public Sub Sum(v As Variant, s as Single)
    s = 0
    For Each c In v
       s = s + c
    Next
End Sub
```

В процедуре DemoForEach вызовем процедуру Sum и для суммирования элементов массива и для суммирования значений ячеек диапазона "A1:A7"

```
Sub DemoForEach ()
    Dim a() As Single
    Dim sa as Single
    Dim sr as Single
  'Получаем размерность массива как случайное целое число
  'на промежутке от 5 до 10
    n = Int(5 + 5 * Rnd())
    ReDim a(n) As Single
    For i = 1 To n
       a(i) = Rnd()
    Next
'Вызываем процедуру Sum для массива a()
    Sum a, sa
    MsgBox "S(массива)=" & sa
'Вызываем процедуру Sum ячеек диапазона
    Sum Range("A1:A7"), sr
    MsgBox "S(диапазона)=" & sa
End Sub
```

Оператор **For Each** позволил применить одну и ту же процедуру для различных объектов: массива и диапазона.

Упражнения[1] 2

1. Найти массив абсолютных приростов ΔY ряда динамики Y(N) (Y — одномерный числовой массив).В одномерном массиве определить первый отрицательный элемент и его номер.

2. Протабулировать функцию y=sin(x) на отрезке [1,5] с шагом h=0,5. Вывести предпоследнее положительное значение функции.

3. При табулировании функции y=cos(x+a) на отрезке [1,10] с шагом h=1 определить сумму значений y , больших p.

4. В простую переменную последовательно вводят N чисел. Определить, сколько чисел больше своих соседей справа и слева.

5. Проверить, упорядочены ли по убыванию элементы одномерного массива.

6. Найти сумму элементов массива $X(20)$ по следующему правилу : $S = \sum_{n=1}^{k} x_n$, если $x_n < x_{n+1}$.

7. Найти скалярное произведение векторов $A(n)$ и $B(n)$.

8. Дана строка длиной до 254 символов. Удалить все пробелы перед знаками препинания.

9. На причале ждут посадки C пассажиров. Для каждого из них известен вес вместе с багажом. Капитан хочет посадить на борт как можно больше пассажиров. Определить максимально возможное количество пассажиров, если известна грузоподъемность судна.

10. Население двух стран равно Н1 и Н2, средний прирост населения П1 и П2 соответственно. Дано, что Н1>Н2 и П1<П2, вычислить, через сколько лет население Н2 превзойдет население Н2.

11. Ввести двумерный массив A(N,N) . Составить алгоритм подсчета среднего арифметического значения двумерного массива. Вычислить отклонение от среднего для всех элементов двумерного массива.

12. Составить алгоритм нахождения числа строк двумерного массива A(N,N) , количество отрицательных элементов в которых больше P.

13. Ввести одномерный массив из N случайных целых чисел в диапазоне от -5 до 25. Вычислить среднее арифметическое от остатков от деления значений элементов на их индексы.

---

[1] Для решения предложенных задач рекомендуется разобрать материал предложенный в лаб.работах №7-№10

14. Получить из двумерного массива одномерный.

15. Получить из одномерного массива двумерный.

16. В одномерном массиве определить первый отрицательный элемент и его номер.

17. Исключить из массива A1..AN первый отрицательный элемент.

18. Перенести в хвост одномерного массива все отрицательные элементы.

19. Перенести в начало одномерного массива все нечетные элементы.

20. В одномерном массиве найти первую группу повторяющихся элементов.

21. Исключить из массива A(N) первый четный элемент, следующий за максимальным.

22. В одномерном массиве перенести в конец минимальный элемент.

23. Найти минимальное значение функции Y=Sin(X)*X , на отрезке [C,D] с шагом 0.001. Реализовать цикл с постусловием.

24. Найти $S = \sum_{n=1}^{20} \frac{(-1)^n x^{2n-1}}{(2n-1)!}$

25. Для целочисленного массива $(a_1, a_2, ..., a_{75})$ определить, является ли сумма его элементов четным числом, и вывести на печать ДА или НЕТ.

26. Сравнить сумму четных и сумму нечетных чисел натурального ряда от 1 до n.

27. Найти сумму 25 чисел, полученных по рекуррентной формуле: $a_{n+1} = a_n + a_{n-1}; a_0 = 1, a_1 = 1.$

28. Задан одномерный массив. Все его элементы, не равные нулю, переписать (сохраняя их порядок) в начало массива, а нулевые - в конец массива.

29. Найти значение функции $f(x) = \begin{cases} \sum_{i=1}^{20} \frac{x^{2i+1}}{i}, & x \geq 10 \\ \prod_{i=1}^{20} \frac{x^{2i+1}}{i}, & 0 < x < 10 \end{cases}$.

30. Найти сумму двадцати первых членов ряда $z = \sum_{n=1}^{20} (-1)^n \frac{(\frac{\pi}{3})^{2n+1}}{n!}$.

31. В массиве X(N) каждый элемент равен либо 0, либо 1, или 2. Переставить элементы массива так, чтобы сначала располагались все нули, затем все единицы и, наконец, все двойки (дополнительного массива не заводить).

32. Задан целочисленный массив B(30). Подсчитать количество перемен знака элементов в массиве (нулевые элементы пропускать).

33. Составить программу поиска первого элемента последовательности $a_i = \left(1 - \frac{1}{i+1}\right)$, меньшего суммы предыдущего и заданного eps. Печатать номер найденного элемента.

34. Дано целое число N (> 1). Вывести наименьшее целое K, при котором выполняется неравенство 2K >N, и само значение 2K.

35. Дано целое число N (> 2) и две вещественные точки на числовой оси: A, B (A<B). Отрезок [A, B] разбит на равные отрезки длины H с концами в N точках вида A, A + H, A + 2H, A + 3H, ..., B. Вывести значение H и набор из N точек, образующий разбиение отрезка [A, B].

36. Ввести двумерный массив A(N,M). Составить алгоритм замены всех нулевых элементов на минимальный элемент.

37. Ввести двумерный массив A(N,N). Составить алгоритм замены всех отрицательных элементов на среднее арифметическое значение элементов двумерного массива.

38. Составить алгоритм нахождения числа строк двумерного массива A(N,N) , количество отрицательных элементов в которых больше P.

39. Ввести двумерный массив размером 7*4 . Найти наибольший элемент двумерного массива. Удалить строку с максимальным элементом.

40. Ввести двумерный массив размером 7*7. Найти максимальный элемент двумерного массива.

21. Оператор безусловного перехода GoTo

Оператор безусловного перехода GoTo задает переход программы на новую инструкцию, которая помечена специальным образом.
Синтаксис оператора безусловного перехода:

GoTo <указатель>,

указатель — метка.

Для использования оператора безусловного перехода надо какой-то строке присвоить метку. Метка — это текстовая строка, заканчивающаяся двоеточием, или число без двоеточия, указанные перед инструкцией. Процедуры VBA могут содержать любое количество меток, а оператор GoTo не определяет переход за пределы процедуры.

Оператор **GoTo** как правило используется, если не существует другого способа выполнить действие. Единственной ситуацией, когда оператор **GoTo** действительно необходим является перехват ошибок (См. Лаб. раб. №12 и раздел 22).

22. Перехват и обработка ошибок

При выполнении программы могут происходить ошибки: синтаксические, которые нужно исправить перед тем, как выполнять процедуру, и ошибки выполнения, которые происходят непосредственно в процессе выполнения процедуры. Как правило, ошибка в процессе выполнения вызывает остановку программы, а пользователь видит диалоговое окно, в котором отображается описание ошибки. В хорошо написанном приложении пользователь не наблюдает этих служебных сообщений. В приложение включают специальные процедуры, перехватывающие ошибки и выполняющие соответствующие действия.

Перехват ошибки производит оператор On Error. Он устанавливает, что программа должна делать в случае появления ошибки.

Допустимы следующие варианты записи оператора On Error.

- On Error GoTo <указатель>
  В этом случае программа переходит к специальному разделу кода для обработки ошибки, на который указывает <указатель> (либо метка, либо номер строки). Этот раздел вводится в конце процедуры.
- On Error Resume Next
  При возникновении ошибки происходит передача управления на инструкцию, непосредственно следующую за инструкцией, где возникла ошибка.
- On Error GoTo 0
  Отключается любой активизированный обработчик ошибок в текущей процедуре.

При возникновении ошибки можно использовать объект Err для определения ее номера. Свойство Number объекта Err возвращает код ошибки.

В приведенном ниже примере демонстрируется, как можно перехватить ошибки с помощью оператора On Error.

Пример 22.1. В процедуре обработки события Click кнопки CommandButton1 предусмотрена возможность информирования пользователя о возникшей ошибке.

В обработчике ошибок предусмотрены следующие возможности отклика:

- пользователь информируется об ошибке деления на ноль (ошибка 11), после чего процедура обработки события Click кнопки завершает работу;
- пользователь информируется об ошибке несоответствия типов (ошибка 13) и сообщается о необходимости ввести число, после чего процедура обработки события Click кнопки завершает работу;

- если при выполнении программы возникнут какие-то непредвиденные ошибки, то о них информируется пользователь и прерывается выполнение процедуры обработки события Click кнопки.

```
Private Sub CommandButton1_Click()
   Dim x As Single
   Dim y As Single
   On Error GoTo ErrorHendler
   x = TextBox1.Value
   y = 1 / x
   MsgBox "y=" & y
   Exit Sub
   ErrorHendler:
   Select Case Err.Number
     Case 11
       MsgBox "Деление на ноль!"
       Exit Sub
     Case 13
       MsgBox "Несоответствие типов. Введите число!"
       Exit Sub
     Case Else
       MsgBox "Непредвиденная ошибка."
       Exit Sub
   End Select
End Sub
```

23. Введение в процесс разработки приложений

Если Вы собираетесь заниматься проектированием приложения, то, наверное, представляете не только, как оно будет выглядеть (внешний вид интерфейса), но и степень сложности работы с Вашей программой, учитывая то обстоятельство, что с ней может работать и новичок, и опытный пользователь.

Вы должны представлять, как и откуда будут поступать данные, а также, где и как будут храниться данные, полученные с помощью разработанного приложения, в каком виде Вы собираетесь выводить полученные результаты. Эти вопросы не поставят Вас в тупик, если владеть тем инструментарием, который предоставляет пользователю Excel.

Если Вы хотите, чтобы с Вашим приложением с удовольствием работали и новичок и профессионал, то Вам следует воспользоваться Экранными формами или как их ещё называют - пользовательскими формами (от английского - UserForm). Созданием на форме объектов управления и установкой значений

свойств этих объектов, пользователь создаёт себе условия для работы с будущим приложением как с обычным диалоговым окном.

VBA обладает встроенным набором элементов управления. Используя этот набор и редактор форм не трудно создать любой пользовательский интерфейс, который будет удовлетворять всем требованиям, предъявляемым к интерфейсу в среде Windows. Элементы управления являются объектами. Поэтому, как любые объекты, они обладают свойствами, методами и событиями. Элементы управления создаются при помощи панели инструментов

На этой панели представлены кнопки, позволяющие конструировать элементы управления, а также кнопки вызова окна свойств, перехода в режим конструктора и редактор кода, если панель Элементы управления находится в рабочем листе Excel.

Создание элементов управления на рабочем листе или в форме как правило происходит на начальном этапе конструирования приложения. Иногда используется программное их создание в процессе работы приложения. Но этот подход применяется реже.

Большинство элементов управления можно располагать как на рабочем листе, так и в форме. Но существуют такие элементы управления, которые можно располагать только в форме.

В табл. 9 приведен список основных элементов управления и соответствующих кнопок панели инструментов **Элементы управления** (ControlToolBox).

Таблица 9. Элементы управления

Элемент управления	Имя	Кнопка для создания элемента
Поле	TextBox	
Надпись	Label	
Кнопка	CommandButton	
Список	ListBox	
Поле со списком	ComboBox	
Полоса прокрутки	ScrollBar	
Счетчик	SpinButton	
Переключатель	OptionButton	
Флажок	CheckBox	

Выключатель	ToggleButton	
Рамка	Frame	
Рисунок	Image	

Лабораторная работа №1

Тема: Знакомство с редактором VBA. Создание первой программы.
Задача: познакомиться со структурой проекта VBA, создать программу, которая бы очищала ячейки рабочего листа.

Порядок выполнения

1. Запустите Excel.

2. Для открытия редактора VBA нажмите клавиши <Alt+F11>

3. На рис. 8 показано типичное окно редактора VBA. Не беспокойтесь, если ваше окно редактора VBA отличается от показанного на рисунке 8. Далее подробно рассмотрим визуализацию основных частей редактора VBA.

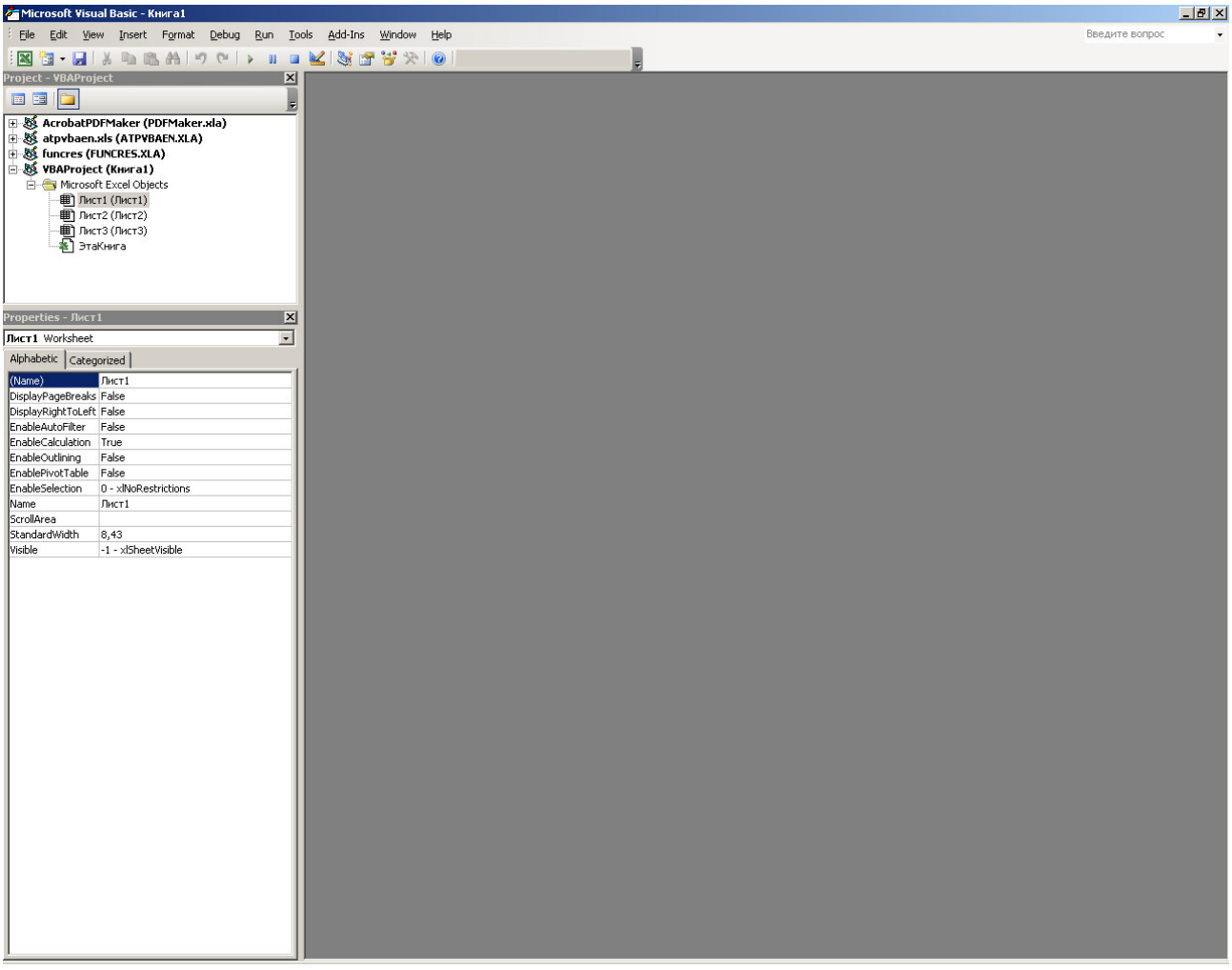


Рисунок 8. Окно редактора VBA

4. Закройте все раскрытые окно, щелкнув на кнопке Закрыть в правом верхнем углу каждого окна.

5. Отразим окно Диспетчер проектов, выполнив команду **View(Вид)/Project Explorer** или щелкните на кнопке Project Explorer, расположенной на панели инструментов Стандартная. Окна диспетчера проектов для показаны на рис. 9.

Некоторые отличия объектов Visual Basic в различных версиях Microsoft Office в рамках предложенного в данном учебном пособии материала являются несущественными.

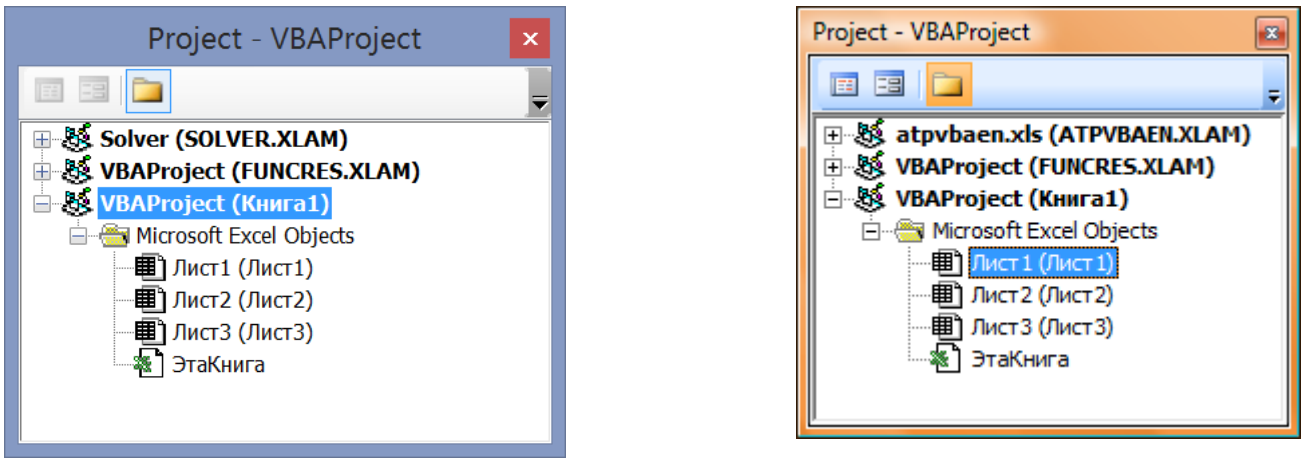


Рисунок 9. Окно Диспетчер проектов (слева для Microsoft Office 20163, справа – Microsoft Office 2007)

По умолчанию проект состоит из модулей рабочих листов и модуля ЭтаКнига. Код, имеющий непосредственное отношение к рабочему листу (например, код обработки событий листа), помещается в соответствующий этому листу модуль. Модуль ЭтаКнига содержит код обработки событий рабочей книги.

6. Выполните команды: **Insert(Вставить)/UserForm**, **Insert(Вставить)/Module**, **Insert(Вставить)/Class Module**. Структура проекта показана на рисунке 10.

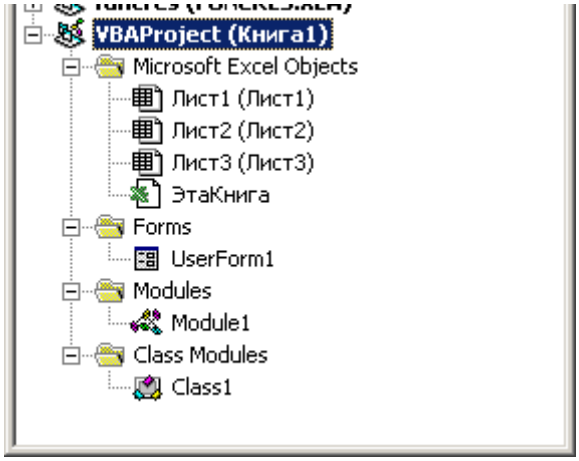


Рисунок 10. Диспетчер проектов содержит список всех модулей проекта

Excel позволяет создавать формы (объект UserForm) для взаимодействия с пользователем. Разработка приложений пользователя будет рассматриваться позже.

В стандартных модулях (Modules) хранятся процедуры, доступные из любых других объектов проекта Excel.

Модули классов (Class Modules) предназначены для создания пользовательских объектов. Помимо этого, модули классов позволяют программистам обмениваться фрагментами кода, не вдаваясь в подробности работы последнего. Изучение модулей класса не предусмотрено в данном курсе.

9. Для эффективной работы в редакторе VBA рекомендуется открыть окно свойств, которое предназначено для редактирования параметров различных компонентов — рабочих листов, книг, модулей или элементов управления форм. Список параметров компонента зависит от его типа.

9.1. Выберите объект Лист1 в Окне проекта (выполните двойной щелчок на объекте Лист1).

9.2. Чтобы открыть окно свойств объекта Лист1, выберите команду меню **View(Вид)/Properties Window (Окно свойств)** или щелкните на кнопке Project Properties (Свойства проекта), расположенной на панели инструментов Стандартная  .

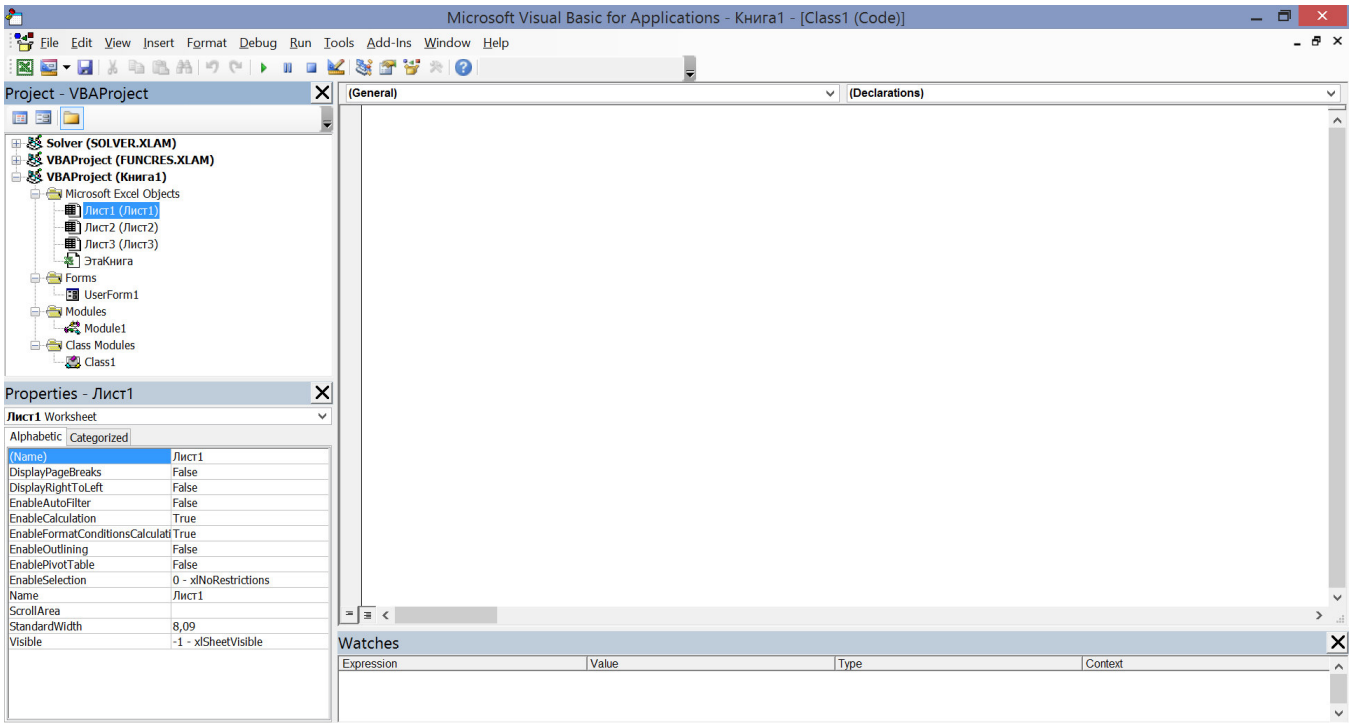


Рисунок 11. Окно проекта, окно свойств и модуль редактирования кода объекта Лист1

Т.о. в редакторе VBA должны быть открыты три окна: Окно проекта, окно свойств и модуль объекта Лист1 (окно редактирования кода) (см. рис. 11).

10. Напишем программу, с помощью которой будем очищать ячейки диапазона, например, A1:A3 на листе Лист1 (Должно быть открыто окно редактирования кода объекта Лист1). Для этого выполните следующие действия.

10.1. Выберите команду Insert/Procedure (Вставить/Процедуру). Заполните поля в появившемся диалоговом окне согласно рис. 12.

10.2. Щелкните на кнопке ОК. Редактор вставит первую и последнюю строки новой процедуры в окно редактирования кода (рис. 13).

10.3. Между первой и последней строкой новой процедуры наберите код **`Range("A1:A3").`**

После ввода точки на экране отобразиться список компонентов (рис. 14), которые логически должны завершить данную инструкцию.

Отображение списка компонентов, логически завершающих вводимую инструкцию, является одним из интеллектуальных качеств редактора кода.

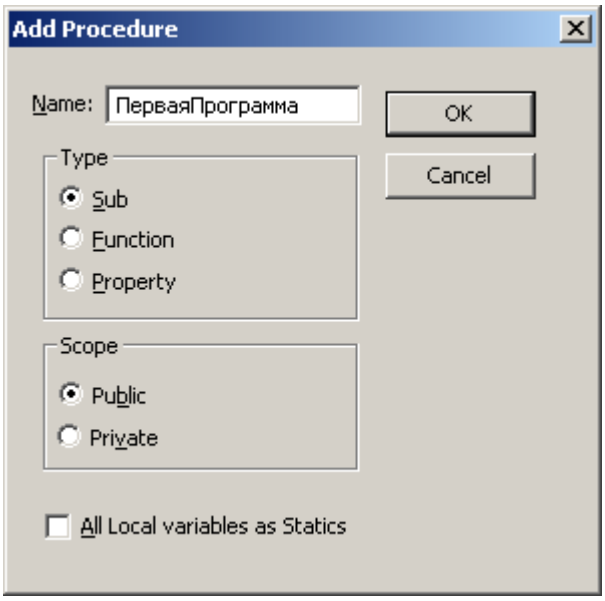


Рисунок 12. Диалоговое окно Вставить процедуру

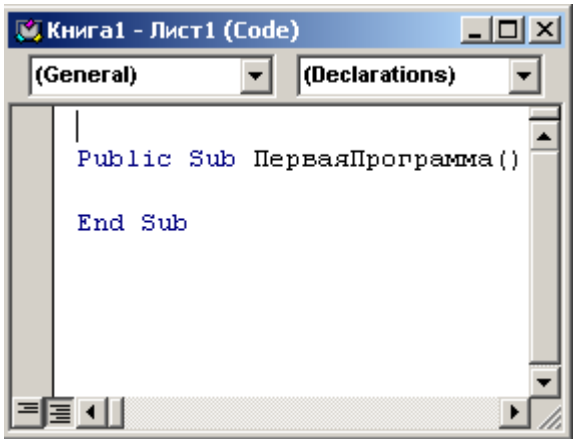


Рисунок 13. Редактор VBA добавляет первую и последнюю строки кодов программы

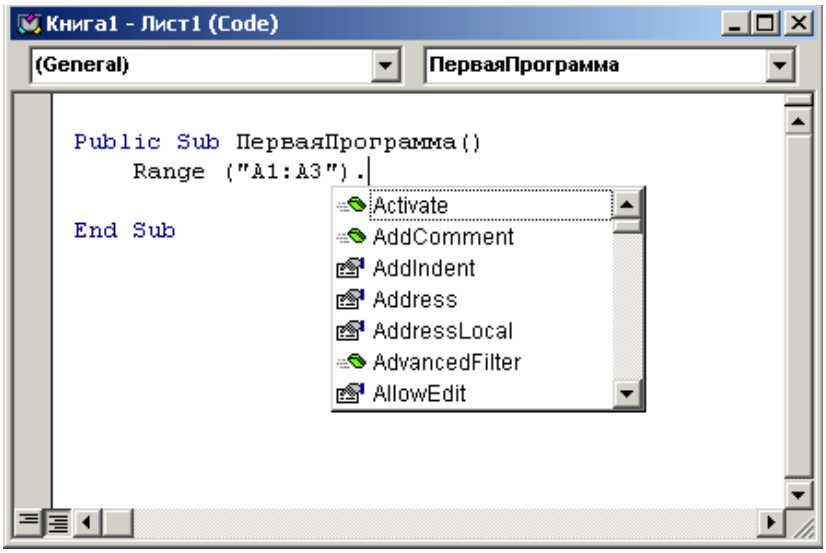


Рисунок 14. Список компонентов объекта Range

10.4. Для того чтобы очисть диапазон необходимо вызвать метод Clear. Выберите в списке метод Clear и выполните или двойной щелчок на выбранном элементе, или нажмите клавишу Tab.
Т.о. получаем следующую процедуру

```
Public Sub ПерваяПрограмма()
    Range("A1:A3").Clear
End Sub
```

10.5. Для того чтобы проверить синтаксис написания программы необходимо выполнить команду **Debug/Complete VBAProject**.

10.6. Вернитесь в окно Microsoft Excel и сохраните **рабочую книгу** со своей первой программой.

11. Посмотрим, как работает программа.

11.1. Заполните ячейки A1, A2, A3 любыми значениями.

11.2. Вернитесь в редактор VBA. Установите курсор внутри созданной вами программы и нажмите клавишу F5 или выполните команду **Run/Run Sub**, или нажмите соответствующую кнопку на панели элементов Стандартная.

11.3. Вернитесь в рабочий лист Excel и убедитесь в том, что ячейки A1, A2, A3 очищены.

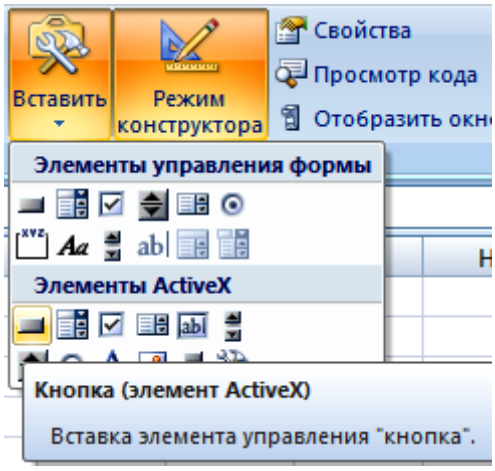
12. Научимся запускать программу с рабочего листа.

12.1. Откройте вкладку **Разработчик**

12.2. Нажмите кнопку Режим конструктора 

на панели **Элементы управления**

12.3. Нажмите кнопку Вставить и выберите элемент Кнопка в элементах управления ActiveX и перенесите его на рабочий лист.



12.4. Выполните двойной щелчок на кнопке. В окне редактирования кода появляется первая и последняя строки новой процедуры.

```
Private Sub CommandButton1_Click()

End Sub
```

Эта процедура будет обрабатывать событие щелчок на кнопке, которая по умолчанию была названа CommandButton1. Такая процедура называется событийной, и ее название состоит из двух частей: названия объекта и соответствующего события.

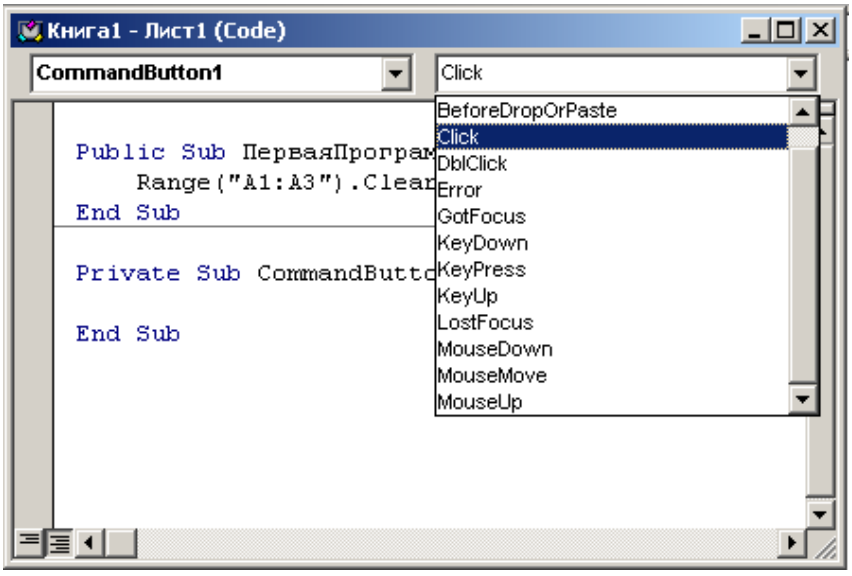


Рисунок 15. Список событий, связанных с объектом CommandButton1

Два раскрывающихся списка в верхней части окна редактора кода облегчают ориентацию в событиях, которые связаны с выбранным объектом. Так на рисунке 15 показан список событий, связанных с объектом CommandButton1.

12.6. Вернитесь на рабочий лист и на панели Элементы управления нажмите кнопку Свойства  .

12.7. В окне свойств измените значение свойства Caption. Вместо Commandbutton1 наберите, например, ПерваяПрограмма. Закройте окно свойств.

12.8. Выйдите из режима конструктора, снова нажмите кнопку Режим конструктора  на панели Элементы управления.

12.9. Заполните ячейки A1, A2, A3 и нажмите кнопку ПерваяПрограмма.

13. Выполните следующие задания самостоятельно.

13.1. Создайте кнопку на рабочем листе, при нажатии на которую очищался бы только формат ячеек. Необходимый метод найдите самостоятельно в списке методов объекта Range.

13.2. Создайте кнопку на рабочем листе, при двойном щелчке на которую очищался бы столбец рабочего листа. Необходимую событийную процедуру создайте, используя список компонентов и список соответствующих им событий в окне редактора кода. (Событие DblClick).

Лабораторная работа №2

Тема: Ввод/вывод данных с помощью диалоговых окон. Запись арифметических и логических выражений на языке VBA

1. Откройте новую рабочую книгу.

2. Войдите в редактор VBA.

3. В окне проекта выберите объект Лист1 и войдите в окно редактирования кода объекта Лист1.

4. Решим следующую задачу: вывести в диалоговое окно сегодняшнюю дату.

Для решения поставленной задачи воспользуемся материалом пункта 15.

Для вывода сегодняшней даты воспользуемся простым окном-сообщением. Кроме того, для того чтобы соединить две строки воспользуемся операцией конкатенции (соединения) &, а для получения сегодняшней даты функцией Date.

4.1. Выполните команду **Insert/Procedure**. в диалоговом окне введите имя процедуры Prog15_1.

4.2. Далее начните вводить программный код процедуры в соответствии с рисунком 16. После ввода запятой при задании фактических параметров процедуры MsgBox появляется подсказка о возможных значениях следующего параметра процедуры MsgBox. Выбираем vbInformation для того чтобы в окне вывода появилась пиктограмма, которая соответствует информационному сообщению.

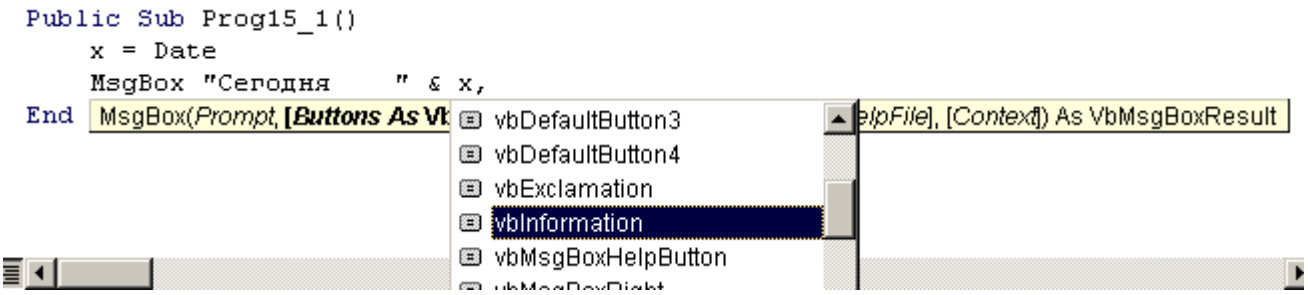


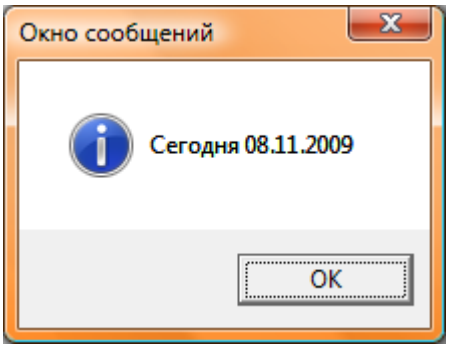
Рисунок 16. Всплывающая интеллектуальная подсказка

Далее вводим параметры в соответствии с указанным ниже кодом.

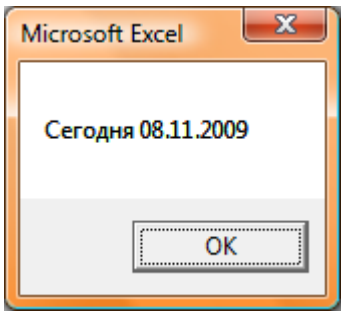
```
Public Sub Prog15_1()
    x = Date
    MsgBox "Сегодня " & x, vbInformation, "Окно сообщений"
End Sub
```

4.3.Выполните отладку программы (**Debug/Compile VBAProject**)

4.5. Запустите созданную процедуру. В результате появится диалоговое окно представленное на рис. 17.а.



а)



б)

Рисунок 17.

Если в функции MsgBox использовать только один параметр (**`MsgBox "Сегодня " & x)`**, получим окно сообщений представленное на рис.17.б.

5. Разработаем диалог с компьютером, который позволит пользователю сравнить свой ответ с правильным. Например, предложим пользователю решить следующую задачу: какое наибольшее десятичное число можно записать тремя цифрами в пятиричной системе счисления?

5.1. Выполните команду **Insert/Procedure**. В диалоговом окне введите имя процедуры Dialog.

5.2. Введите приведенный ниже программный код.

```
Public Sub Dialog()
  Dim st As String
  st = "Какое наибольшее десятичное число можно " _
  & "записать тремя цифрами в пятиричной системе счислени?"
  st = InputBox(st)
  MsgBox "Правильный ответ - 124" _
  & vbCr & "Ваш ответ - " & st
End Sub
```

Разберем как работает данная процедура.

`Dim st As String`	Резервируется ячейка оперативной памяти для хранения строковой переменной
`st = "Какое наибольшее десятичное число можно " _` `& "записать тремя цифрами в пятиричной системе счислени?"`	Переменной st присваивается строковая константа. Причем в коде выполнен перенос строковой константы, она представлена как результат конкатенции двух строковых констант (см. пункт 14)

`st = InputBox(st)`	Вызывается функция InputBox. В результате появляется диалоговое окно представленное на рис. 18. Переменной st будет присвоено значение строки, которую пользователь введет в текстовое поле (например, число 124).
`MsgBox "Правильный ответ - 124" _` `  & vbCr & "Ваш ответ - " & st`	Вызывается оператор MsgBox для вывода простого окна сообщений (см. рис.19). Строковая константа представлена как результат конкатенции двух строковых констант и встроенной константы VBA vbCr (см.замечание ниже). Использование константы позволяет при выводе разбить строку на две.

Замечание	Для обозначения некоторых наиболее часто употребляемых клавиш существуют встроенные константы VBA, например для клавиши <Backspace> — vbBack, клавиши <Tab> — vbTab, клавиши <Enter> — vbCr.

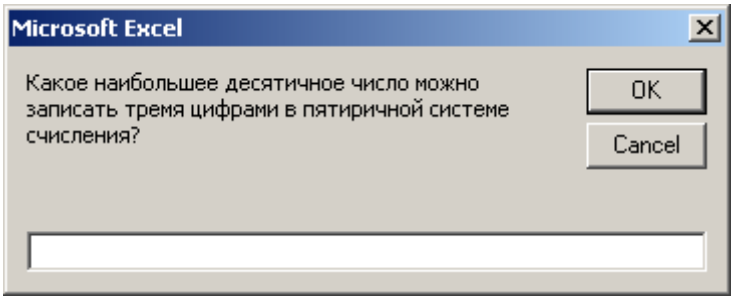


Рисунок 18. Окно ввода ответа

5.3. Выполните команду **Debug/Compile VBAProject** и запустите процедуру, нажав кнопку F5.

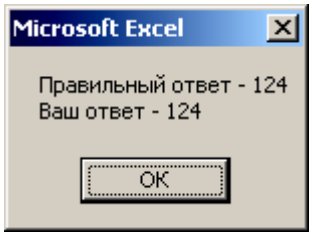


Рисунок 19. Окно сообщений

6. Далее научимся записывать арифметические выражения на языке VBA и получать их значения.

Арифметическое выражение — это последовательность, состоящая из числовых констант, переменных, функций, знаков арифметических операций и круглых скобок. Значением арифметического выражения всегда является число.

Арифметическое выражение в VBA записывается в одну строку.

Арифметические операции, которые используются в VBA, приведены в таблице 4. Порядок выполнения операций в арифметическом выражении задается круглыми скобками. При их отсутствии установлен следующий порядок старшинства арифметических операций: вычисление функции, операция возведения в степень, операции умножения и деления, операции сложения и вычитания.

Операции одинакового старшинства выполняются в том порядке, в котором они следуют в выражении, т.е. слева направо.

Математические функции, которые могут входить в арифметическое выражение, представлены в таблице 5. При записи функции ее аргумент обязательно берется в круглые скобки.

В качестве разделителя целой и дробной части чисел служит точка.

Рассмотрим примеры.

Арифметическое выражение	Запись в VBA
$\sqrt{e^{2x}+e^{3x^2+4}}$	Sqr(Exp(2 * x) + Exp(3 * x ^ 2 + 4)) (Exp(2 * x) + Exp(3 * x ^ 2 + 4)) ^ (1 / 2) Иногда арифметическое выражение может быть записано несколькими способами в VBA. В данном примере корень квадратный может быть записан как с помощью функции Sqr, так и с помощью операции возведения в степень (показатель степени равен ½).
$\frac{\sin x^2+\cos^2 x}{\lvert\lg(x-1{,}2)^2\rvert}$	(Sin(x ^ 2) + Cos(x) ^ 2) / Abs(Log(x - 1.2) ^ 2 / Log(10)) Обратите внимание, что десятичный логарифм мы заменили дробью, содержащей натуральный логарифм. VBA содержит запись только натурального логарифма. Числитель дроби, содержащий сумму, взят в скобки.

6.1. Создайте в модуле редактирования кода объекта Лист1 следующую процедуру.

```
Public Sub Arifm()
```

```
   Dim x As Single, y As Single
   x = InputBox("введите число")
   y = Sqr(Exp(2 * x) + Exp(3 * x ^ 2 + 4))
   MsgBox "y=" & y
End Sub
```

Для ввода значения переменной *x* используется окно ввода. Для вывода значения арифметического выражения простое окно сообщений.

6.2. Выполните команду **Debug/Compile VBAProject** и запустите процедуру Arifm, нажав кнопку F5.

6.3.В текстовое поле диалогового окна введите число 1.3 (разделителем целой и дробной части введите точку) и нажмите кнопку OK.

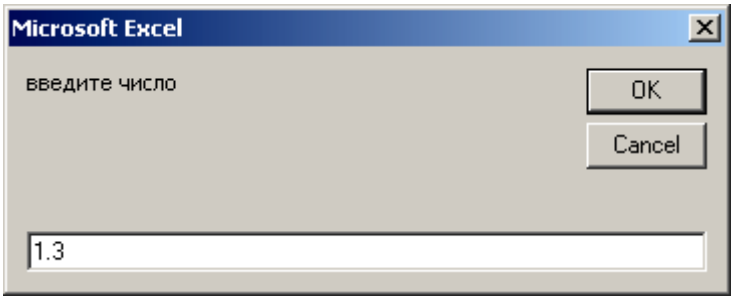


Рисунок 20. Окно ввода числа

На экране появляется сообщение об ошибке, в котором указывается код ошибки и краткое описание ошибки (см. рис.21).

Нажмите кнопку Debug.

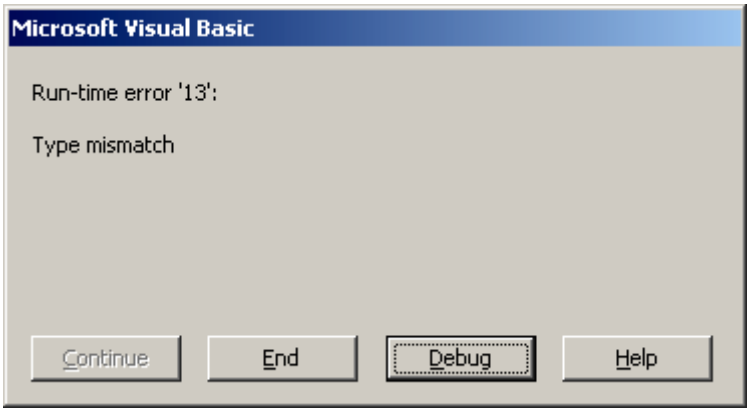


Рисунок 21. Окно вывода сообщения об ошибке

Замечание	Если в сообщении об ошибке отсутствует кнопка Debug, следует выполнить предварительно команду **Tools/Options/**Вкладка **General/**Выбрать переключатель **Break on all Errors.** Повторите пункт 6.3. сначала.

Управление передается в процедуру. Желтым цветом выделяется строка, содержащая ошибку (рис.22).

В данном случае неправильно введено значение переменной *x*.

6.4. Завершите работу программы, нажав кнопку Reset  на стандартной панели инструментов.

```
Public Sub Arifm()
    Dim x As Single, y As Single
    x = InputBox("введите число")
    y = Sqr(Exp(2 * x) + Exp(3 * x ^ 2 + 4))
    MsgBox "y=" & y

End Sub
```

Рисунок 22. Выделение строки с ошибкой

6.5. Вновь запустите программу и в качестве значения *x* введите 1,3 (разделитель целой и дробной части – запятая).

6.6. На экране появляется сообщение о значении арифметического выражения при введенном значении *x* (рис.23). Ошибка ввода исправлена, мы получили результат работы программы.

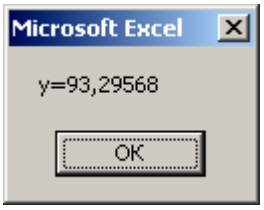


Рисунок 23. Вывод значения выходного параметра

6.7. Исправьте программный код процедуры Arifm, чтобы в результате работы программы появлялось диалоговое окно представленное на рис. 24.

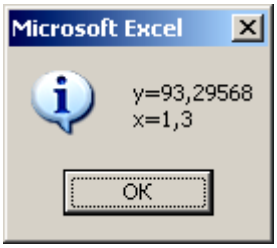


Рисунок 24. Вывод значений выходного и входного параметров

6.8. Напишите процедуры для вычисления значений выражения всех арифметических выражений, приведенных в примерах.

7. Далее научимся записывать логические выражения на языке VBA и получать их значения.

Простейшее логическое выражение — это отношение типа $a < b$ (меньше), $a > b$ (больше), $a = b$ (равно), $a <> b$ (не равно), $a <= b$ (меньше либо равно), $a >= b$ (больше либо равно), где *a, b* константа или переменная (операнды).
Например,

запись математическая	запись на языке VB
$b^2 - 4ac > 0$	b^2-4*a*c>0
$i \neq j$	i<>j
$\sqrt{x} \geq 2{,}83$	Sqr(x)>=2.83

Результат вычисления логического выражения является переменная типа Boolean, т.е. логическое выражение принимает значения истина (true) или ложь (false).

В зависимости от операндов различают сравнения: алгебраические (означающие сравнение числовых величин с учетом знаков); символьные (означающие последовательное попарное сравнение символов слева направо).

Логические выражения кроме отношений могут содержать логические операции: инверсию (Not), конъюнкцию (And), дизъюнкцию (Or), строгую дизъюнкцию (Xor), эквивалентность (Equ) и импликацию Imp).

Запишем логические выражения, позволяющие проверить, входят ли значения переменных в область определения арифметических выражений.

Арифметическое выражение	Условие	Запись условия на VBA
$z = \frac{e^x}{a \ln x}$	(a≠0) и ((x≠1) и (x>0))	a<>0 And x<>1 And x>0 Обратите внимание, что скобки опущены, так как при вычислении значения выражения соблюдается приоритет выполнения операций: сначала операции отношения, затем конъюнкция.
$z = \frac{2}{\sqrt{x^2 + x - 6}}$	(x<-3) или (x>2)	x<-3 or x>2

7.1. Создайте в модуле редактирования кода объекта Лист1 следующую процедуру.

```
Public Sub Logic()
   Dim x As Single, a As Single, y As Boolean
   x = InputBox("Введите значение x")
   a = InputBox("Введите значение a")
   y = a <> 0 And x <> 1 And x > 0
   MsgBox "Условие   " & y
```

```
End Sub
```

Обратите внимание, что переменная *y* объявлена как логическая переменная, т.е. эта переменная может принимать только два значения True (истина) или False (ложь).

7.2. Выполните команду **Debug/Compile VBAProject** и запустите процедуру Logic, нажав кнопку F5.

7.3. Введите значения переменных *a* и *x*. Например *a=3*, *x=4*. На экране появляется диалоговое окно представленное на рис. 25,а. Если же ввести, например, *a=0*, *x=4*, то появится диалоговое окно, представленное на рис. 25,б.

а) б)

Рисунок 25. Окно вывода значения логической переменной

8.Выполните задание №1 контрольной работы.

9. Выполните задание №2 контрольной работы.

Лабораторная работа № 3

Тема: Автоматизация рабочих листов при помощи макросов и создание настраиваемой панели инструментов.

Написание программы, это, в каком-то смысле, обучение компьютера делать то, что вам нужно от него. Для быстрого получения чернового варианта кода, используемого в создаваемом приложении, отлично подходит макрорекодер.

Макрорекодер — это транслятор, создающий программу (макрос) на языке VBA, которая является результатом перевода на языке VBA действий пользователя с момента запуска макрорекодера до окончании записи макроса.

Задача: Создать макрос, формирующего нижний и верхний колонтитул рабочего листа, создать подпрограмму, обеспечивающую перелистывание рабочих листов, создать подпрограмму, отображающую информацию в диалоговое окно, связать все подпрограммы с кнопками настраиваемой панели инструментов.

Порядок выполнения

1. Открыть новую рабочую книгу. Активный рабочий лист Лист1.

2. Создание макроса, формирующего нижний и верхний колонтитул рабочего листа.

2.1. Откройте вкладку **Разработчик.** На панели **Код нажмите** кнопку **Запись макроса.**

2.2. В появившемся диалоговом окне указать следующее:

Имя макроса — Колонтитул, **Сохранить в**: Эта книга, **Сочетание клавиш** Ctrl+[Shift]+K (в текстовое поле введите английскую букву К), **Описание** — макрос записан <введите сегодняшнюю дату>.

Нажмите кнопку OK.

Теперь все ваши действия будут записываться с помощью языка VBA. Перейдем к созданию нижнего и верхнего колонтитула.

2.3. На вкладке **Разметка страницы** на панели **Параметры страницы** нажать кнопку **Печать заголовки**.

Вкладка Страница – ориентация альбомная

2.4. Открыть вкладку Колонтитулы и нажать кнопку Создать верхний колонтитул.

В появившемся диалоговом окне текстовые поля заполнить согласно рис.26. и нажать OK.

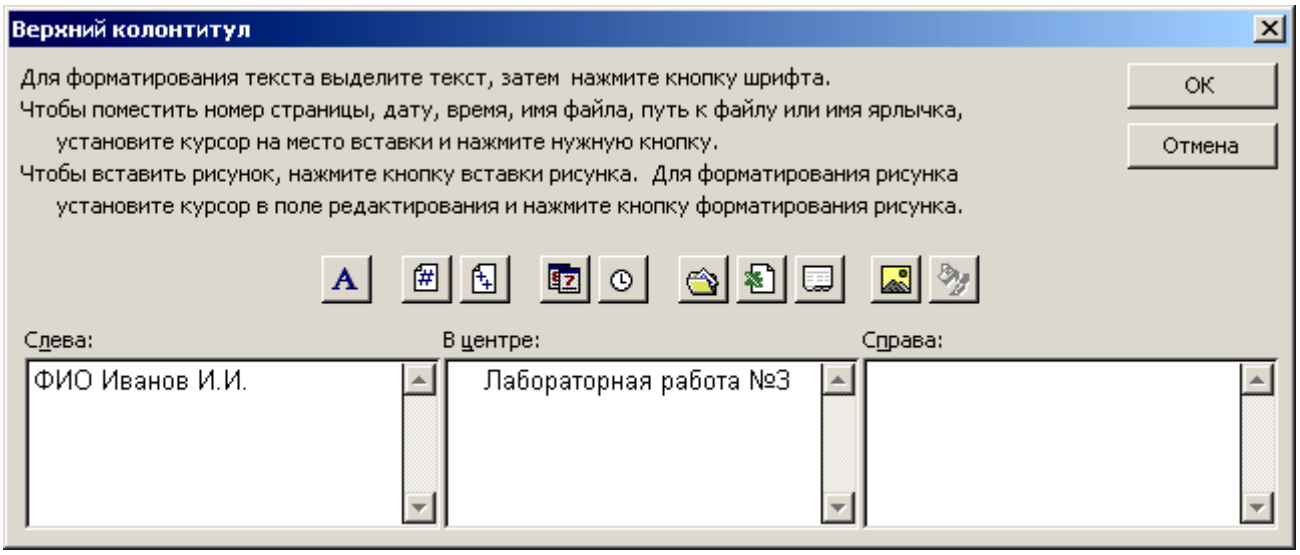


Рисунок 26. Диалоговое окно создания верхнего колонтитула

2.5. Нажать кнопку Нижний колонтитул. В появившемся диалоговом окне текстовые поля заполнить согласно рис.27 (установите курсор в центральном поле, последовательно нажмите кнопки  и ) и нажать OK.

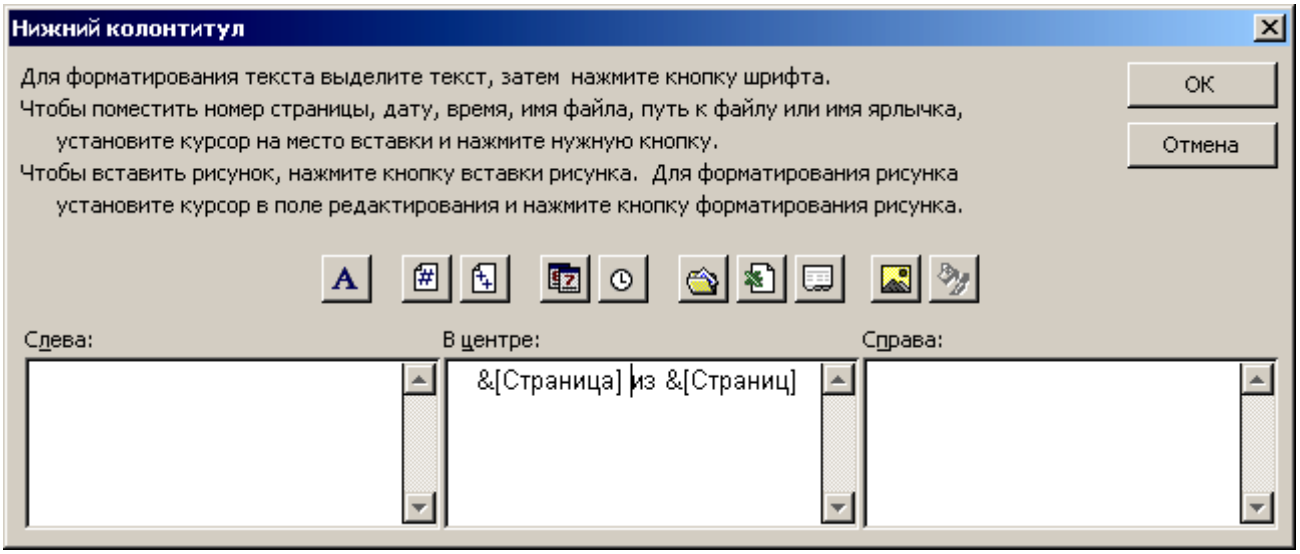


Рисунок 27. Диалоговое окно создания нижнего колонтитула

2.6. Открыть вкладку **Разработчик.** На панели **Код** нажать кнопку **Остановить запись.**

2.7. Войти в редактор VBA (Alt+F11), в окне проекта появился Модуль1. В окне редактирования кода модуля (выполните двойной щелчок на объекте модуль) записан макрос на языке VB.

Сгенерированный макрос выглядит с.о.

```
Sub Колонтитул()
' Макрос3 Макрос
```

```
' Макрос записан 23.10.2019
' Сочетание клавиш: Ctrl+Shift+K
    With ActiveSheet.PageSetup
        .PrintTitleRows = ""
        .PrintTitleColumns = ""
    End With
    ActiveSheet.PageSetup.PrintArea = ""
    With ActiveSheet.PageSetup
        .LeftHeader = "ФИО Иванов И.И."
        .CenterHeader = "Лабораторная работа №3"
        .RightHeader = ""
        .LeftFooter = ""
        .CenterFooter = "&P из &N"
        .RightFooter = ""
        .LeftMargin = Application.InchesToPoints(0.787401575)
        .RightMargin = Application.InchesToPoints(0.787401575)
        .TopMargin = Application.InchesToPoints(0.984251969)
        .BottomMargin = Application.InchesToPoints(0.984251969)
        .HeaderMargin = Application.InchesToPoints(0.5)
        .FooterMargin = Application.InchesToPoints(0.5)
        .PrintHeadings = False
        .PrintGridlines = False
        .PrintComments = xlPrintNoComments
        .PrintQuality = 600
        .CenterHorizontally = False
        .CenterVertically = False
        .Orientation = xlPortrait
        .Draft = False
        .PaperSize = xlPaperA4
        .FirstPageNumber = xlAutomatic
        .Order = xlDownThenOver
        .BlackAndWhite = False
        .Zoom = 100
        .PrintErrors = xlPrintErrorsDisplayed
    End With
End Sub
```

Строки кода, начинающиеся со знака апострофа, являются комментариями и игнорируются Excel. Комментарии создаются на основе информации, введенной в окне Запись макроса (сюда, в частности, относится сочетание клавиш, использующееся для вызова макроса).

Замечание	Комментарий не определяет сочетание клавиш. Другими словами, изменив в комментарии сочетание клавиш, вы ничего не добьетесь. Изменить сочетание клавиш можно только с помощью диалогового окна Макрос.

Обратите внимание на оператор **With – End With,** который позволяет программисту выполнить несколько операций над одним и тем же объектом без повторений этого объекта при работе с его свойствами и методами. В нашем приме-

ре оператор **With – End With** применен для **`ActiveSheet.PageSetup`** (параметры страницы активного рабочего листа).

2.8. Вернитесь в рабочую книгу MS Excel.

2.9. Введите в любую ячейку рабочего листа Лист1 любые данные, например, в ячейку А1 текст «Макрос».

2.10. Перейдите на вкладку **Файл**. В списке выберите команду **Печать.**

Excel позволяет посмотреть, как будет выглядеть страница при печати, в данном случае мы добавили верхний и нижний колонтитулы.

2.11. Перейдите на Лист2. Добавим колонтитулы на Лист2, выполнив макрос Колонтитул. Нажмите комбинацию клавиш Ctrl+Shift+K (раскладка клавиатуры английская) или откройте вкладку **Разработчик** и нажмите кнопку **Макросы** на панели **Код**.

В появившемся окне отображается список макросов. Выбрав необходимый макрос его можно выполнить, просмотреть код (войти), изменить, удалить или изменить параметры.

Заполните любую ячейку рабочего листа Лист2 и выполните предварительный просмотр. Страница теперь также имеет верхний и нижний колонтитулы.

3. Создание подпрограммы, обеспечивающие перелистывание рабочих листов

3.1. Переименовать лист Лист1 в «Счета», Лист2 – в «Основной», Лист3 – в «Итоги»

3.2. Войти в редактор VBA (Alt+F11).

3.3. В редакторе VBA выполнить команду **Insert/Module**.

3.4. Выполнить команду **Insert/Procedure.**

3.5. В появившемся диалоговом окне указать следующее:

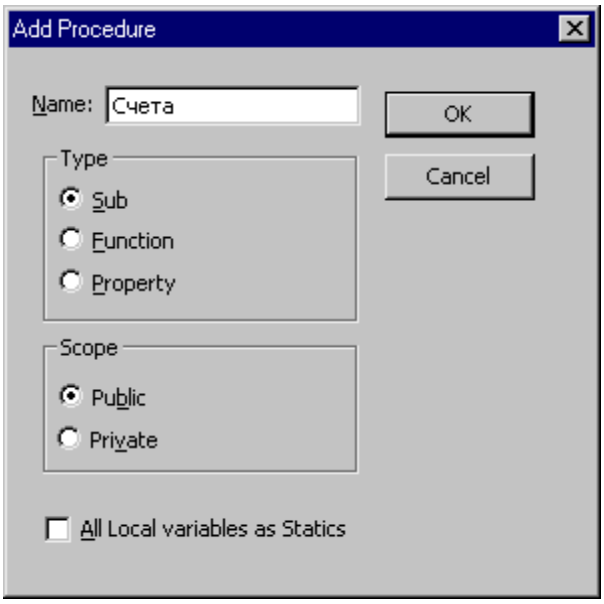


Рисунок 28. Диалоговое окно Создание процедуры

3.5. Поместите следующий код программы внутри созданной процедуры.

```
Sheets("Счета").Activate
```

(Метод Activate активизирует указанный рабочий лист).
Новую подпрограмму можно найти в разделе General Модуля (см. рис.29).

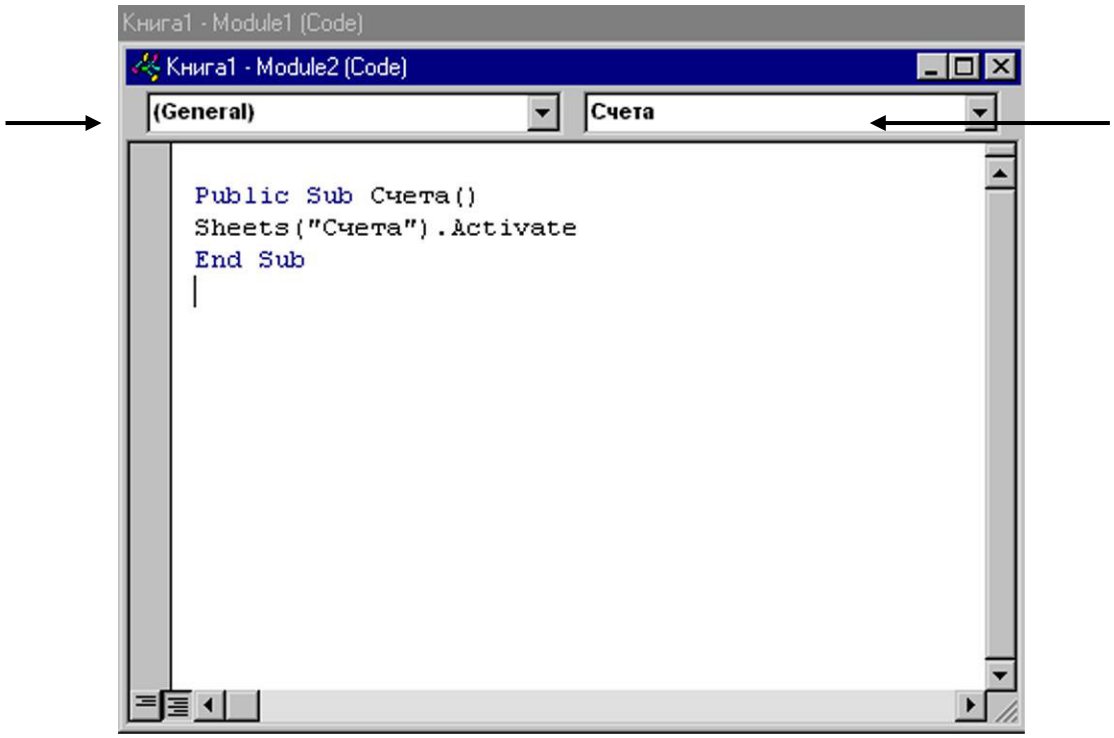


Рисунок 29. Метод Activate активизирует указанный рабочий лист

3.6. Создать аналогично еще две процедуры для активизации листов «Основной», «Итоги».

```
Public Sub Основной()
Sheets("Основной").Activate
End Sub
```

```
Public Sub Итоги()
Sheets("Итоги").Activate
End Sub
```

3.7. Выполнить команду **Debug/Compile VBA Project**.

4. Создание программы, отображающую информацию в диалоговые окна.

4.1. В редакторе VBA выполнить команду **Insert/Module**

Т.о. в окне проекта отобразится уже три модуля. Первый модуль содержит макрос, второй — процедуры активизации книг.

4.2. В окно редактирования кода Модуля3 вставить следующую процедуру

```
Public Sub Info()
    MsgBox "Лабораторная работа №3", _
    vbInformation, "Информация"
End Sub
```

4.3.Выполнить команду **Debug/Compile VBA Project**.

5. Назначить кнопки панели инструментов для запуска созданных подпрограмм.

5.1. Кнопки для запуска программ разместим на **Панели быстрого доступа**.

Выберите вкладку **Параметры**, затем нажмите Параметры.

5.2. В окне **Параметры Excel** выбрать опцию **Панели быстрого доступа.**

В списке **Настройка панели быстрого доступа** выберите название своей книги.

В списке **Выбрать команды из** выбрать **Макросы.** В появившемся списке последовательно выбирайте программы (макросы) **Info**, **Итоги**, **Основной**, **Счета** и с помощью кнопки **Добавить** добавить в список кнопок **Панели быстрого доступа**. Измените иконки кнопок с помощью кнопки **Изменить**.

5.3. Закрыть диалоговое окно, нажав кнопку **ОК.**

На **Панели быстрого доступа** появились четыре дополнительные кнопки.

6. Осуществить переход на лист Итоги и выполнить макрос Колонтитул. Воспользовавшись опрцией Предварительный просмотр, обнаружить результаты работы Макроса Колонтитул (предварительно введите данные в любую ячейку рабочего листа).

7. Осуществите перелистывание рабочих листов с помощью созданных кнопок на панели инструментов (панели быстрого доступа).

8. Выполнить Макрос INFO, воспользовавшись кнопкой Информация.

9. Выполнить самостоятельно. В новой рабочей книге создать макрос, формирующего нижний и верхний колонтитул рабочего листа (в верхний колонтитул по-

местить следующую информацию: ФАМ, группа, шифр, «Контрольная работа по информатике», в нижний колонтитул номер страницы, название рабочего листа), создать подпрограмму, обеспечивающую перелистывание рабочих листов, создать подпрограмму, отображающую информацию в диалоговое окно (текст в диалоговом окне: «Контрольная работа по информатике выполнена студентом <ФАМ> группы <номер группы>»), связать все подпрограммы с кнопками настраиваемой панели инструментов.

Лабораторная работа №4

Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. *Программирование алгоритмов линейной структуры*

Алгоритм – система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (входной информации) в желаемый результат (выходную информацию) за конечное число шагов.

Способы описания алгоритмов.

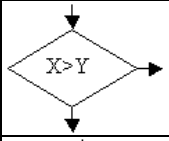
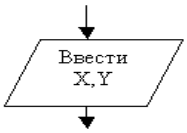
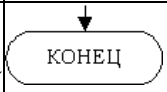
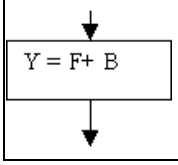
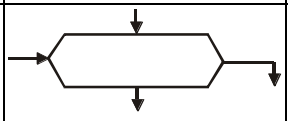
1. Словесно-формальное описание алгоритма, т.е. описание алгоритма с помощью слов и формул. Это наиболее простой способ.

2. Графическое описание алгоритма, т.е. описание алгоритма с помощью схем алгоритмов.

Граф, в котором вершинам соответствуют шаги, а ребрам переходы между шагами, называется блок-схемой алгоритма. Его вершины могут быть двух видов: вершины, из которых выходит одно ребро (их называют операторами), и вершины, из которых выходят два ребра (их называют логическими

условиями). Кроме того, имеется единственный оператор конца, из которого не выходит ни одного ребра, и единственный оператор начала

Для наглядности операции разного вида изображаются в блок-схеме различными геометрическими фигурами. Внутри блоков указывается поясняющая информация, характеризующая выполняемые ими действия.

	Блок начала алгоритма		Ветвление (проверка условий)
	Блок ввода или вывода		Блок окончания алгоритма
	Блок действия		Начало цикла

Графическое представление алгоритма, как правило, связывают со структурным подходом в составлении алгоритмов.

Этап проектирования алгоритма следует за этапом формального решения задачи, на котором определены входные и выходные данные, а также зависимости между ними.

Как и при разработке любой сложной системы, при построении алгоритма используют дедуктивный и индуктивный методы. При дедуктивном методе рассматривается частный случай общеизвестных алгоритмов. Индуктивный метод применяют в случае, когда не существует общих алгоритмических решений. Широкое применение получило структурное программирование "сверху - вниз". Эта технология программирования представляет собой процесс пошагового разбиения алгоритма на все более мелкие части с целью получить такие элементы, для которых можно легко написать конкретные предписания.

Структурная алгоритмизация основывается на двух принципах:

1) последовательная детализация "сверху - вниз";

2) ограниченность базового набора структур для построения алгоритмов любой степени сложности.

Из принципов вытекают требования структурного программирования:

1) программа должна составляться мелкими шагами; таким образом, сложная задача разбивается на достаточно простые, легко воспринимаемые части;

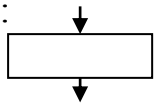
2) логика программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Можно перечислить основные свойства и достоинства структурного программирования:

- ✓ возможность преодоления барьера сложности программ;
- ✓ возможность демонстрации правильности программ на различных этапах решения задачи;
- ✓ наглядность программ;
- ✓ простота модификации (внесение изменений) программ.

Базовый набор структурной алгоритмизации содержит линейные, разветвляющиеся и циклические структуры.

Линейный алгоритм - набор команд, выполняемых последовательно во времени, друг за другом. Такой порядок выполнения блоков называется **естественным**. Такие алгоритмы применяют для описания обобщенного решения задачи в виде последовательности модулей. Блок-схема базовой структуры следования:



1. Решим следующую задачу: поменять местами содержимое двух ячеек рабочего листа Excel, например, A1и B1.

1.1. Разработка алгоритма решения задачи.

Алгоритм решения задачи представлен на рисунке 30. Основная идея решения поставленной задачи заключается в использовании дополнительной ячейки, в которой запоминается значение переменной *x*. Если сразу выполнить присваивание *x*=*y*, то исходное значение переменной *x* будет утеряно. Аналогично, если первым выполнить оператор *y*=*x*, то пропадет исходное значение *y*.

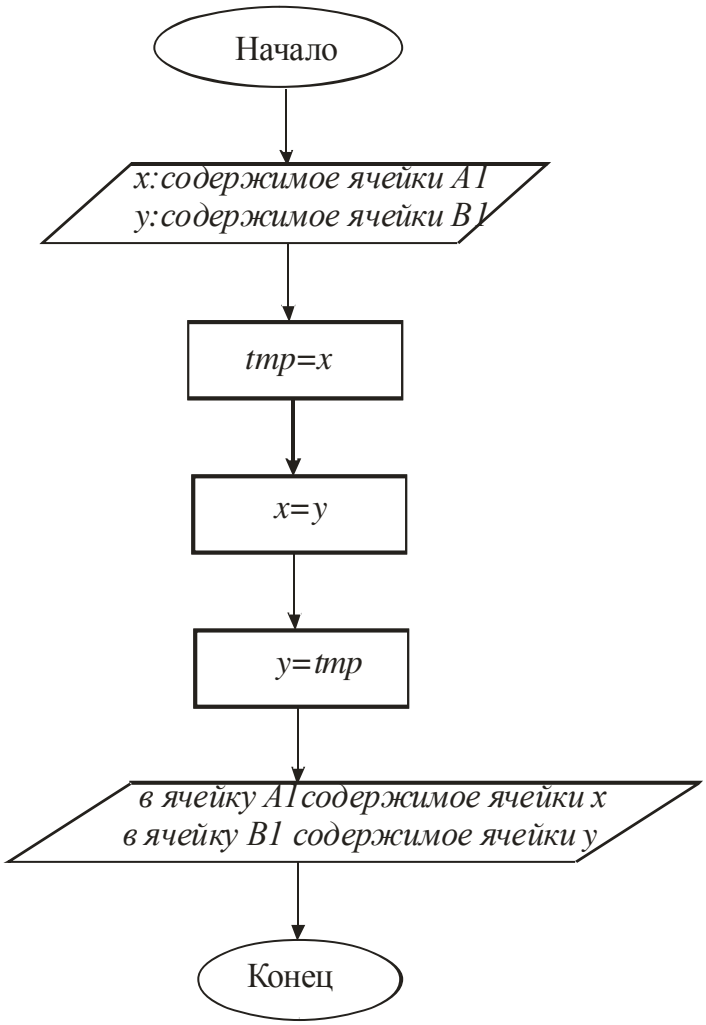


Рисунок 30.

Словесно-формальная запись алгоритма представлена ниже.

1. В ячейку *x* оперативной памяти запишем содержимое ячейки A1.
2. В ячейку *y* оперативной памяти запишем содержимое ячейки B1.
3. В ячейку tmp запишем содержимое ячейки *x.*
4. В ячейку *x* запишем содержимое ячейки *y.*
5. В ячейку *y* запишем содержимое ячейки *tmp.*
6. В ячейку A1 рабочего листа запишем содержимое ячейки *x*.
7. В ячейку B1 рабочего листа запишем содержимое ячейки *y*.

Для программирования данного алгоритма достаточно воспользоваться оператором присваивания (см. пар.12).

Запишем, как будет выглядеть каждый шаг алгоритма на языке VBA.

1. В ячейку *x* оперативной памяти запишем содержимое ячейки A1.	1. x= Range("A1")
2. В ячейку *y* оперативной памяти запишем содержимое ячейки B1.	2. y = Range("B1")
3. В ячейку tmp запишем содержимое ячейки *x.*	3. tmp = x
4. В ячейку *x* запишем содержимое ячейки *y*	4. x = y
5. В ячейку *y* запишем содержимое ячейки *tmp*	5. y = tmp
6. В ячейку A1 рабочего листа запишем содержимое ячейки *x*.	6. Cells(1, 1) = x

7. В ячейку B1 рабочего листа запишем содержимое ячейки *y*.	7. Cells(1, 2) = y

Обратите внимание, что один и тот же объект, ячейку рабочего листа, можно описать по-разному. Так Range("A1") и Cells(1,1) являются описанием одного и того же объекта – ячейки рабочего листа A1 (см. пар.2).

1.2. Перейдем к реализации алгоритма в среде VBA.

1.2.1. Запустите Excel.

1.2.2. Для открытия редактора VBA нажмите клавиши <Alt+F11>

1.2.3. В окне Project Explorer выполните двойной щелчок на объекте Лист1.

1.2.4. В появившемся окне редактирования кода введите следующую процедуру (не забудьте выполнить команду **Insert/Procedure**).

```
Public Sub Замена()
    Dim tmp As Variant, x As Variant, y As Variant
    x = Range("A1")
    y = Range("B1")
    tmp = x
    x = y
    y = tmp
    Cells(1, 1) = x
    Cells(1, 2) = y
End Sub
```

Обратите внимание, что тип переменных x, y, и tmp указан как Variant. Такой выбор связан с возможностью наличия в ячейках рабочего листа разнородной информации.

1.2.5. Для проверки синтаксиса процедуры выполните команду **Debug/Complete VBAProject.**

1.2.6. В ячейки A1 и B1 рабочего листа Лист1 введите числа 2 и 3 соответственно.

1.2.7. Редактор VBA содержит великолепный отладчик программного кода, который позволит нам не только устранить недостатки программного кода, но и поможет разобраться, как работает программа в VBA. Воспользуемся возможностью пошагового выполнения кода.

Разместите курсор внутри процедуры Замена и выберите команду меню **Debug/Step Into** (**Отладка/Пошаговое выполнение**) или нажмите клавишу F8.

Сейчас редактор VBA находится в режиме пошагового выполнения кода. Строка, которая будет выполнена следующей, выделена желтым цветом. Кроме того, на нее указывает желтая стрелка, расположенная слева (см. рис.31).

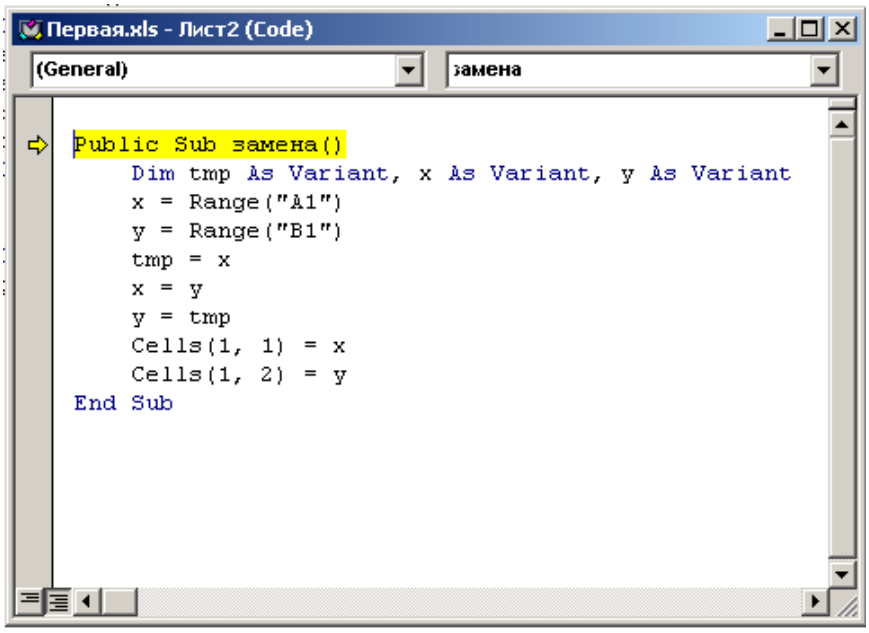


Рисунок 31. Отладчик готов выполнить первую строку процедуры

1.2.8. В режиме пошагового выполнения кода появляется возможность с помощью указателя мыши узнать значение выражения или переменной. Подведите указатель мыши к переменной *x* и задержите в таком положении пару секунд. На экране появилась подсказка, содержащая текущее значение переменной *x* (см. рис.32). Ключевое слово Empty означает, что переменная в данный момент пуста. Аналогично посмотрите значения других переменных.

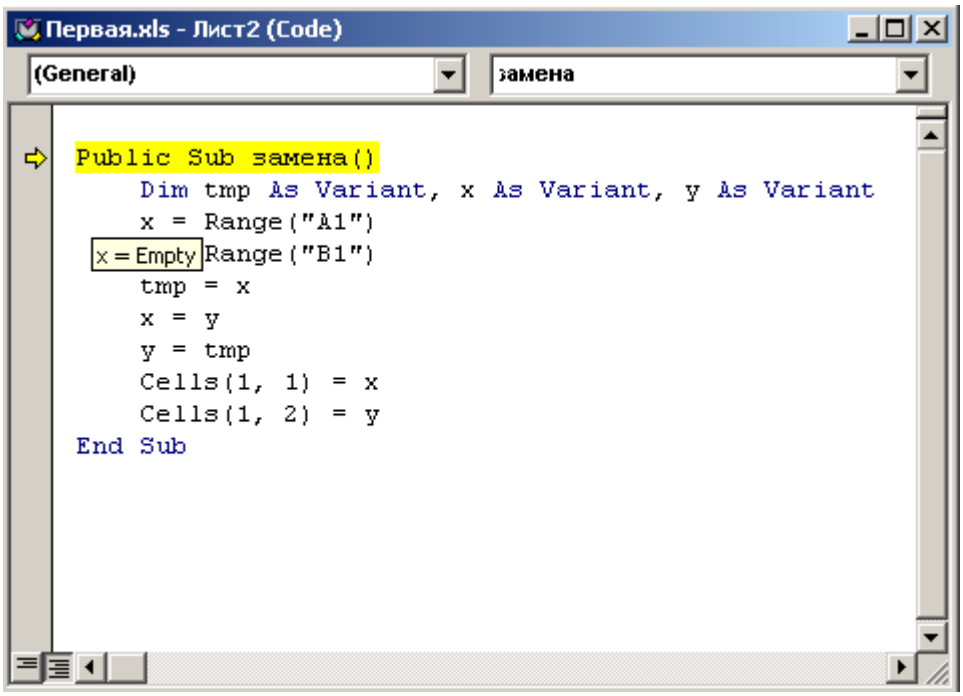


Рисунок 32. Подсказка в режиме пошагового выполнения кода

1.2.9. Нажмите клавишу F8. Желтым цветом выделяется строка, содержащая код

x= Range("A1").

1.2.10. Еще раз нажмите клавишу F8. Оператор **x= Range("A1**") будет выполнен (ячейка *x* получит значение ячейки A1 рабочего листа) и желтым будет подсвечена следующая строка, готовая к исполнения.

1.2.11. Подведите указатель мыши к переменной *x*. В подсказке указывается текущее значение переменной *x*, оно равно 2. Далее подведите указатель мыши к ячейкам *y* и *tmp*. Оно по прежнему равно Empty. Будем записывать текущее значение переменных в таблицу трассировки. Если значение ячейки изменилось, при выполнении строки программного кода, ячейку таблицы будем выделять серым цветом. Если ячейка в текущий момент пуста будем ставить прочерк.

Оператор	x	y	tmp
x= Range("A1")	2	—	—

1.2.12. Еще раз нажмите клавишу F8. Посмотрите чему равно значение переменных *x*, *y*, *tmp*. Результат запишите в таблицу.

Оператор	x	y	tmp
y = Range("B1")	2	3	—

1.2.13. Реализуйте пошаговое выполнение программы до конца, последовательно нажимая клавишу F8. Следите за значениями переменных. Результаты запишите в таблицу. У вас должна получиться приведенная ниже таблица трассировки.

Оператор	x	y	tmp
x= Range("A1")	2	—	—
y = Range("B1")	2	3	—
tmp = x	2	3	2
x= y	3	3	2
y = tmp	3	2	2
Cells(1, 1) = x	3	2	2
Cells(1, 2) = y	3	2	2

Обратите внимание, что последние два оператора тела процедуры не меняют значения переменных *x* и *y*, а передают их значения в ячейки рабочего листа.

1.2.14. Убедитесь, что ячейки A1 и B1 рабочего листа Лист1 поменяли свои значения.

1.3. Выделим алгоритм обмена значений двух ячеек в отдельную процедуру и научимся вызывать процедуру с параметрами.

1.3.1. Процедуры доступные из любых объектов проекта хранятся в стандартных модулях. Выполните команду **Insert/Module**. Откроется окно стандартного модуля.

1.3.2. Выполните команду **Insert/Procedure**. Назовите новую процедуру ZamenaXY. В строке заголовка запишите параметры, которые являются одновре-

менно и входящими, и выходящими. Для нашей задачи это названия ячеек оперативной памяти, содержание которых надо поменять местами. В тело процедуры введите программный код в соответствии с приведенным ниже.

```
Public Sub ZamenaXY(x As Variant, y As Variant)
    Dim tmp As Variant
    x= Range("A1")
    y = Range("B1")
    tmp = x
    x= y
    y = tmp
End Sub
```

1.3.3. На рабочем листе Лист2 создайте кнопку **CommandButton1**. Назовите ее Замена. Свяжите с ней следующую событийную процедуру (выполните двойной щелчок на кнопке CommandButton1 и войдите в окно редактора кода).

```
Private Sub CommandButton1_Click()
    a = Range("A1")
    b = Range("B1")
    ZamenaXY a, b
    Cells(1, 1) = a
    Cells(1, 2) = b
End Sub
```

Разберем, как работает данная процедура.

`Dim a As Variant, b As Variant`	В памяти компьютера резервируется две ячейки оперативной памяти для хранения значений переменных, имеющих тип Variant.
`a = Range("A1")`	Переменной *a* передается содержимое ячейки A1 рабочего листа Лист2
`b = Range("B1")`	Переменной *b* передается содержимое ячейки B1 рабочего листа Лист2.
`ZamenaXY a, b`	Вызывается процедура ZamenaXY. Формальным параметрам x и y присваивается значение переменных *a* и *b* и управление передается процедуре.
`Cells(1, 1) = a`	Ячейке A1 рабочего листа передается значение ячейки *a* оперативной памяти.
`Cells(1, 2) = b`	Ячейке B1 рабочего листа передается значение ячейки *b*.

1.3.4. Проверьте, как работает созданная кнопка.

2. Выполните самостоятельно задание приведенное ниже.

- По алгоритму, приведенному ниже, создайте процедуру.

- Определите, какую задачу можно решить, реализуя представленный алгоритм, какие переменные являются входящими, какие — выходящими, какой тип должны иметь переменные, чтобы избежать ошибок во время выполнения процедуры.
- Запишите таблицу трассировки.
- Создайте кнопку на рабочем листе, вызывающую созданную процедуру.

Алгоритм.

1 шаг. a = a – b

2 шаг. b = b+a

3 шаг. a = b – a

Лабораторная работа №5

Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. ***Программирование алгоритмов разветвляющей структуры***

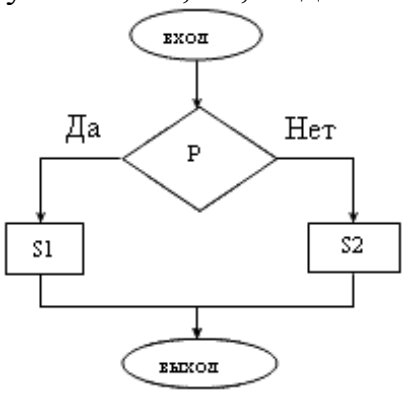
На практике редко удается представить схему алгоритма решения задачи в виде линейной структуры. Часто в зависимости от каких-либо значений промежуточных результатов необходимо организовать вычисление либо по одним, либо по другим формулам, т.е. в зависимости от выполнения некоторого логического условия вычислительный процесс должен идти по одной или по другой ветви.

Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате которого обеспечивается переход на один из двух возможных шагов, называется разветвляющимся алгоритмом.

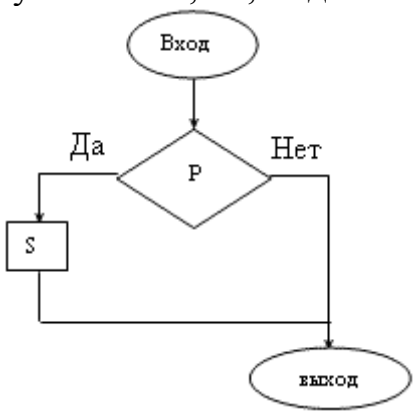
В общем случае количество ветвей в таком алгоритме разветвляющей структуры не обязательно равно двум.

Блок-схемы базовых структур разветвляющихся алгоритмов

Развилка полная
(P- условие S1, S2, S- действие)



Развилка неполная
(P- условие S1, S2, S- действие)



Каждая управляющая структура ветвления имеет один вход и один выход. Ветвления содержат блок условия P, в котором записывают логическое выражение. В зависимости от значений логического выражения выполняется либо действие S1 (истина), либо S2 (ложь). Аналогично происходит и в управляющей структуре неполного ветвления Только в этом случае, если условие P истинно, то выполняется действие S, в противном случае никаких действий не выполняется.

Для программирования алгоритмов разветвляющей структуры используется оператор ветвления (см. пункт 16).

1. Написать программу для решения следующей задачи: вычислить значение функции $z = x^3 / y$, где $y = \sin nx + 0{,}5$.

1.1. Разработка алгоритма решения задачи.

Казалось бы, решение этой задачи можно описать алгоритмом линейной структуры. Однако для удовлетворения свойств массовости и результативности алгоритма необходимо, чтобы при любых исходных данных был получен результат или сообщение о том, что задача не может быть решена при заданных исходных данных. Действительно, если $y = 0$, задача не может быть решена, так как деление на 0 невозможно. Поэтому в алгоритме необходимо предусмотреть этот случай и выдать в качестве результата информацию о том, что $y = 0$. Т.о., рассматриваемый вычислительный процесс должен иметь две ветви: в одной, если $y \neq 0$, необходимо вычислить и отпечатать значение переменной z, а в другой — вывести на печать информацию о том, что $y = 0$.

Этот вычислительный процесс можно описать условным выражением:

$$\begin{cases} \text{вычислить } z = x^3 / y, \text{ если } y \neq 0; \\ \text{вывести } y = 0, \quad \text{если} \quad y = 0. \end{cases}$$

Далее каждому шагу словесно-формального описания алгоритма поставим в соответствие операторы VBA.

Словесно-формальное описание алгоритма	Реализация алгоритма на VBA
1. Ввод переменной x	`x = InputBox("Введите x")`
2. Ввод переменной y	`y = InputBox("Введите y")`
3. Если $y \neq 0$, то вычислить $z = x^3 / y$ и вывести значение z	`If y <> 0 Then z = x^ 3 / y: MsgBox "z=" & z`
4. Если $y = 0$, то вывести $y = 0$	`If y = 0 Then MsgBox "y=" & y`

Обратите внимание на третий шаг. В операторе ветвления, если условие истинно, то выполняется два операторов, разделенных двоеточием. Кроме того мы использовали неполные условные операторы (отсутствует блок else), если условие будет ложным, то управление передается на следующий за условным оператору.

Поскольку условия $y \neq 0$ и $y = 0$ являются взаимоисключающими, то два оператора ветвления можно объединить в один оператор ветвления.

Реализация алгоритма с помощью полного оператора ветвления приведена ниже.

Словесно-формальное описание алгоритма	Реализация алгоритма на VBA
1. Ввод переменной x	`x = InputBox("Введите x")`
2. Ввод переменной y	`y = InputBox("Введите y")`

3. Если $y \neq 0$, то вычислить $z = x^3 / y$ и вывести значение z	`If y <> 0 Then`
	`  z = x^ 3 / y`
	`  MsgBox "z=" & z`
	`Else`
	`  MsgBox "y=" & y`
	`End If`

4. Если $y = 0$, то вывести $y = 0$	

1.2. Реализация решения задачи на ЭВМ.

1.2.1. Запустите Excel.

1.2.2. Для открытия редактора VBA нажмите клавиши <Alt+F11>.

1.2.3. Выполните команду **Insert/Module**.

1.2.4. Для создания новой процедуры выполните команду **Insert/Procedure**. В появившемся диалоговом окне введите название процедуры Prog1.

1.2.5. В тело процедуры введите программный код в соответствии с приведенным ниже.

```
Public Sub Prog1()
    Dim x As Single, y As Single, z As Single
    x = InputBox("Введите x")
    y = InputBox("Введите y")
    If y <> 0 Then
        z = x^ 3 / y
        MsgBox "z=" & z
    Else
        MsgBox "y=" & y
    End If
End Sub
```

1.2.5. Для проверки синтаксиса процедуры выполните команду **Debug/Complete VBAProject.**

1.2.6. Посмотрим, как работает условный оператор. Для этого воспользуемся созданием точки прерывания.

Длина некоторых процедур может достигать сотен строк. Чтобы добраться к проблемному участку кода, совсем не обязательно пошагово выполнять все предшествующие ему строки. Достаточно создать точку прерывания, и выполнение макроса будет остановлено на ее границе.

Чтобы создать точку прерывания, щелкните на полосе слева от строки кода, перед выполнением которой необходимо сделать остановку. Строка кода будет

выделена красно-коричневым цветом, а слева от нее появится такого же цвета маркер (см. рис. 33).

1.2.7. Запустите процедуру, нажав клавишу F5.

Последовательно появятся диалоговые окна для ввода значений переменных *x* и *y* (например, x=3, y=5). Выполнение процедуры остановится на границе точки прерывания, а соответствующая строка кода будет выделена желтым цветом (см. рис.33).

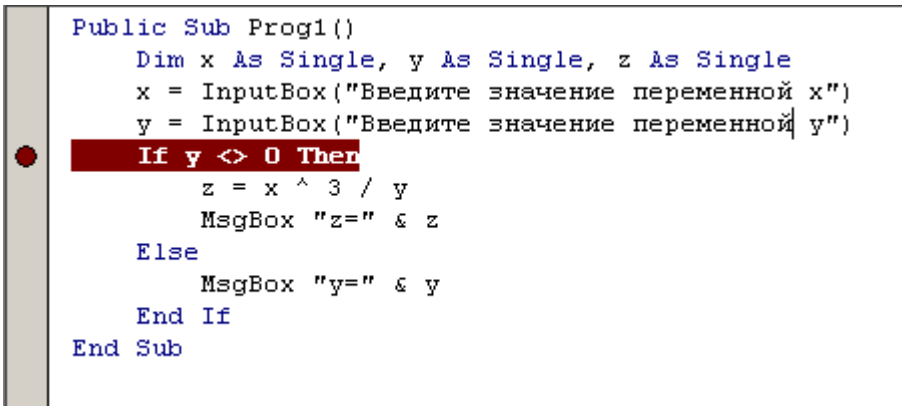


Рисунок 33. Красно-коричневый маркер слева от строки кода свидетельствует о наличии точки прерывания.

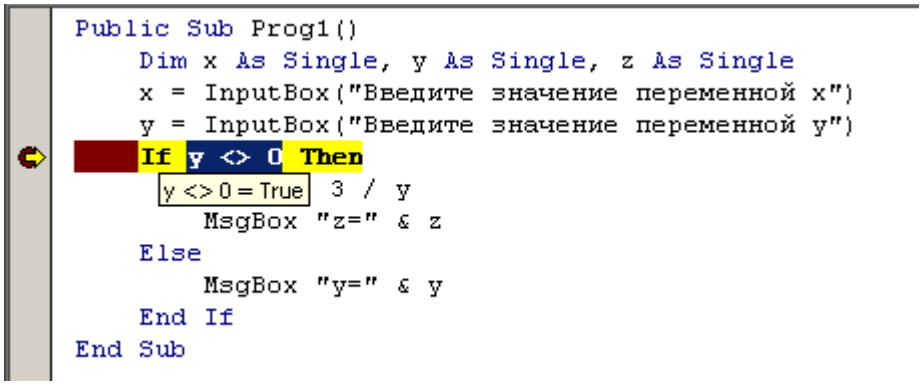


Рисунок 34. Вычисление значения с помощью указателя мыши

1.2.8. Узнаем значение логического выражения в условном операторе при уже заданном значении *y*. Для этого выделите логическое выражение и задержите в таком положении указатель мыши (см. рис.34). Значение логического выражения равно True, следовательно, управление будет передано блоку операторов

```
z = x^ 3 / y
MsgBox "z=" & z
```

Для завершения работы программы последовательно нажимайте F8.

1.2.9. Вновь запустите программу и введите значение *y* равное 0. Используя точку прерывания, проследите за работой условного оператора. Уберите точку прерывания, щелкнув на маркере точки прерывания.

2. Проверьте как работает программа представленная в примере 16.1. Обратите внимание, что управление в условном операторе зависит от того, какая будет

нажата кнопка при выводе окна сообщений. Используйте точку прерывания для просмотра работы условного оператора.

3. Выполните самостоятельно задание №3 контрольной работы.

Лабораторная работа № 6

Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. ***Программирование алгоритмов разветвляющей структуры. Создание функции пользователя.***

Задача: Создать функцию пользователя для определения стоимости заказа клиента фирмы «Марс», торгующей печатной продукцией, с учетом скидки. Если продается от 100 до 200 экземпляров книги, то скидка от ее отпускной цены составляет 7%, если продается от 201 до 300 экземпляров, то скидка составляет 10%, а если свыше 300 экземпляров —15 %. Кроме того, для постоянных клиентов предусмотрена дополнительная скидка 5%.

1. **Формализация задачи.** Определим исходные и выходные данные.

Исходные данные: ЦенаОднойКниги (ZN) —цена одной книги, Количество (K) — количество, SK — признак постоянного клиента (если нет скидки значение 0, в противном случае 1).

Промежуточные данные: STB — стоимость партии книг без учета скидки.

Выходные данные: Стоимость(ST) – Стоимость партии книг с учетом скидки.

2. **Разработка алгоритма.** Изобразим алгоритм решения задачи с помощью блок-схемы представленной на рисунке 35. Как видно из блок-схемы управляющие структуры ветвления вложены друг в друга. Для реализации таких алгоритмов можно применить многострочный условный оператор или оператор выбора.

3. **Реализация алгоритма на ЭВМ.** Создадим пользовательскую функцию.

   3.1 Открыть новую рабочую книгу.

   3.2 Войти в редактор VBA (Alt+F11).

   3.3 В редакторе VBA выполнить команду **Insert/Module**.

   3.4 Выполнить команду **Insert/Procedure**. В появившемся диалоговом окне указать название процедуры Стоимость и выберите вид процедуры Function.

Ранее мы все программы оформляли в виде процедур. При создании пользовательской функции подпрограмма записывается в виде процедуры-функции, т.к. функция имеет только один выходной параметр, имя которого совпадает с именем процедуры-функции, что и требуется для поставленной задачи. По окончании работы подпрограммы пользователь должен получить одно значение — стоимость покупки. В скобках процедуры-функции указываются входные переменные и их тип, если тип переменных опущен, то переменные имею тип по умолчанию Variant.

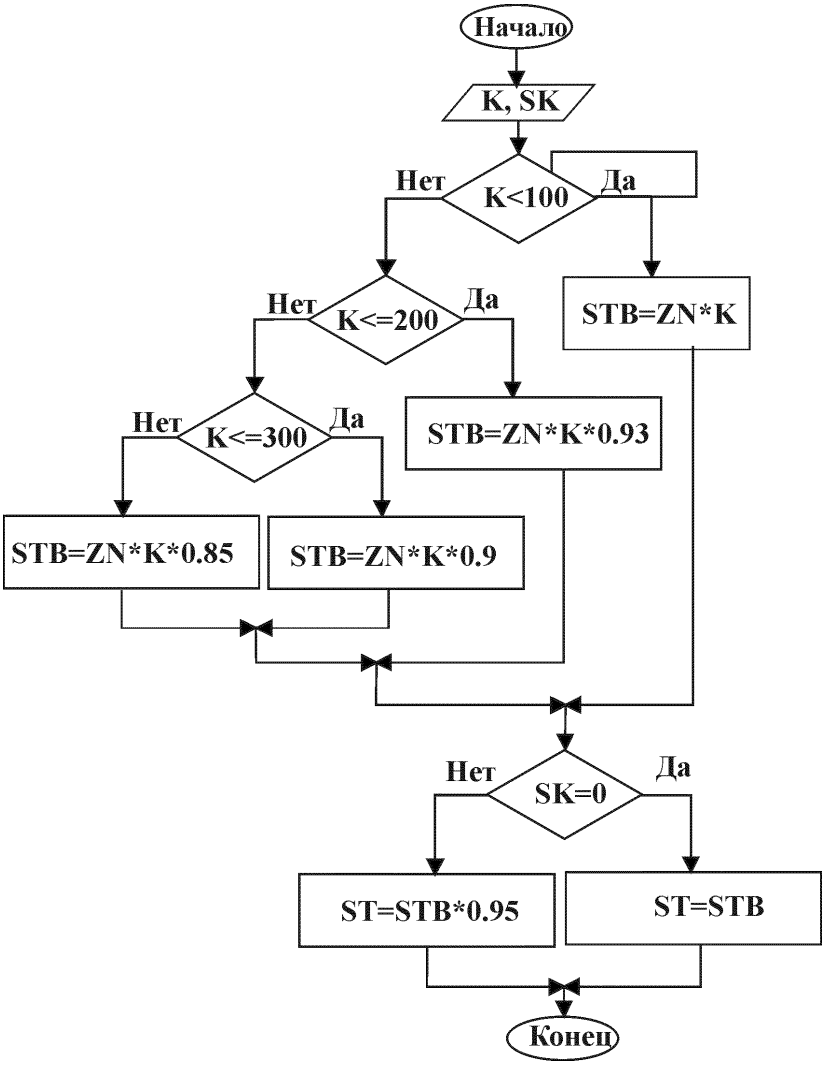


Рисунок 35. Блок-схема решения задачи с помощью условного оператора

3.5 Поместите следующий код программы в модуль.

```
Public Function Стоимость(ЦенаОднойКниги, Количество, Скидка)
    If Количество < 100 Then
      STB = ЦенаОднойКниги * Количество
    ElseIf Количество <= 200 Then
      STB = ЦенаОднойКниги * Количество * 0.93
    ElseIf Количество <= 300 Then
      STB = ЦенаОднойКниги * Количество * 0.9
    Else
      STB = ЦенаОднойКниги * Количество * 0.85
    End If
    If Скидка = 0 Then
      Стоимость = STB
    Else
```

```
        Стоимость = STB * 0.95
    End If
End Function
```

4. Выполнить команду **Debug/Compile VBA Project**.

5. Вернуться в рабочий лист Excel.

Настроить электронную таблицу как показано на рисунке 36.

Использовать для вызова функции Стоимость Мастер функций. Применение русскоязычных имен делает текст написанной программы ясным и прозрачным. Названия всех параметров функции Стоимость в окне мастера функций выводятся также на русском. Ясная структура диалогового окна позволяет применять функцию Стоимость любому пользователю.

D2 =Стоимость(A2;B2;C2)

	A	B	C	D	E
1	Цена одной книги	Количество	Скидка	Стоимость	
2	102	120	1	10814,04	
3					

Рисунок 36. Результат работы программы на рабочем листе

6. Изучите оператор выбора (пункт 16) и создайте программу для реализации в среде VBA процедуры представленной в примере 16.1. Выполните реализацию алгоритма решения поставленной в начале этой лабораторной работы задачи с помощью оператора выбора.

7. Выполните задание №4 контрольной работы.

Лабораторная работа №7

Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. ***Программирование алгоритмов циклической структуры. Циклы с параметром***

Циклическим алгоритмом называется алгоритм, предусматривающий многократное повторение одной и той же группы действий над новыми данными.
Эта группа действий называется *телом цикла*, а ее однократное повторение называется *итерацией*. Переменная, в которой хранится количество выполненных итераций, называется *счетчиком*.
Цикл называется *арифметическим или циклом с параметром*, если число повторений цикла известно заранее или может быть вычислено. Переменную, изменяющуюся в цикле, называют параметром цикла. Блок-схема цикла с параметром представлена на рис. 37.

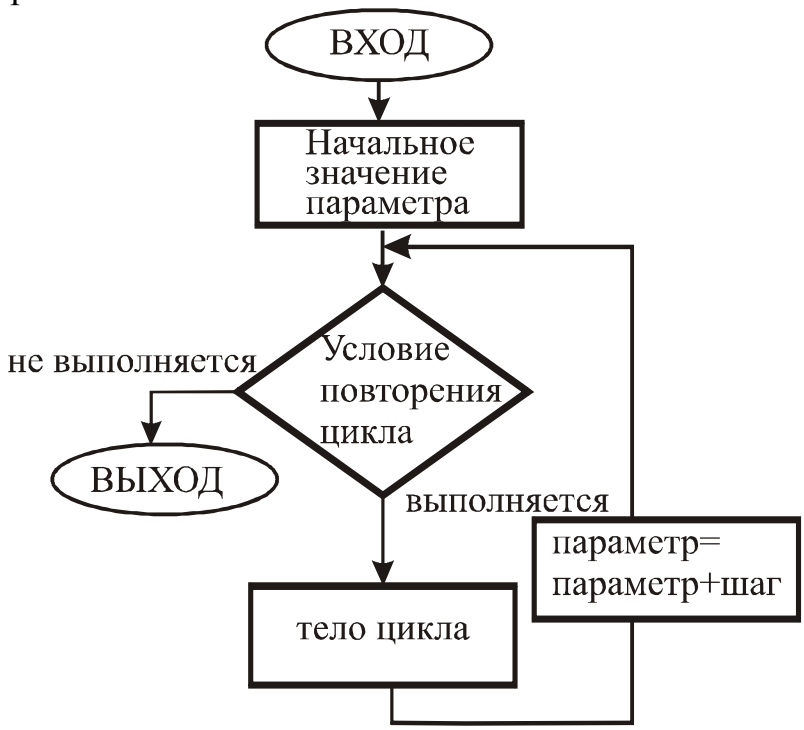


Рисунок 37. Блок-схема базовой структуры Цикл с параметром

Задача 1.

Дано число A — цена 1 кг. конфет. Вывести стоимость 0,1, 0,2, …, 1 кг. конфет.

1.1. Разработка алгоритма.

Способ 1

Пусть переменная x — вес конфет в кг., переменная c — стоимость конфет.

Тогда,
для x=0.1 – $c=A*x$

для x=0.2 – c=A*x
и т.д.
Можно заметить, что каждое новое значение переменной x получается из предыдущего добавлением числа 0.1.

Запишем в ячейку x ноль. Тогда 10 раз нужно выполнить следующие действия
x=x+0.1
*c=A*x*
Вывести x и c

Блок-схема алгоритма решения данной задачи представлена на рис.38. Переменная *i* является параметром цикла.

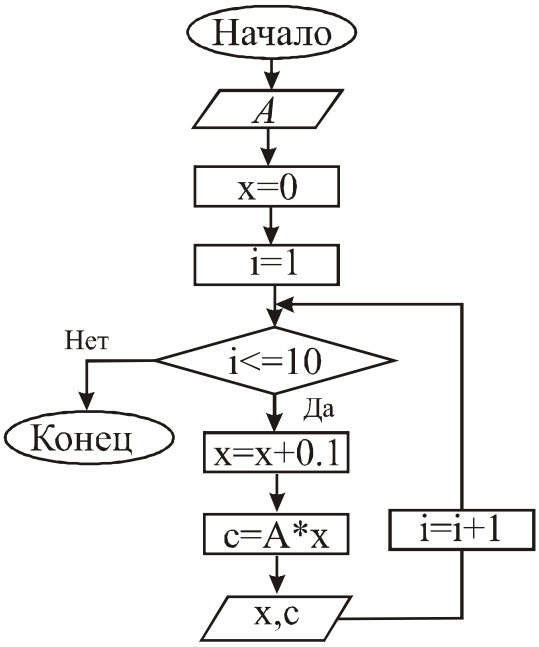


Рисунок 38. Блок-схема решения задачи с циклическим оператором

Алгоритмы циклической структуры с параметром реализуются в среде VBA с помощью оператора цикла For … Next (см пункт18).

Программная реализация. Способ 1

Процедура, реализующая решение данной задачи представлена ниже.

```
Public Sub PrFor1()
    Dim a As Single, i As Integer, x As Single, c As
Single
    Cells(1, 1) = "Вес": Cells(1, 2) = "Стоимость"
    a = InputBox("Введите цену кг. конфет")
    x = 0
    For i = 1 To 10
        x = x + 0.1
        c = a * x
        Cells(i + 1, 1) = x: Cells(i + 1, 2) = c
```

```
        Next
    End Sub
```

Разберем как будет работать данная процедура построчно.

`Cells(1, 1) = " Вес ":` `Cells(1, 2) = " Стоимость "`	На активном рабочем листе в ячейку A1 вводится текст «Вес», в B1 — «Стоимость»
`a = InputBox("Введите цену кг. конфет ")`	Значение ячейки a задается с помощью диалогового окна, например, a=40
`x = 0`	Ячейка x получает значение равное 0
`For i = 1 To 10`	Начало цикла переменная i получает значение равное 1, если i не больше 10, то выполняется тело цикла. 1<10, следовательно, выполняется тело цикла
`x = x + 0.1`	Переменная x получает значение 0.1
`c = a * x`	Переменная c получает значение 4
`Cells(i + 1, 1) = x:` `Cells(i + 1, 2) = c`	В ячейку, расположенную на второй строке первого столбца (A2) заносится значение переменной x, в ячейку второй строки второго столбца — значение переменной c
`Next`	Параметр цикла i увеличивается на шаг. в нашем случае на 1, поскольку в начале цикла значение шага не указано (i=2). Управление передается в начало цикла.
`For i = 1 To 10`	Если i не больше 10, то выполняется тело цикла. 2<10, следовательно, выполняется тело цикла.
И т.д. Будет выполнено всего десять итераций. Как только переменная I получит значение равное 11 управление будет передано оператору, стоящему после оператора цикла. В нашем случае на конец процедуры.	

Способ 2

Составим таблицу зависимости x от i.

i	1	2	3	4	5	…
x	0.1	0.2	0.3	0.4	0.5	…

Нетрудно заметить, что *x* выражается через *i* по следующей формуле: *x=0.1*i*. Кроме изменения формулы вычисления значений переменной *x*, в новой процедуре обойдемся без переменной *c*.

Программная реализация. Способ 2

```
Public Sub PrFor2()
    Dim a As Single, I As Integer, x As Single
    Cells(1, 1) = "Вес": Cells(1, 2) = "Стоимость"
    a = InputBox("Введите цену кг. конфет ðà")
        For I = 1 To 10
        x = I * 0.1
```

```
            Cells(I + 1, 1) = x: Cells(I + 1, 2) = a * x
        Next
    End Sub
```

1.2. Создайте в окне редактирования кода объекта Лист1 новой рабочей книги процедуры PrFor1, PrFor2.

1.3. Создайте точки прерывания в начале оператора цикла. Для лучшего понимания работы оператора цикла выполните итерации пошагово с помощью нажатия кнопки F8.

1.4. Исправьте программный код процедур PrFor1, PrFor2, так чтобы на рабочий лист выводилась и цена товара, и его наименование.

Задача 2.

В ячейки рабочего листа вывести N первых чисел Фибоначчи.

2.1. Разработка алгоритма

Числа Фибоначчи определяются с.о.: первое и второе число равны 1, а каждое следующее равно сумме двух предыдущих. В результате получается последовательность 1, 1, 2, 3, 5, 8, 13, 21, 34, …, задаваемая следующим соотношением:

$$x_1 = 1; \quad x_2 = 1;$$
$$x_{i+2} = x_i + x_{i+1}, \quad i = 1,2,3,...$$

Такое соотношение называется рекуррентным, а последовательность рекуррентной (если дано начальное число последовательности, а каждое следующее число последовательности выражается по формуле $x_{i+1} = f(x_i), \quad i = 1,2,...$, где $f(x)$ — некоторая функция, то такая последовательность называется *рекуррентной*, а сама формула — *рекуррентным соотношением*).

Способ 1

Будем хранить последовательность в одномерном массиве a и его элементы передавать ячейкам рабочего листа.

Понятно, что длина массива будет зависеть от числа n, которое пользователь будет задавать с помощью окна ввода.

Разберем решение задачи по шагам.

1 шаг. *a(1)=1*

2 шаг. *a(2)=1*

3 шаг. *a(3)=a(1)+a(2)=1+1=2*

4 шаг. *a(4)=a(2)+a(3)=1+2=3*

…

i шаг. *a(i)=a(i-2)+a(i-1)*

…

n шаг. *a(n)=a(n-2)+a(n-1)*

Одна и та же формула повторяется от третьего, до *N*-го шага. Следовательно, для вычисления последовательности Фибоначчи воспользуемся циклом. Параметр цикла *i* изменяется от 3 до n с шагом равным 1.

```
a(1) = 1: a(2) = 1
For i = 3 To n
    a(i) = a(i - 2) + a(i - 1)
Next
```

Для лучшего понимания алгоритма составим таблицу трассировки.

Пусть n=5.

Оператор		i	a(1)	a(2)	a(3)	a(4)	a(5)
`a(1) = 1`		–	1	–	–	–	–
`a(2) = 1`		–	1	1	–	–	–
`For i = 3 To 5`	1-я итерация	3	1	1	–	–	–
`a(i) = a(i - 2) + a(i - 1)`		3	1	1	2	–	–
`Next`		4	1	1	2	–	–
`For i = 3 To 5`	2-я итерация	4	1	1	2	–	–
`a(i) = a(i - 2) + a(i - 1)`		4	1	1	2	3	–
`Next`		5	1	1	2	3	–
`For i = 3 To 5`	3-я итерация	5	1	1	2	3	–
`a(i) = a(i - 2) + a(i - 1)`		5	1	1	2	3	5
`Next`		6	1	1	2	3	5

Запишем полностью процедуру для решения поставленной задачи.

```
Public Sub PrFor3()
    Dim a() As Integer, n As Integer
    n = InputBox("Введите требуемое количество первых
    чисел Фибоначчи")
    ReDim a(n) As Integer
    a(1) = 1: a(2) = 1
    Cells(1, 1) = a(1): Cells(2, 1) = a(2)
    For i = 3 To n
        a(i) = a(i - 2) + a(i - 1)
        Cells(i, 1) = a(i)
    Next
End Sub
```

Обратите внимание на объявление массива *a*. Массив *a* объявляется как динамический.

Данная процедура имеет недостаток. Поскольку числа Фибоначчи выводятся в первый столбец рабочего листа, и процедура может вызываться несколько раз, то целесообразным является прежде чем заполнять ячейки рабочего листа очистить первый столбец (новая строка программы выделена другим шрифтом).

Т.о. процедура имеет вид:

```
Public Sub PrFor3()
    Dim a() As Integer, n As Integer
    Range("A:A").Clear
    n = InputBox("Введите требуемое количество первых
    чисел Фибоначчи ")
    ReDim a(n) As Integer
    a(1) = 1: a(2) = 1
    Cells(1, 1) = a(1): Cells(2, 1) = a(2)
    For i = 3 To n
        a(i) = a(i - 2) + a(i - 1)
        Cells(i, 1) = a(i)
    Next
End Sub
```

Способ 2

Заметим, что использование массива для хранения последовательности Фибоначчи не является необходимым условием для решения поставленной задачи.

Разработаем алгоритм, в котором используется три переменные: переменная x — для текущего числа и переменные a и b — для двух предыдущих чисел.

Вначале a=1 и b=1.

На первом шаге мы вычислим x=a+b и выведем значение x. Теперь нужно изменить значения a и b, чтобы на шаге 2 с помощью той же формулы x=a+b вычислить следующее значение x. Для этого необходимо выполнить присваивания

a=b : b=x

Повторяя описанные действия в цикле, мы будем последовательно находить все числа Фибоначчи.

```
a = 1: b = 1
For i = 3 To n
    x = a + b
    Cells(i, 1) = x
    a = b
    b = x
Next
```

Составим таблицу трассировки для случая n=5.

Оператор		i	a	b	x
`a = 1`		–	1	–	–
`b = 1`		–	1	1	–
`For i = 3 To 5`	1-я итерация	3	1	1	–
`x = a + b`		3	1	1	2
`a = b`		3	1	1	2
`b = x`		3	1	2	2
`Next`		4	1	2	2
`For i = 3 To 5`	2-я итерация	4	1	2	2
`x = a + b`		4	1	2	3
`a = b`		4	2	2	3
`b = x`		4	2	3	3
`Next`		5	2	3	3
`For i = 3 To 5`	3-я итерация	5	2	3	3
`x = a + b`		5	2	3	5
`a = b`		5	3	3	5
`b = x`		5	3	5	5
`Next`		6	3	5	5

2.2. Реализуем решение задачи в среде VBA.

В окно редактирования кода объекта Лист1 новой рабочей книги введите процедуры PrFor3, PrFor4. Обратите внимание, что в цикле появился оператор Cells(i,1)=x. Он предназначен для вывода чисел Фибоначчи на рабочий лист.

```
Public Sub PrFor4()
    Dim n As Integer, x As Integer
    n = InputBox("Введите число чисел Фибоначчи")
    a = 1: b = 1
    Cells(1, 1) = a: Cells(2, 1) = b
    For i = 3 To n
        x = a + b
        Cells(i, 1) = x
        a = b
        b = x
    Next
End Sub
```

2.3. Перед выводом чисел Фибоначчи в первый столбец рабочего листа его нужно очистить. Вставьте необходимый оператор в процедуру PrFor4 самостоятельно.

2.4. Создайте точки прерывания в начале оператора цикла. Для лучшего понимания работы оператора цикла выполните итерации пошагово с помощью нажатия кнопки F8.

Задача 3.

Последовательно с клавиатуры ввести *n* целых чисел и найти их сумму.

3.1. Разработка алгоритма.

Количество вводимых чисел будем хранить в переменной *n*. Введенные по очереди с клавиатуры числа будем хранить в переменной *x*. Поскольку числа будут вводится *n* раз целесообразно использовать оператор цикла с параметром.

```
n = InputBox("Введите количество чисел")
For i = 1 To n
    x = InputBox("Введите число")
Next
```

Будем выводить введенные числа в ячейки рабочего листа.

```
n = InputBox("Введите количество чисел ")
For i = 1 To n
    x = InputBox("Введите число ")
    Cells(i, 1) = x
Next
```

Каждое введенное число нужно присовокупить к общей сумме, которую будем хранить в ячейке *s*. Прибавлять к текущей сумме очередное число будем по формуле: *s=s+x*. Но для того чтобы прибавить к *s* первое число необходимо в ячейку *s* положить ноль, т.е. обнулить ячейку *s*.

Замечание	Аналогично накоплению суммы накапливается произведение с той лишь разницей, что для его накопления используется формула *P=P*x*, а начальное значение ячейки *P* должно быть равно единице.

Итак, добавляем в тело цикла формулу для накопления суммы.

```
n = InputBox("Введите количество чисел ")
s = 0
For i = 1 To n
    x = InputBox("Введите число ")
    Cells(i, 1) = x
    s = s + x
```

```
Next
```

3.2. Реализуем решение задачи в среде VBA.

В окно редактирования кода объекта Лист1 новой рабочей книги введите текст процедуры PrFor5. Внимательно разберите каждую строчку программного кода.

```
Public Sub PrFor5()
    Dim n As Integer, i As Integer
    Dim x As Integer, s As Integer
    Range("A:B").Clear
    n = InputBox("Введите количество чисел ")
    s = 0
    For i = 1 To n
        x = InputBox("Введите число")
        Cells(i, 1) = x
        s = s + x
    Next
    Cells(n + 1, 1) = "Итого": Cells(n + 1, 2) = s
End Sub
```

3.3. Запустите процедуру PrFor5.

3.4. Создайте новую процедуру PrFor6, которая бы позволяла вычислять сумму *n* **действительных** чисел, и результат выводился бы в простое окно сообщений.

3.5. Создайте новую процедуру PrFor61, которая бы позволяла вычислять произведение *n* **действительных** чисел и результат выводился бы в простое окно сообщений.

Задача 4.

Последовательно с клавиатуры ввести *n* чисел и найти среднее арифметическое положительных чисел.

4.1. Разработка алгоритма.

Как и в предыдущей задаче, количество вводимых чисел будем хранить в переменной *n*, введенные по очереди с клавиатуры числа будем хранить в переменной *x*. Сумму положительных чисел будем хранить в ячейке *s*. Но для вычисления среднего арифметического необходимо знать количество положительных чисел, используем для этого ячейку *k*.

Можно записать типовой алгоритм для подсчета количества требуемых элементов.

```
k=0
Цикл
начало_цикла
```

```
    ввести x
    если x удовлетворяет определенному условию то k=k+1
конец_цикла
```

Для решения поставленной *n* раз предстоит повторять следующие действия:

1. ввести число *x*;
2. если *x>0* , то следует увеличить содержимое ячейки s на *x*, а содержимое ячейки *k* на 1.

Следовательно, для реализации алгоритма нужен оператор цикла с параметром. Для того чтобы прибавить к *s* и *k* первое число необходимо ячейки *s* и *k* обнулить перед выполнением оператора цикла.

```
s = 0: k = 0
For i = 1 To n
    x = InputBox("Введите число")
    Cells(i, 1) = x
    If x > 0 Then
        s = s + x
        k = k + 1
    End If
Next
```

Прежде чем поменять значение ячейки *s* на среднее арифметическое (s = s / k) необходимо проверить есть ли во введенной последовательности чисел положительные, т.е. найти значение логического выражения $k \neq 0$.

В зависимости от значения логического выражения на рабочий лист выводятся соответствующие строковые константы.

```
If k <> 0 Then
 s = s / k
 Cells(n + 1, 1) = "Среднее арифметическое = " & s
Else
 Cells(n + 1, 1) = "На ноль делить нельзя"
End If
```

4.2. Реализуем решение задачи в среде VBA.

В окно редактирования кода объекта Лист1 введите текст процедуры PrFor7. Внимательно разберите каждую строчку программного кода.

```
Public Sub PrFor7()
    Dim n As Integer, i As Integer
    Dim x As Single, s As Single, k As Integer
    Range("A:B").Clear
    n = InputBox("Введите количество чисел")
    s = 0: k = 0
```

```
    For i = 1 To n
        x = InputBox("Введите число x")
        Cells(i, 1) = x
        If x > 0 Then
            s = s + x
            k = k + 1
        End If
    Next
    If k <> 0 Then
     s = s / k
     Cells(n + 1, 1) = "Среднее арифметическое =" & s
    Else
     Cells(n + 1, 1) = "Числа не введены"
    End If
End Sub
```

4.3. Протестируйте работу программы (проверьте правильность решения поставленной задачи). Несколько раз запустите программу с различными наборами данных (только положительные числа, только отрицательные числа, положительные и отрицательные числа).

5. Разберите программную реализацию решения задачи 18.1., в которой необходимо построить таблицу значений функции

$$y=\begin{cases}\dfrac{\ln(1+x)}{x-6}, \text{если } x \ge 1\\ \dfrac{e^{x}+e^{-x}}{2}, \text{если } x < 1\end{cases}, x\in[-2;5], \text{ шаг } 0{,}5.$$

Блок-схема решения задачи представлена на рисунке 39. обратите внимание, в данном алгоритме параметром является переменная *x*, тело цикла имеет один оператор — полное ветвление. Причем параметр изменяется с шагом отличным от единицы.

5.1. Программная реализация предложенного алгоритма представлена в примере 18.1. Отладьте и запустите представленную процедуру в редакторе VBA. Используйте пошаговое выполнение процедуры для создания таблицы трассировки для *x* от -2 до 0.

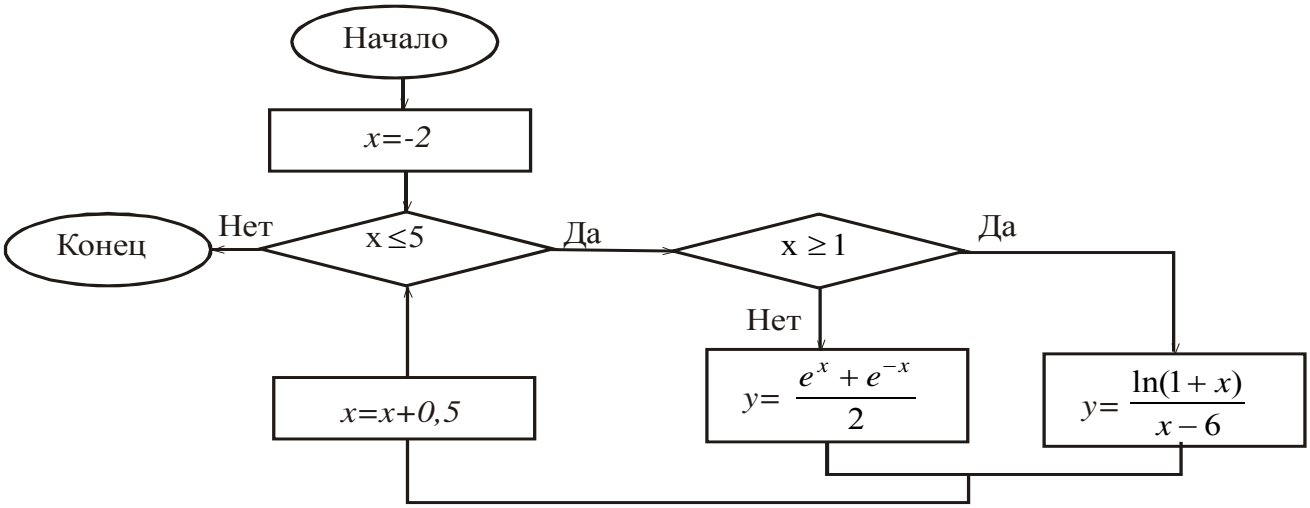


Рисунок 39. Блок-схема вычисления значений функции на отрезке

5.2. Переменная *x* пробегает значения от -2 до 5, увеличиваясь на 0,5. нетрудно подсчитать количество итераций цикла: (5-(-2)/0,5+1, т.е. 15.

Составим таблицу зависимости *x* от *i*, где *i* — счетчик, пробегающий значения от 1 до 15 с шагом равным 1.

i	1	2	3	4	5	…
x	-2	-1,5	-1	-0,5	0	…

Заметим, что x выражается через *i* по следующей формуле x=-2,5+0,5*i.

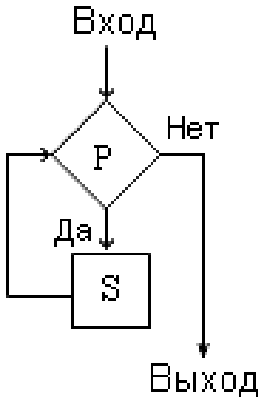
Напишите новую процедуру для решения поставленной задачи, в которой используется полученная формула, *i* является параметром цикла. Запустите новую процедуру. Сравните результаты работы двух процедур, реализующих решение поставленной задачи.

Лабораторная работа №8

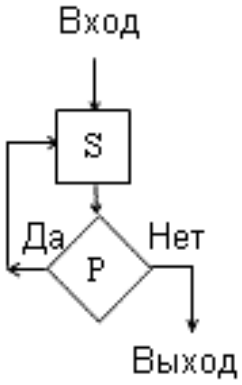
Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. ***Программирование алгоритмов циклической структуры. Циклы с неизвестным числом повторений***

Цикл, для которого нельзя указать числа повторений, и проверка окончания цикла происходит по достижению нужного условия, называется **итерационным**.

Цикл с предусловием
(может не выполняться ни разу)



Цикл с постусловием
(выполняется хотя бы раз)



Цикл с предусловием начинается с проверки условия P выхода из цикла. Это логическое выражение. Если оно истинно, то выполняются те действия, которые должны повторяться. В противном случае, если логическое выражение P ложно, то этот цикл прекращает свои действия.

Цикл с постусловием функционирует иначе. Сначала выполняется один раз те действия, которые подлежат повторению, затем проверяется логическое выражение P, определяющее условие выхода из цикла. Проверка его осуществляется тоже по-другому. Если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае - происходит повторение действий, указанных в цикле. Повторяющиеся действия в цикле называются «телом цикла».

Алгоритмы циклической структуры с неизвестным числом повторений реализуются в среде VBA, например, с помощью оператора цикла Do (см.пр.19). Как указано в пункте 19 существует несколько разновидностей цикла Do: цикл Do с предусловием и цикл Do с постусловием.

Задача 1.

Вычислить сумму всех четных двузначных чисел.

1.1. Разработка алгоритма

Понятно что, для решения поставленной задачи мы можем воспользоваться циклом For…Next. Но с помощью цикла Do можно решать те же задачи, что и с использованием цикла For. Для этого требуется завести переменную выступающую в роли параметра цикла For. В конце каждой итерации цикла эту переменную следует увеличивать или уменьшать.

Решение с помощью оператора цикла For.

Нетрудно посчитать, что количество четных двузначных чисел равно 45. В качестве параметра цикла будем использовать переменную *i*, четное двузначное число будем хранить в переменной *d*, сумму в переменной – *s*. Сначала в ячейку *d* положим число равное 12, далее в конце каждой итерации значение ячейки *d* будем увеличивать на 2.

Получаем следующее решение задачи.

```
s = 0: d = 10
For i = 1 To 45
    s = s + d
    d = d + 2
Next
```

В качестве параметра цикла можно использовать и непосредственно переменную d. Тогда решение задачи имеет следующий вид

```
s = 0
For d = 10 To 98 Step 2
    s = s + d
Next
```

Решение с помощью оператора цикла Do.

Рассмотрим вариант, когда параметром цикла будет выступать переменная *d*. Цикл будет повторяться до тех пор, пока *d* не будет превосходить число 98, т.е. будет истинным условие d < = 98 или ложным условие d > 98

Решение задачи с помощью цикла с предусловием.

```
s = 0: d = 10
Do While d <= 98
    s = s + d
    d = d + 2
Loop
```

Решение задачи с помощью цикла с постусловием.

```
s = 0: d = 10
Do
    s = s + d
    d = d + 2
Loop Until d > 98
```

1.2. Создайте четыре процедуры для решения поставленной задачи, запустите процедуры и убедитесь, что в результате выполнения процедур будет получен одинаковый результат.

Задача 2.

Дано целое число n(>1). вывести наименьшее из целых чисел k, для которых сумма 1 + 2 + 3 +…+ k , будет больше или равна n, и саму эту сумму.

2.1. Разработка алгоритма.

Будем накапливать сумму *s*, пока она остается меньше *n*. Для хранения очередного слагаемого заведем переменную *i* и будем увеличивать ее в цикле на 1 перед добавлением к сумме *s*.

```
s = 0: i = 0
Do While s < n
    i = i + 1
    s = s + i
Loop
k = i
```

Обратите внимание на то, что переменная *i* вначале увеличивается, а потом добавляется к сумме. Поэтому до цикла следует положить *i=0*, а после завершения цикла в переменной *i* будет содержаться требуемое значение *k*.

Для лучшего понимания работы алгоритма составим таблицу трассировки оператора цикла для *n=5*.

Оператор		i	s	Условие s<n
Do While s < n	1-я итерация	0	0	true
i = i + 1		1	0	true
s = s + i		1	1	true
Do While s < n	2-я итерация	1	1	true
i = i + 1		2	1	true
s = s + i		2	3	true
Do While s < n	3-я итерация	2	3	true
i = i + 1		3	3	true
s = s + i		3	6	true
Do While s < n		3	6	false

Итак, цикл останавливает работу как только условие становится ложным, далее переменной *k* присваивается значение переменной *i* , в нашем случае *i=3*.

Теперь запишем решение поставленной задачи с помощью цикла Do с постусловием.

```
s = 0: i = 0
Do
    i = i + 1
```

```
    s = s + i
Loop Until s >= n
k = i
```

Составим таблицу трассировки для оператора цикла с постусловием, n=5.

Цикл останавливает работу, как только условие становится истинным, далее переменной *k* также присваивается значение переменной *i* , в нашем случае *i=3*.

Оператор		i	s	Условие s>=n
`i = i + 1`	1-я итерация	1	0	–
`s = s + i`		1	1	–
`Loop Until s >= n`		1	1	false
`i = i + 1`	2-я итерация	2	1	false
`s = s + i`		2	3	false
`Loop Until s >= n`		2	3	false
`i = i + 1`	3-я итерация	3	3	false
`s = s + i`		3	6	false
`Loop Until s >= n`		3	6	true

2.2. Реализуйте решение поставленной задачи в среде VBA. Создайте две процедуры для решения поставленной задачи (с циклом с предусловием и постусловием), сравните результаты работы двух процедур.

2.3. Решите следующую задачу: дано целое число n(>1). Вывести наибольшее из целых чисел k, для которых сумма 1+2+…+k будет меньше или равна n, и саму эту сумму.

3. Реализуйте в среде VBA решение задач, приведенных в примерах 19.1 и 19.2.

Лабораторная работа №9

Тема: Основа разработки алгоритмов и реализация алгоритмов в среде программирования VBA. *Приемы обработки одномерных и двумерных числовых массивов*

Под структурой данных типа массив понимают однородную структуру однотипных данных, одновременно хранящихся в последовательных ячейках оперативной памяти. Эта структура должна иметь имя и определять заданное количество данных (элементов). Однотипность данных определяет возможность использования циклических алгоритмов для обработки всех элементов массива. Количество итераций цикла определяется количеством элементов массива. Одновременное хранение в памяти всех элементов массива позволяет решать большой набор задач, таких как, поиск элементов, упорядочение и изменение порядка следования элементов. Доступ к любому элементу массива осуществляется по его номеру (индексу). Поэтому для обращения к элементу массива используют имя массива (номер элемента), например, A(5) (см. пункт 7).

Задачи на обработку одномерных и двумерных числовых массивов решаются подобно разобранным в предыдущих лабораторных работах. Разница заключается лишь в способе хранения информации. Поэтому решение поставленных задач разбирается не так подробно как в предыдущих лабораторных работах.

Задача 1.

Сгенерировать элементы одномерного числового массива A(N) на отрезке [0,1] и сохранить значения массива в ячейках рабочего листа.

1.1. Разработка алгоритма

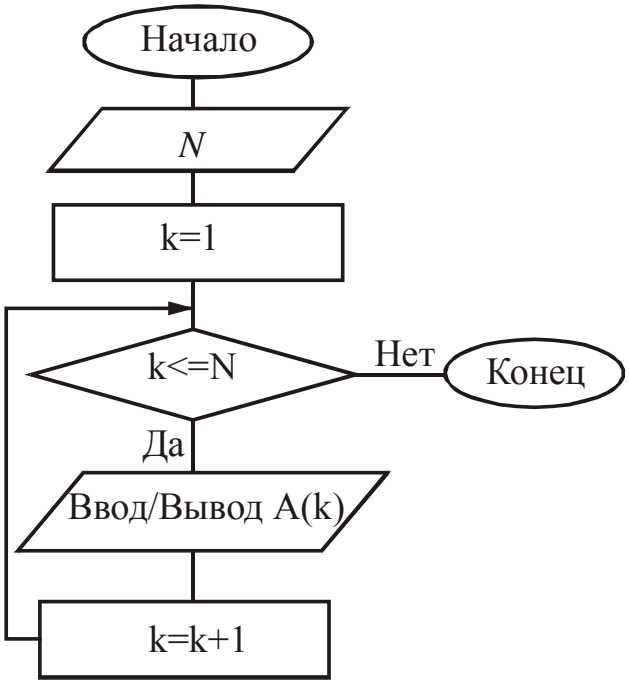


Рисунок 40.

Решение поставленной задачи можно свести к решению задачи ввода/вывода элементов массива. Под вводом можно понимать не только генерацию элементов массива с помощью генератора случайных чисел, но и получение элементов массива с помощью заданной формулы, ввод элементов массива с помощью окна сообщений, считывание элементов массива с ячеек рабочего листа и т.д.

Алгоритм решения поставленной задачи представлен на рис.40. (*В дальнейшем, при рассмотрении алгоритмов обработки одномерных массивов в целях устранения дублирования алгоритм ввода/вывода элементов массива будем заменять одним блоком, подразумевая,*

что он реализуется по схеме, циклического алгоритма, представленного на рисунке 40).

В циклическом алгоритме представленном на рис. 40 условие выхода из цикла определяется значением параметра цикла *k* (начальное значение *k=1*, конечное значение *k=N*), переменная *k* определяет количество итераций циклического алгоритма ввода элементов массива. На каждом шаге итерации переменная *k* (значение номера элемента массива A) изменяется на 1, то есть происходит переход к новому элементу массива.

Реализуем ввод/вывод одномерного массива в виде процедуры.

```
Sub PrVvod()
   Dim A() as Single
   N=InputBox("Введите размерность массива.")
   ReDim a(1 to N) as Single
   Range("A:A").Clear
   For k=1 To N
     A(k)=Rnd()
     Cells(k, 1) = A(k)
   Next
End Sub
```

Разберем как работает данная процедура.

`Dim A() as Single`	Массив A объявляем как динамический
`N=InputBox("Введите размерность массива.")`	Размерность массива задаем с помощью диалогового окна ввода
`ReDim a(1 to N) as Single`	Переопределяем размерность массива. В оперативной памяти выделяется n+1 ячейка для хранения вещественных чисел
`Range("A:A").Clear`	Поскольку процедура может вызываться многократно, а данные мы помещаем в первый столбец рабочего листа, необходимо очисть столбец A перед сохранением новых данных
`For k=1 To N`	Начало цикла с параметром. Параметр k изменяется от 1 до n с шагом равным 1.
`A(k)=Rnd()`	Генерируем с помощью датчика случайных чисел элементы массива A. Функция Rnd генерирует случайные числа на отрезке [0,1].
`Cells(k, 1) = A(k)`	Выводим значения элементов массива на рабочий лист. Счетчик элементов массива используем как счетчик строк рабочего листа Excel.
`Next`	Конец цикла

1.2. Создайте новую рабочую книгу Массивы.

1.3. Войдите в редактор VBA. Выполните команду **Insert/Module**.

1.4. В окно редактирования кода объекта Module1 вставьте текст процедуры PrVvod. процедура PrVvod должна быть доступна из любого модуля рабочей книги. (Вначале выполните команду **Insert/Procedure**. Проследите за тем чтобы был выделен переключатель Public.)

1.5. Запустите процедуру PrVvod. Обратите внимание, что элементы массива выводятся на рабочий лист, который является активным на момент запуска процедуры. Это не совсем правильно. Изменим процедуруPrVvod. Введем входной параметр имя листа, на который будут выводится элементы массива.

```
Sub PrVvod(nm As String)
   Dim A() As Single
   n = InputBox("Введите размерность массива.")
   ReDim A(1 To n) As Single
   Sheets(nm).Range("A:A").Clear
   For k = 1 To n
     A(k) = Rnd()
     Sheets(nm).Cells(k, 1) = A(k)
   Next
End Sub
```

Поскольку теперь процедура PrVvod имеет входной параметр, ее можно вызвать только из другой процедуры.

1.6. Создайте процедуру Vvod, для вывода элементов массива на рабочий лист Лист1.

```
Public Sub Vvod()
    PrVvod "Лист1"
End Sub
```

1.7. Запустите процедуру Vvod.

Задача 2.

Сгенерировать элементы одномерного числового массива A(N) на отрезке [c,d] и сохранить значения массива в ячейках рабочего листа.

2.1. Разработка алгоритма.

Очевидно, что задача 2 практически ничем не отличается от задачи 1. нужно только лишь поменять функцию для генерации случайных чисел.

Известно, что для получения случайного числа на отрезке [a,b] необходимо применять следующую формулу: $c+(d-c)*r$, где r — случайное число на отрезке [0,1].

Преобразуем процедуру PrVvod: добавим два входных параметра a и b, и изменим формулу в теле цикла для получения элементов массива.

Т.о. получаем следующую процедуру.

```
Sub PrVvod(nm As String, c As Single, d As Single)
   Dim a() As Single
   n = InputBox("Введите размерность массива.")
   ReDim a(1 To n) As Single
   Sheets(nm).Range("A:A").Clear
   For k = 1 To n
     a(k) = c + (d - c) * Rnd()
     Sheets(nm).Cells(k, 1) = a(k)
   Next
End Sub
```

2.2. Пусть a = =-3, b = 4. Получим элементы массива A(15) и запишем их на Лист2. Для этого создайте процедуру Vvod1. обратите внимание, что вместо трех формальных параметров процедуры PrVvod задаются фактические параметры.

```
Public Sub Vvod1()
    PrVvod "Лист2", -3, 4
End Sub
```

2.3. Запустите процедуру Vvod1. Посмотрите на результаты работы процедуры.

Задача 3.

Найти в одномерном целочисленном массиве B из N элементов сумму и количество четных чисел.

3.1. Разработка алгоритма.

Алгоритм решения поставленной задачи в виде блок-схемы представлен на рис. 41. В этом алгоритме переменная *i* — счетчик элементов массива, *s* — сумма четных чисел в массиве, *k*— переменная для подсчета количества четных чисел в массиве.

В теле цикла должно проверяться условие: если очередной элемент массива четный, его значение присоединяется к сумме, хранящейся в ячейке *s*, а значение переменной *k* увеличивается на 1.

Для лучшего понимания задачи создайте таблицу трассировки для *n=5* и массива *B*, такого что *B(1)=0, B(2)=-3, B(3)=5, B(4)=-7, B(5)=3*.

Для того чтобы узнать является ли число четным достаточно найти остаток от деления числа на 2. Если остаток равен 0, то число четное. Для нахождения остатка от деления воспользуемся операцией mod.

```
s = 0:k=1
```

```
For i = 1 To n
    If b(i) mod 2 = 0 Then s = s + b(i): k = k + 1
Next
```

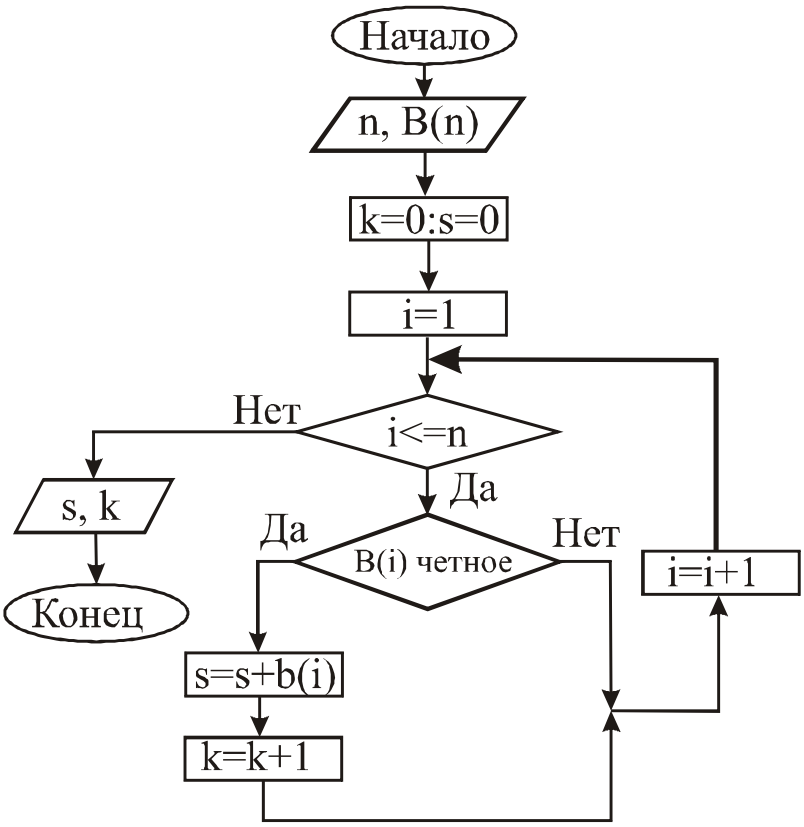


Рисунок 41. Блок-схема решения задачи с циклическим оператором

Для реализации решения поставленной задачи в среде VBA необходимо, прежде всего заполнить массив A. Для заполнения элементов массива A случайными числами ранее была создана процедура PrVvod для ввода элементов массива A. Чтобы сделать процедуру PrVvod универсальной, размерность массива *n* должна стать параметром процедуры PrVvod, , массив, ввод элементов которого будет производить данная процедура, также станет параметром процедуры PrVvod. Кроме того, массив *B* должен быть целочисленным. Если требуется сгенерировать случайные числа из целочисленного интервала, например [1,20], достаточно в процедуре PrVvod определить тип элементов массива как целочисленный.

Т.о. решение поставленной задачи реализуется с помощью двух процедур PrVvod и Arr3.

```
Sub PrVvod(nm As String, c As Single, d As Single, _
n As Integer, a() As Integer)
   Sheets(nm).Range("A:A").Clear
   For k = 1 To n
     a(k) = c + (d - c) * Rnd()
     Sheets(nm).Cells(k, 1) = a(k)
   Next
End Sub
```

```
Public Sub Arr3()
    Dim n As Integer, b() As Integer, i As Integer
    Dim s As Single
    n = InputBox("Введите размерность массива.")
    ReDim b(n) As Integer
    PrVvod "Лист2", 1, 20, n, b
    s = 0: k = 0
    For i = 1 To n
       If b(i) Mod 2 = 0 Then s = s + b(i): k = k + 1
    Next
    MsgBox "Сумма четных чисел массива равно" & s _
    & vbCr & "Их количество равно " & k
End Sub
```

3.2. В окне редактирования кода объекта Module1 измените процедуру PrVvod и создайте процедуру Arr3.

3.3. Запустите процедуру Arr3 несколько раз, меняя размерность массива.

3.4. В рабочей книге Массивы переименуйте рабочий лист «Лист1» в «Задача3». Сделайте изменения в процедуре Arr3, так чтобы и элементы массива и сумма выводились бы в ячейки рабочего листа Задача3.

3.5. В рабочую книгу Массивы вставьте новый рабочий лист Задача3_1 и выполните пример 18.2. Преобразуйте процедуру, приведенную в примере, т.о., чтобы элементы массива выводились в ячейки рабочего листа Задача3_1.

3.6.* Реализуйте решение задачи, приведенное в примере 18.3. в среде VBA.

Задача 4.

Переписать положительные элементы целочисленного массива A(N) в массив B.

4.1. Разработка алгоритма

Схема алгоритма представлена на рисунке 42. Переменная *i* является счетчиком массива *B*. Массив *B* выводится на печать, если в нем есть хотя бы один элемент, в противном случае выводится сообщение о том, что массив *A* не содержит положительных элементов.

Запишем алгоритм решения задачи на языке VBA.

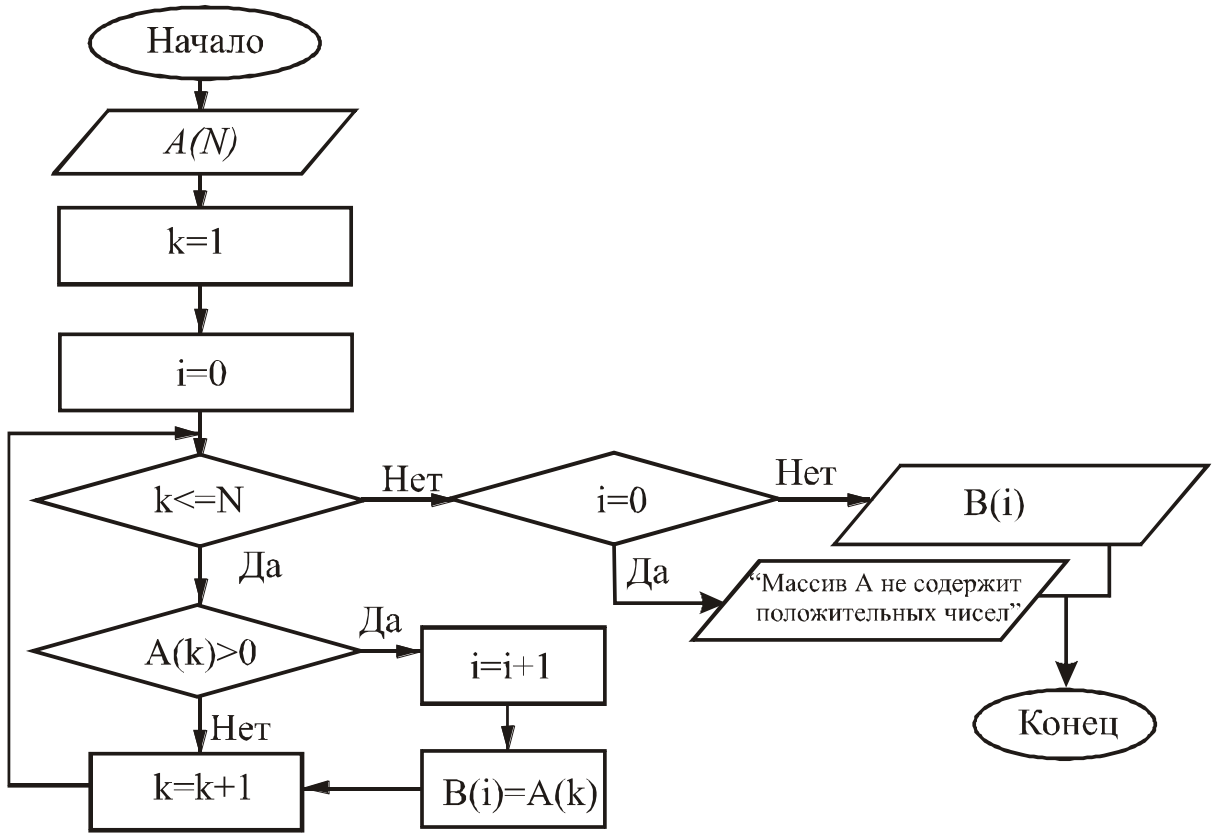


Рисунок 42. Блок-схема решения задачи с циклическим оператором и условным оператором

```
'Обнуляем счетчик элементов массива b, массив b пуст
i = 0
'В цикле проверяем каждый элемент массива a
For k = 1 To n
'Если элемент массива a положительный, то
'счетчик элементов массива bувеличиваем на 1и
'заносим в массив b очередной положительный элемент массива a
    If a(k) > 0 Then i = i + 1: b(i) = a(k)
Next
'Если количество элементов в массиве b не равно нулю, то
'выводим элементы массива b на рабочий лист
'в противном случае выводим  сообщение
If i <> 0 Then
    For k = 1 To i
        Sheets("Лист2").Cells(k, 2) = b(k)
    Next
Else
    MsgBox "Массив А не содержит положительных элементов"
End If
```

4.2. В окне редактирования кода объекта Module1 введите текст процедуры Arr4. Обратите внимание, что результаты работы программы выводятся на Лист2.

```
Public Sub Arr4()
    Dim n As Integer, a() As Integer, i As Integer
    Dim s As Single, b() As Integer
    n = InputBox("Введите размерность массива.")
    ReDim a(n) As Integer
    ReDim b(n) As Integer
    Sheets("Лист2").Range("B:B").Clear
    PrVvod "Лист2", -3, 4, n, a
    i = 0
    For k = 1 To n
        If a(k) > 0 Then i = i + 1: b(i) = a(k)
    Next
    If i <> 0 Then
        For k = 1 To i
            Sheets("Лист2").Cells(k, 2) = b(k)
        Next
    Else
        MsgBox " Массив A не содержит положительных элементов "
    End If
End Sub
```

4.3. Запустите процедуру Arr4. Протестируйте программу. Сгенерируйте в массиве *A* только отрицательные числа. Появляется ли окно сообщений?

4.4. В рабочей книге Массивы.xls переименуйте рабочий лист «Лист2» в «Задача4». Сделайте изменения в процедуре Arr4, так чтобы и элементы массивов *A, B* и сумма выводились бы в ячейки рабочего листа Задача4.

5. **Задача 5.**

В массиве B(16) заменить элементы, имеющие четные индексы на 0.

5.1. Разработка алгоритма.

Для решения поставленной задачи достаточно воспользоваться циклом с параметром. Например, решение задачи можно записать с.о.

```
For i = 2 To 16 Step 2
    b(i) = 0
Next
```

5.2. Для ввода массива воспользуемся процедурой PrVvod. Для решения задачи в окно редактирования кода объекта Module1 введите текст процедуры Arr5.

```
Public Sub Arr5()
    Dim n As Integer, i As Integer, b(16) As Integer
    Sheets("Лист3").Range("B:B").Clear
```

```
    PrVvod "Лист3", -3, 3, 16, b
    For i = 2 To 16 Step 2
        b(i) = 0
        Sheets("Лист3").Cells(i - 1, 2) = b(i - 1)
        Sheets("Лист3").Cells(i, 2) = b(i)
    Next
End Sub
```

Обратите внимание, что массив *B* имеет фиксированный размер, исходный и измененный массивы выводятся в ячейки рабочего листа в соседние столбцы. Поскольку в цикле переменная *i* принимает только четные индексы для вывода элементов массива, имеющих нечетные индексы необходимо записать дополнительный оператор: **`Sheets("Лист3").Cells(i - 1, 2) = b(i - 1).`**

5.3. Запустите процедуру Arr5.

5.4. В рабочей книге Массивы переименуйте рабочий лист «Лист3» в «Задача5». Сделайте изменения в процедуре Arr5, так чтобы элементы исходного и измененного массивов выводились бы в ячейки рабочего листа Задача5.

Задача 6.

Расположить элементы массива A(n) по возрастанию.

6.1. Разработка алгоритма

Среди алгоритмов обработки массивов немаловажную роль играют алгоритмы сортировки. Сортировка (упорядочение) — это изменение порядка следования элементов в зависимости от их значения. Существует множество алгоритмов сортировки.

Рассмотрим сортировку методом прямого обмена. Основную идею можно описать следующим образом. При первом проходе вдоль массива, берется первый элемент массива и сравнивается с остальными. Если встречается элемент с меньшим значением, то элементы меняются местами. Все последующие записи сравниваются с новым значением. В результате элемент с наименьшим значением окажется первым элементом массива. Далее сравнивается второй элемент с остальными и т.д.

Для реализации алгоритма сортировки массива возникает необходимость создания цикла, содержащего в своем теле другой цикл. Такие вложенные друг в друга циклы относятся к структурам *вложенных циклов*. Порядок вложенности циклов, когда в теле внутреннего цикла содержатся другие циклы, может быть достаточно большим. Этот порядок определяется методом, с помощью которого достигается решение поставленной задачи. Так, при обработке одномерных массивов, как правило, удается построить алгоритмическую схему без вложения циклов. Однако в ряде случаев при решении таких задач без вложенных циклов не обойтись.

Отметим, что все вложенные друг в друга циклы, включая наружный, должны иметь счетчики с различными именами. Вне этих циклов счетчики могут быть использованы как обычные переменные или как счетчики других циклов.

Запишем алгоритм сортировки методом прямого обмена на языке VBA в виде отдельной процедуры, параметрами которой являются массив и размерность массива.

```
Sub Sort(a() As integer, n As Integer)
   For i = 1 To n - 1
      For j = i + 1 To n
         If a(i) > a(j) Then
      'Перестановка элементов a(i) и a(j)
            Temp = a(j)
            a(j) = a(i)
            a(i) = Temp
         End If
      Next
   Next
End Sub
```

6.2. В целях лучшего понимания алгоритма сортировки предусмотрим вывод элементов массива после каждой итерации в ячейки рабочего листа.

Для этого добавим операторы в процедуру Sort, они выделены курсивом и назовем новую процедуру SortTras.

```
Sub SortTras(a() As Integer, n As Integer)
   Cells(1, 2) = "i": Cells(1, 3) = "j"
   Cells(1, 5) = "Элементы массива"
   l = 1: fl = False
   For i = 1 To n - 1
      For j = i + 1 To n
         If a(i) > a(j) Then
           Перестановка элементов a(i) и a(j)
           Temp = a(j)
           a(j) = a(i)
           a(i) = Temp
           fl = True
         End If
         st = "   "
         For k = 1 To n
            st = st & a(k) & "     "
         Next
         l = l + 1
         Cells(l, 2) = i
         Cells(l, 3) = j
```

```
            Cells(l, 5) = st
            If fl Then Cells(l, 5).Font.ColorIndex = 8
            fl = False
        Next
        l = l + 1
    Next
End Sub
```

6.3. В рабочей книге Массивы создайте рабочий лист Сортировка.

6.4. В окне редактирования кода объекта Module1создайте три процедуры Sort, SortTras и Arr6. В процедуре Arr6 вызываем процедуру PrVvod для генерации массива *A* и процедуру SortTras для сортировки массива *A* и визуализации промежуточных результатов.

Обратите внимание на инструкцию **Sheets("Ñîðòèðîâêà").Select**. В результате выполнения **Select** выбирается рабочий лист Сортировка и далее при работе с ячейками рабочего листа обращение ведется только к листу Сортировка.

```
Public Sub Arr6()
    Dim n As Integer, a() As Integer
    Sheets("Сортировка").Select
    Range("A:F").Clear
    n = InputBox("Введите размер массива")
    ReDim a(n) As Integer
    PrVvod "Сортировка", -10, 10, n, a
    SortTras a, n
End Sub
```

6.5. Запустите процедуру Arr6. Введите размерность массива равную 4.

6.6. В результате на рабочем листе Сортировка появляется результат работы программы, представленный на рис. 43 (поскольку массив заполняется случайными числами, столбец A и E у вас выглядит по-другому).

	A	B	C	D	E	F
1	-2	i	j		Элементы массива	
2	10	1	2		-2 10 -7 4	
3	-7	1	3		-7 10 -2 4	
4	4	1	4		-7 10 -2 4	
5						
6		2	3		-7 -2 10 4	
7		2	4		-7 -2 10 4	
8						
9		3	4		-7 -2 4 10	
10						
11						
12						

Рисунок 43. Визуализация процедуры сортировки
на рабочем листе MS Excel

Массив *A* записан в первом столбце. Второй и третий столбцы содержат значения *i, j* для каждой итерации. Элементы массива записаны в виде строковой константы. Если цвет строки, содержащей элементы массива, не черный, то порядок элементов массива изменялся: менялись местами *A(i)* и *A(j)* элементы и в результате получен массив, элементы которого располагаются в порядке, указанном в столбце E.

Например (см. рис.43), после сравнения первого элемента со вторым порядок элементов массива не изменился (элементы массива выделены черным цветом), после сравнения первого элемента с третьим произошел обмен между первым и третьим элементом массива и в результате получен следующий порядок элементов массива: -7, 10, -2, 4 (элементы массива выделены голубым цветом). Далее сравниваются первый и четвертый элементы (-7 и 4), порядок элементов массива A не изменился.

Переходим ко второй итерации: сравнивается второй элемент массива с третьим и четвертым, наименьший становится вторым элементом массива.

Далее третий элемент массива сравнивается с четвертым. Т.о. последняя заполненная ячейка столбца E содержит упорядоченный по возрастанию массив A.

Разберите результаты работы вашей программы.

6.7. Теперь создадим процедуру Arr61, для упорядочивания массива *A* без визуализации промежуточных результатов.

```
Public Sub Arr61()
    Dim n As Integer, a() As Integer
    Sheets("Сортировка").Select
    Range("A:E").Clear
    n = InputBox("Введите размерность массива")
    ReDim a(n) As Integer
    PrVvod "Сортировка", -10, 10, n, a
    Sort a, n
    'вывод упорядоченного массива
    For i = 1 To n
        Cells(i, 2) = a(i)
    Next
End Sub
```

6.8. Запустите процедуру Arr61. Убедитесь, что задача сортировки массива решена правильно.

Замечание

При сортировке массива по убыванию также можно воспользоваться алгоритмом прямого обмена. Достаточно поменять знак в условном операторе тела цикла.

```
If a(i) < a(j) Then
   Temp = a(j)
   a(j) = a(i)
   a(i) = Te
End If
```

6.8.* Применим процедуру sort для сортировки названий листов активной рабочей книги. Создайте приведенную ниже процедуру и запустите ее. Система сообщит об ошибке. Попытайтесь найти и исправить ошибку.

```
Sub SortSheets()
    Dim SheetNames() As Variant
    Dim SheetCount As Integer
  'Определение количества листов,
  'изменение размерности массива
  'Свойство Count коллекции Sheets
  'возвращает число раб. листов
    SheetCount = ActiveWorkbook.Sheets.Count
    ReDim SheetNames(1 To SheetCount) As Variant
  'Заполнение массива названиями листов
  'Свойство Name коллекции Sheets
  'возвращает имя рабочего листа
    For i = 1 To SheetCount
       SheetNames(i) = ActiveWorkbook.Sheets(i).Name
    Next
  'Сортировка массива в возрастающем порядке
    sort SheetNames, SheetCount
  'Метод Move коллекции Sheets
  'выполняет перемещение листа в другоеместо рабочей книги
    For i = 1 To SheetCount
       Sheets(SheetNames(i)).Move Before:=Sheets(i)
    Next
End Sub
```

Задача 7.

Найти сумму отрицательных элементов в каждой строке матрицы A(n,m).

7.1. Разработка алгоритма

В отличие от одномерного массива, в котором использовался только один номер для определения местоположения элемента и требовался только один цикл для ввода/вывода и обработки элементов массива, в двумерном массиве для обра-

ботки и ввода/вывода элементов необходимы два вложенных друг в друга цикла. Внешний цикл предназначен для изменения номера строки *i*, а второй, внутренний, - для изменения номера столбца *j* в текущей строке *i*.

Будем генерировать элементы матрицы с помощью формулы генерации случайных чисел на отрезке [b,c] A(i, j)=b+(c-b)*rnd().

С помощью цикла

```
For j = 1 To m
     a(1,j) = c + (d - c) * Rnd()
Next
```

можно заполнить первую строку матрицы *A(n,m)*.

Аналогично вторую строку заполняем с помощью цикла

```
For j = 1 To m
     a(2,j) = c + (d - c) * Rnd()
Next
```

Т.о. надо *n* раз повторить оператор цикла для заполнения *n* строк матрицы, изменяя лишь номер строки в операторе, стоящем в теле цикла. Следовательно, можно заключить оператор цикла также в цикл:

```
For i = 1 To n
    For j = 1 To m
        a(i, j) = b + (c - b) * Rnd()
    Next
Next
```

Для разработки алгоритма вычисления в строках и столбцах двумерного массива можно воспользоваться следующим методом: поскольку строки (столбцы) матрицы представляют собой одномерные массивы, то, зафиксировав строку (столбец), решаем поставленную задачу для одномерного массива (в рассматриваемом примере можно применить алгоритм нахождения суммы отрицательных элементов одномерного массива), далее помещаем разработанный алгоритм в тело внешнего цикла по строкам (столбцам).

Найдем сумму отрицательных элементов в первой строке матрицы.

```
s = 0
For j = 1 To m
    If a(1, j) < 0 Then s = s + a(1, j)
Next
```

Аналогично найдем сумму отрицательных элементов во второй строке матрицы.

```
s = 0
For j = 1 To m
```

```
    If a(2, j) < 0 Then s = s + a(1, j)
Next
```

Если запоминать сумму отрицательных элементов каждой строки матрицы в одной ячейке *s*, то продолжая вычисления до строки с номером *n*, в ячейке *s* окажется лишь сумма отрицательных элементов последней строки.

Следовательно, сумму отрицательных элементов строки с номером *i* нужно хранить в ячейке *s(i),* т.о. получаем одномерный массив *s*.

Решение задачи сводится к выполнению последовательности шагов:

1 шаг. Находим сумму отрицательных элементов в первой строке.

```
s(1) = 0
For j = 1 To m
    If a(1, j) < 0 Then s(1) = s(1) + a(1, j)
Next
```

2 шаг. Находим сумму отрицательных элементов во второй строке.

```
s(2) = 0
For j = 1 To m
    If a(1, j) < 0 Then s(2) = s(2) + a(1, j)
Next
```

и т.д.

i шаг. Находим сумму отрицательных элементов в i-той строке.

```
s(i) = 0
For j = 1 To m
    If a(1, j) < 0 Then s(i) = s(i) + a(1, j)
Next
```

и т.д.

n шаг. Находим сумму отрицательных элементов в n-той строке.

```
s(n) = 0
For j = 1 To m
    If a(1, j) < 0 Then s(n) = s(n) + a(1, j)
Next
```

Очевидно, что решение задачи можно свести к оператору цикла по переменной *i*, которая определяет номер строки.

```
For i = 1 To n
      S(i) = 0
      For j = 1 To m
         If A(i, j) < 0 Then S(i) = S(i) + A(i, j)
      Next
Next
```

7.2. Реализуем решение задачи в среде VBA.

Исходные данные и результат работы программы поместим на рабочий лист Excel.

В окно редактирования кода объекта Module1 создайте процедуру PR_ForFor. Обратите внимание, что двумерный массив A объявляется как динамический.

```
Public Sub PR_ForFor()
   Dim A() As Single
   Dim S() As Single
   n = InputBox("Введите количество строк")
   m = InputBox("Введите количество столбцов")
   ReDim A(1 To n, 1 To m) As Single
   ReDim S(1 To n) As Single
  'Очистка столбцов рабочего листа для
  'ввода новых данных
   Sheets(2).Range("A:Y").Clear
   Cells(1, 1) = "Матрица"
   Cells(1,m+2)="Сумма отриц-ных элементов строки"
   c = -3: b = 5
  'Заполнение матрицы и
  'перенос элементов матрицы на рабочий лист
   For i = 1 To n
     For j = 1 To m
        A(i, j) = c + (b - c) * Rnd()
        Cells(i + 1, j) = A(i, j)
     Next
   Next
  'Цикл по строкам
   For i = 1 To n
      S(i) = 0
    'Цикл по столбцам
      For j = 1 To m
  'Если отрицательный элемент найден,
  'увеличить сумму на  его значение
         If A(i, j) < 0 Then S(i) = S(i) + A(i, j)
      Next
    'Результат перенести в ячейку рабочего листа
      Cells(i + 1, m + 2) = S(i)
   Next
End Sub
```

7.3. Запустите процедуру PR_ForFor.

Результат работы процедуры приведен на рис.44 .

	A	B	C	D	E	F	G	H	I	J
1	Матрица						Сумма отрицательных элементов строки			
2	3,900955	3,32384	-0,01171	4,695625	3,971567		-0,01171			
3	-2,55011	4,596453	-0,08785	1,198947	3,136893		-2,63796			
4	-2,57196	1,739666	0,749601	-0,61468	1,981574		-3,18664			
5										

Рисунок 44. Результат работы процедуры PR_ForFor

7.4. Решите следующую задачу: найти произведение четных чисел в каждом столбце матрицы A(7,8).

Лабораторная работа №10

Тема: Создание пользовательской формы. Алгоритм нахождения максимума и минимума одномерного массива.

Задача: определить максимальный и минимальный доход фирмы за 20 лет.

Для решения поставленной задачи применим технологию объектно-ориентированного программирования, включающую технологию процедурного программирования.

1. Формализация задачи. Решение поставленной задачи сводится в нахождению минимума и максимума одномерного массива. Определим исходные и выходные данные.

Исходные данные: одномерный массив A(20), состоящий из вещественных чисел, каждое из которых определяет доход фирмы за *i*-й год (i=1,2,…20).

Выходные данные: max— максимальный доход фирмы, min — минимальный доход фирмы.

2. **Разработка алгоритма**. Для решения поставленной задачи можно применить стандартный алгоритм нахождения максимума [минимума].

Рассмотрим алгоритм поиска элемента с максимальным значением в одномерном массиве A(N) .

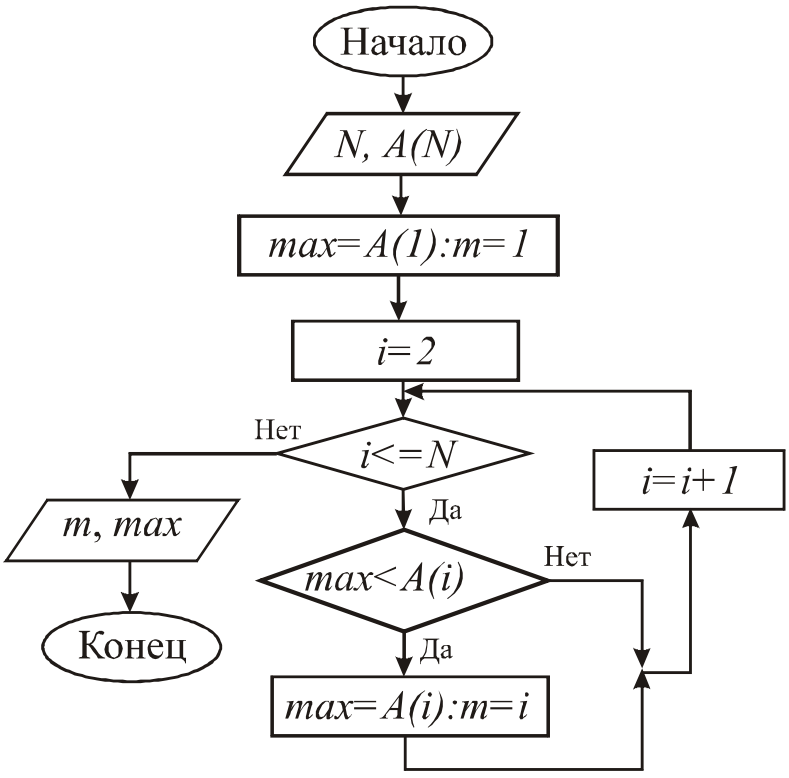


Рисунок 45. Алгоритм поиска элемента с максимальным значением в одномерном массиве A(N)

Введем обозначения *i* — текущий номер элемента, *A(i)* — текущее значение элемента массива, *N* — количество элементов одномерного массива, *m* — номер максимального элемента массива, *max* — значение максимального элемента массива.

Основной идеей алгоритма является выполнение сравнения текущего элемента массива *A(i)* и элемента с максимальным значением *max*, определенным на предыдущем шаге итерации.

По алгоритму изображенному на рис.45 в качестве начального максимального элемента берется первый элемент массива. Поскольку сравнивать первый элемент массива с собой не имеет смысла, цикл выполняется, начиная со второго элемента.

Для нахождения максимального элемента в массиве запишем приведенный выше алгоритм на языке VBA.

```
Max= a(1):M=1
For i = 2 To N
     If Max < a(i) Then Max= a(i):M=i
Next
```

Для лучшего понимания работы алгоритма составим таблицу трассировки для массива: *a(1)* = -5, *a(2)* = 6, *a(3)* = 5, *a(4)* = 6 (*N*=4).

Оператор		i	m	max
`Max= a(1):M=1`		–	1	-5
`For i = 2 To N`	1-я итерация	2	1	-5
`If Max< a(i) Then Max= a(i):M=i`	1-я итерация	2	2	6
`Next`	1-я итерация	3	2	6
`For i = 2 To N`	2-я итерация	3	2	6
`If Max< a(i) Then Max= a(i):M=i`	2-я итерация	3	2	6
`Next`	2-я итерация	4	2	6
`For i = 2 To N`	3-я итерация	4	2	6
`If Max< a(i) Then Max= a(i):M=i`	3-я итерация	4	2	6
`Next`	3-я итерация	5	2	6

Т.о. максимальный элемент равен 6, а его индекс в массиве *a* равен 2.

Немного изменим алгоритм нахождения максимального элемента в массиве.

```
Max= a(1):M=1
For i = 2 To N
     If Max <= a(i) Then Max= a(i):M=i
Next
```

Очевидно, что применив данный алгоритм к примеру, получим, что максимум также равен 6, но его индекс равен 4.

Итак, если применяется строгое неравенство в условном операторе, то находим максимальный элемент с минимальным индексом, если нестрогое неравенство, то — максимальный элемент с максимальным индексом.

Аналогично приведенному выше алгоритму вычисляется минимальный элемент одномерного массива, необходимо лишь в условном операторе знак отношения поменять на противоположный.

3. Реализация алгоритма на ЭВМ.

Прежде чем приступить к написанию программного кода спроектируем пользовательскую форму и определим где будут находиться исходные данные и каким образом будет осуществляться вывод выходных данных.

Пусть исходные данные содержится на листе Данные рабочей книге в диапазоне A2:A21 (см. рис.46; данные можно сгенерировать с помощью формулы =-500+1100*СЛЧИС(); далее нужно выделить диапазон A2:A21; скопировать данные в буфер обмена, выполнить команду **Правка/Специальная вставка** (Для Microsoft Office 2007 **–** Вкладка **Главная,** панель **Буфер обмена,** в списке **Вставить** команда **Специальная вставка)**, в диалоговом окне выбрать переключатель Значения).

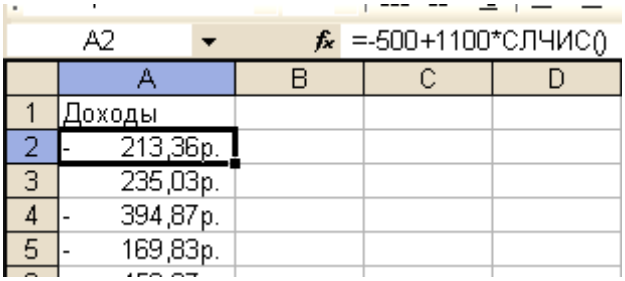


Рисунок 46. Фрагмент рабочего листа
с генерацией случайных чисел

Результат выведем в текстовое поле в пользовательской форме. Кроме этого, выделим цветом ячейки рабочего листа, содержащие минимальный и максимальный элементы.

4. **Создание пользовательской формы**

4.1. Выберите команду **Сервис/Макрос/Редактор Visual Basic**.

4.2. Выберите команду **Insert/UserForm**. В редакторе Visual Basic появятся: окно с пользовательской формой и панель инструментов **Панель элементов**. Необходимые для проектирования формы элементы указаны на рисунке 47: пользовательская форма имеет одну надпись, одно поле ввода, две кнопки.

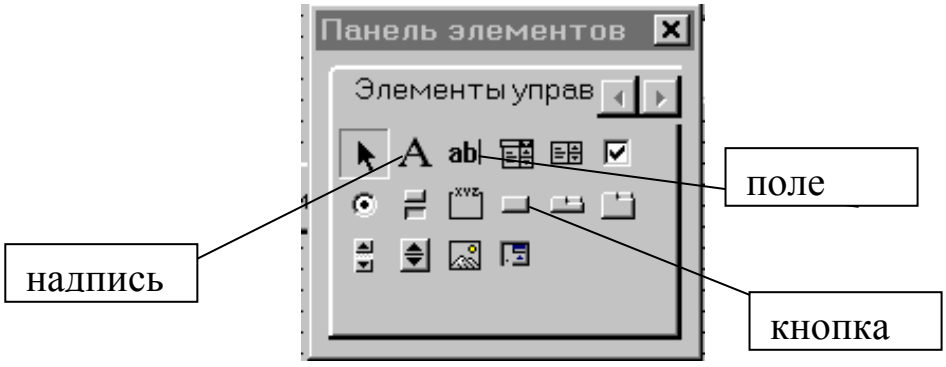


Рисунок 47. Панель элементов

4.3. Используя диалоговое окно Свойства (для вызова окна свойств выполните команду **View/Properties Windows**) и Панель элементов, простым перетаскиванием элементов управления на форму, создайте из пользовательской формы диалоговое окно, показанное на рис. 48.

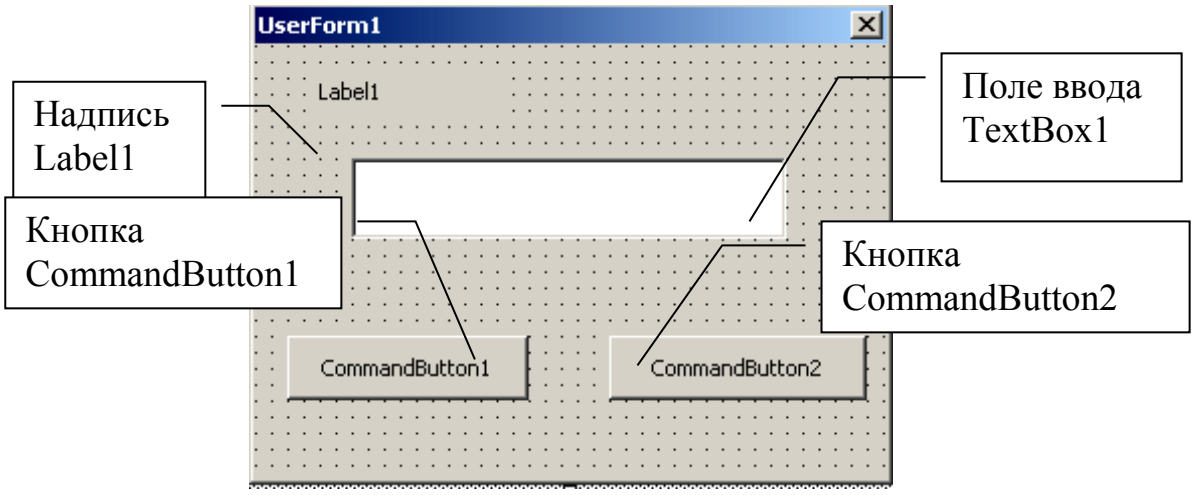


Рисунок 48. Создание пользовательской формы

4.4. Определите значение свойства Caption (Надпись, отображаемая на элементе управления, для формы — текст, отображаемый в строке заголовка формы) для элементов управления и формы (Табл.10).

Таблица 10. *Значение Свойства* ***Caption*** *элементов управления и формы*

Объект	Значение
UserForm1	Нахождение максимального и минимального дохода
Label1	Результат
CommandButton1	Минимум
CommandButton2	Максимум

4.5. Для того чтобы написать процедуру обработки событий нажатия кнопки минимум или максимум (событие Click), дважды щелкните на элементе управления CommandButton. Можно воспользоваться командой контекстного

меню формы View Code. Далее в модуле формы можно воспользоваться выбором объекта, помещенного на форму и события с ним связанного. **Имя событийной процедуры вводить вручную не следует.** (См. рис. 49)

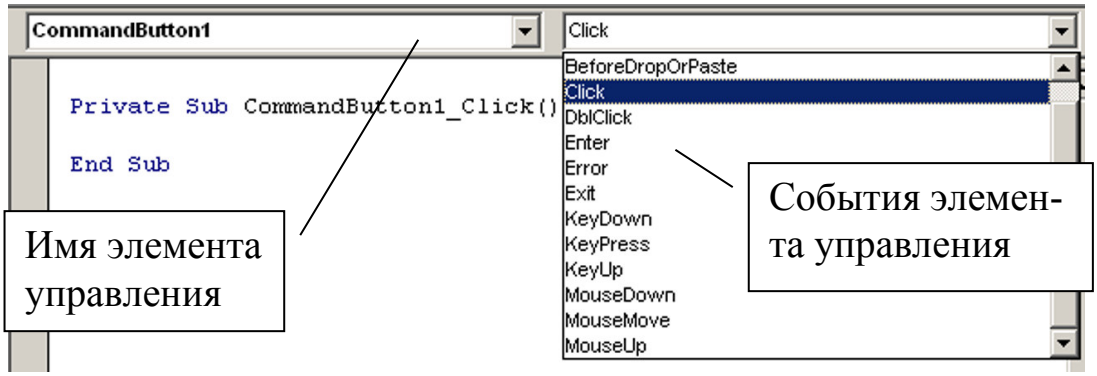


Рисунок 49. Выбором объекта, помещенного на форму и события с ним связанного

4.6. В окно редактирования кода необходимо ввести программный код приведенный ниже.

```
Dim a(20) As Single
Private Sub CommandButton1_Click()
   Min = a(1): M=1
   For i = 2 To 20
     If Min > a(i) Then Min = a(i):M=i
   Next
   TextBox1.Text = Min
 'Выделить желтым цветом ячейку с минимальным значением
   Cells(M+1,1).Interior.ColorIndex=6
End Sub
Private Sub CommandButton2_Click()
  Max= a(1):M=1
   For i = 2 To 20
     If Max< a(i) Then Max= a(i):M=i
   Next
   TextBox1.Text = Max
 'Выделить синим цветом ячейку с максимальным значением
   Cells(M+1,1).Interior.ColorIndex=5
End Sub

Private Sub UserForm_Initialize()
'Очистить форматы в столбце А
   Range("A:A").ClearFormats
  'считывание данных в массив с рабочего листа
   For i = 1 To 20
     a(i) = Cells(i + 1, 1)
```

```
    Next
End Sub
```

4.7. Выполнить команду **Debug/Compile VBA Project.**

5. Процесс создания диалогового окна и процедур, связанных с ним, завершен. Для того чтобы проверить как работает созданная программа, нажмите кнопку F5. На экране на фоне рабочего листа отобразится диалоговое окно Нахождение максимального и минимального дохода. Поочередно нажмите кнопки минимум и максимум в текстовом поле отобразится минимум и максимум исходного массива соответственно (см. рис.50), на рабочем листе ячейки, содержащие минимум и максимум будут выделены цветом.

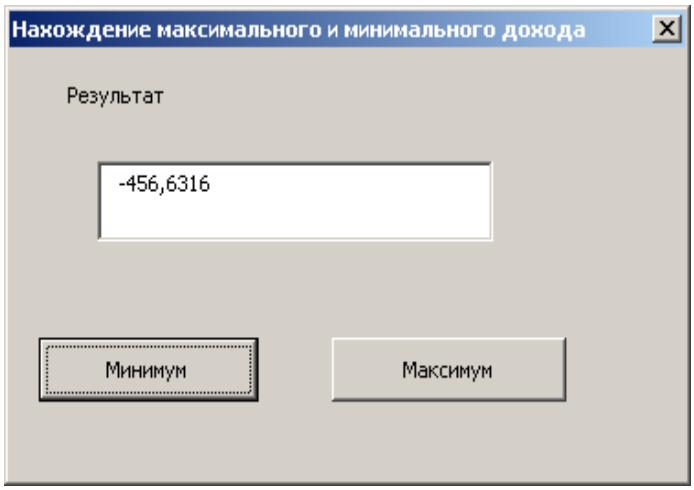


Рисунок 50. Пользовательская форма для вычисления минимального и максимального элемента массива

6. Для запуска формы с рабочего листа Excel выполните следующие действия.

6.1. Откройте вкладку **Разработчик**

6.2. Нажмите кнопку Режим конструктора 

на панели **Элементы управления**

6.3. Нажмите кнопку Вставить и выберите элемент Кнопка в элементах управления ActiveX и перенесите его на рабочий лист.

7. Выполните двойной щелчок на кнопке и организуйте следующую процедуру

```
Private Sub CommandButton1_Click()
    UserForm1.Show
End Sub
```

Лабораторная работа №11

Тема: Заполнение элемента управления Список. Выбор нескольких элементов из списка. Выполнение специфицированных операций над выбранными элементами из списка.

Задача: Найти сумму, произведение, среднее арифметическое выбранных элементов некоторого набора чисел.

Для решения поставленной задачи применим технологию объектно-ориентированного программирования, включающую технологию процедурного программирования.

1. **Формализация задачи.** Определим исходные и выходные данные.

Исходные данные: набор чисел которые можно записать как одномерный массив (1, 3, 4, 5 ,6, 7, 8, 10)
Выходные данные: s — сумма выбранных элементов, p — произведение выбранных элементов, sa — среднее арифметическое выбранных элементов.

2. **Разработка алгоритма**. Для решения поставленной задачи можно применить стандартные алгоритмы нахождения суммы и произведения. В переменной k хранится количество выбранных элементов. (См. рис. 51)

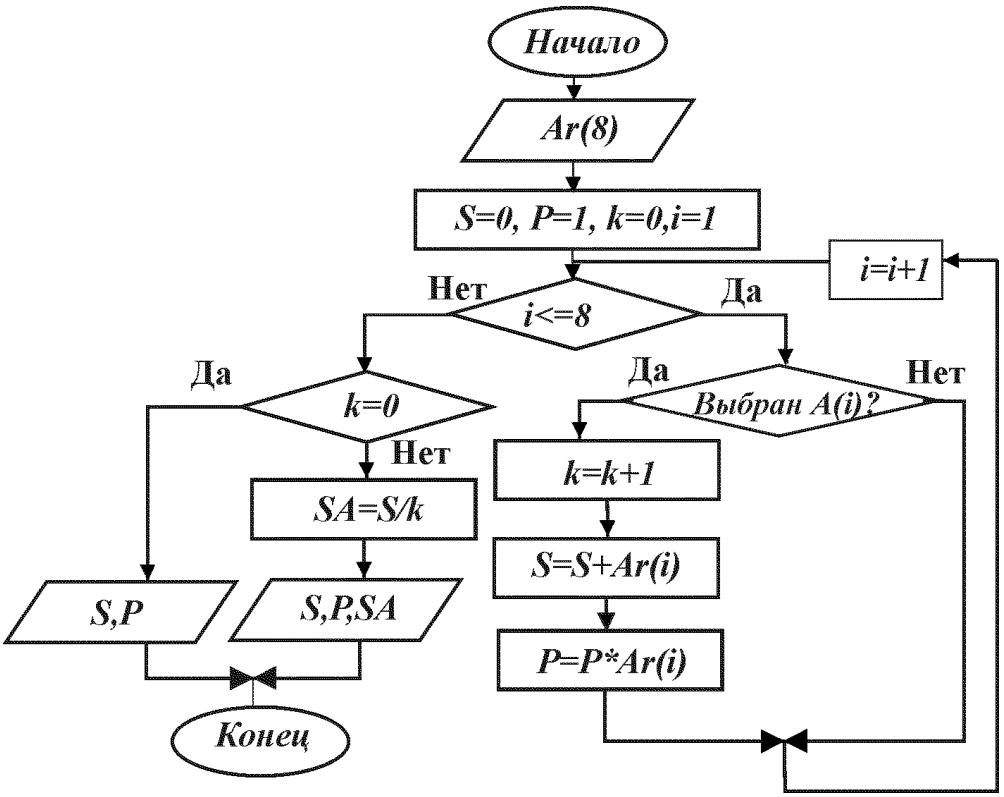


Рисунок 51.

3. Реализация алгоритма на ЭВМ.

Создадим приложение, которое позволит выбрать несколько чисел, выводимых в списке в диалоговом окне Операции над элементами списка.

3.1. Выберите команду **Сервис/Макрос/Редактор Visual Basic**.

3.2. Выберите команду **Insert/UserForm**. В редакторе Visual Basic появятся: окно с пользовательской формой и панель инструментов Панель элементов. Необходимые для проектирования формы элементы указаны на рисунке: пользовательская форма имеет список, одно поле ввода, две кнопки, три переключателя, рамку, надпись. В группе Операция следует установить один из переключателей: Сумма, Произведение или Среднее, чтобы указать какая операция будет выполняться над выбранными числами. Нажатие кнопки Вычислить должно привести к выполнению операции и выводу результата в поле Результат. (См. рис.52)

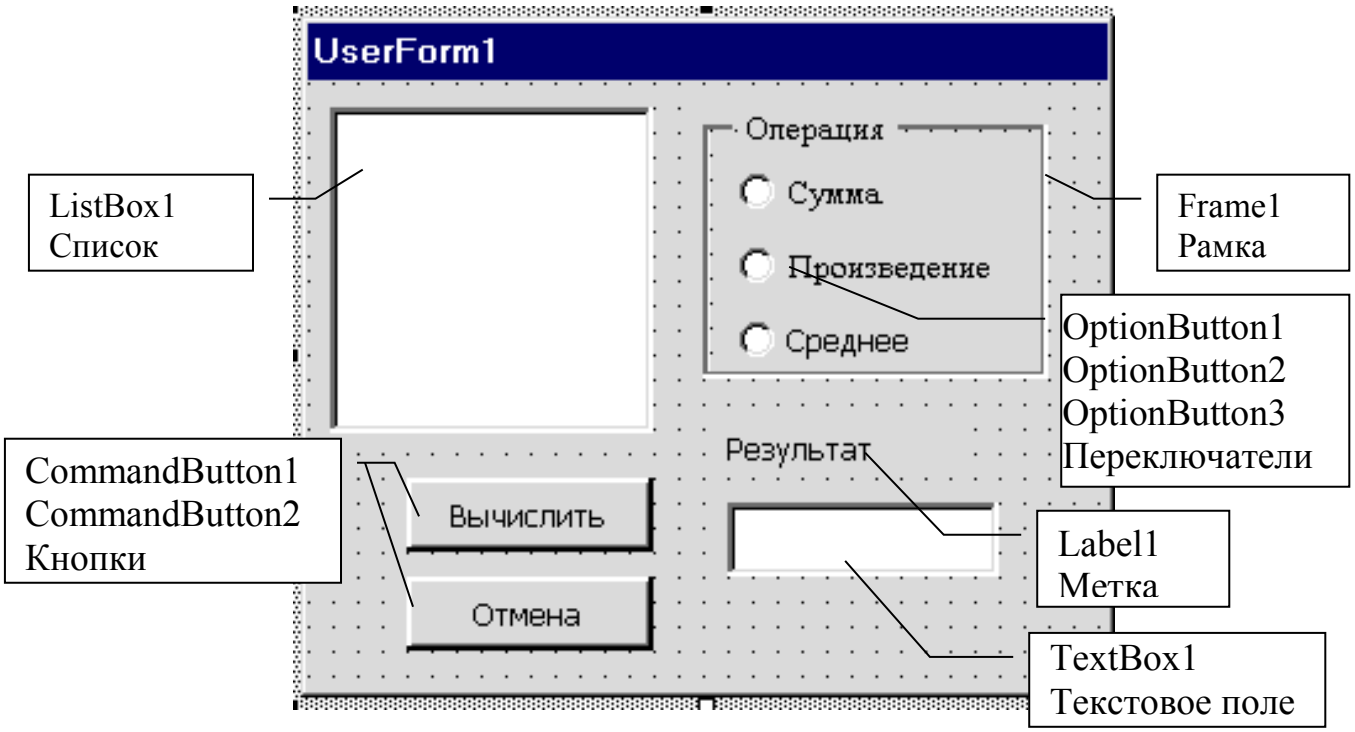


Рисунок 52. Проектирование пользовательской формы

В редакторе VBA создать пользовательскую форму. Значения свойства Caption элементов управления приведены в таблице 11.

Таблица 11. *Значение Свойства Caption элементов управления*

Объект	Значение
Frame1	Операция
OptionButton1	Сумма
OptionButton2	Произведение
OptionButton3	Среднее

Label1	Результат
CommandButton1	Вычислить
CommandButton2	Отмена

3.3. Приведенные ниже процедуры связать с элементами пользовательской формы. Комментарии можно не вводить в код программы. Они приведены для объяснения логики программы.

```
Private Sub ListBox1_Change()
  'Процедура очищает поле Результат при изменении Списка
    TextBox1.Text = ""
End Sub

Private Sub OptionButton1_Click()
  'Процедура очищает поле Результат при щелчке на переключателе
  TextBox1.Text = ""
End Sub

Private Sub OptionButton2_Click()
  'Процедура очищает поле Результат при щелчке на переключателе
    TextBox1.Text = ""
End Sub

Private Sub OptionButton3_Click()
  'Процедура очищает поле Результат при щелчке на переключателе
    TextBox1.Text = ""
End Sub

Private Sub CommandButton1_Click()
  'Процедура проведения вычислений с выбранными элементами
  'списка в зависимости от выбранной операции
  Dim i As Integer 'i-вспомогательная переменная
  Dim n As Integer
  'n-играет роль счетчика числа выбранных элементов из списка
  Dim s As Double  's-сумма выбранных элементов из списка
  Dim p As Double
  'p-произведение выбранных элементов из списка
  Dim sa As Double 'sa-среднее арифметическое значение
  'выбранных элементов из  списка
  Dim r As Double 'r-в эту переменную записывается результат,
  'найденный в зависимости от выбранной операции
  'при выборе первого переключателя вычисляется
  'сумма выбранных элементов
  If OptionButton1.Value = True Then
     s = 0
```

```
    With ListBox1
    For i = 0 To .ListCount - 1
  'свойство списка ListCount возвращает
  'число элементов списка
        If .Selected(i) = True Then
            s = s + .List(i)
   'свойство списка Selected имеет допустимые значения
   'True (если элемент списка выбран) и False (в противном случае)
   'свойство списка List возвращает элемент списка,
   'стоящий на пересечении указанных строки и столбца
        End If
    Next i
    End With
    r = s
End If
'при выборе второго переключателя вычисляется
'произведение выбранных элементов
If OptionButton2.Value = True Then
    p = 1
    With ListBox1
    For i = 0 To .ListCount - 1
          If .Selected(i) = True Then
             p = p * .List(i)
          End If
    Next i
    End With
    r = p
End If
'при выборе третьего переключателя вычисляется ср. арифметическое
If OptionButton3.Value = True Then
    sa = 0
    n = 0
    With ListBox1
    For i = 0 To .ListCount - 1
          If .Selected(i) = True Then
             n = n + 1
             sa = sa + .List(i)
          End If
    Next i
    End With
    If n <> 0 Then
      r = sa / n
    End If
```

```
  End If
  'результат выводится в текстовое поле
  TextBox1.Text = CStr(Format(r, "fixed"))
End Sub
Private Sub CommandButton2_Click()
  'процедура закрытия диалогового окна
   UserForm1.Hide
End Sub
Private Sub UserForm_Initialize()
  'процедура инициализации диалогового окна
  'заполнение списка и установка режима выбора
  'нескольких элементов из списка
With ListBox1
      .List = Array(1, 3, 4, 5, 6, 7, 8, 10)
      .ListIndex = 0
      .MultiSelect = fmMultiSelectMulti
     'свойство списка ListIndex возвращает номер текущего элем. списка
     'нумерация элементов списка начинается снуля
     'сво-во списка MultiSelect устанавливает способ выбора элем. списка
    'выбор только одного элемента, нескольких элементов
End With
  'первоначальный выбор переключателя сумма при
  'инициализации диалогового окна и задание
  'текста всплывающих подсказок у переключателей
With OptionButton1
      .Value = True
      .ControlTipText = "Сумма выбранных элементов"
End With
OptionButton2.ControlTipText = "Произведение элементов"
OptionButton3.ControlTipText="Среднее значение элементов"
  'Поле Результат не доступно для пользователя
  TextBox1.Enabled = True
  'назначение клавише <Enter> функции кнопки вычислить
  'и задание текста всплывающей подсказки
  With CommandButton1
       .Default = True
       .ControlTipText = "Нахождение результата"
  End With
  'Задание заголовка пользовательской формы
  UserForm1.Caption = "Операция над элементами списка"
End Sub
```

3.4. Выполнить команду **Debug/Compile VBA Project**.

4. Процесс создания диалогового окна и процедур, связанных с ним, завершен. Для того чтобы проверить, как работает созданная программа, нажмите кнопку F5.

5. Создайте на рабочем листе кнопку для запуска разработанной формы.

Лабораторная работа №12*

Тема: Использование элемента управления RefEdit. Перехват и обработка ошибок. Пример управления видимостью элементов управления. Решение задачи с данными, которые имеют структуру двумерного массива.

Задача: Ведется учет годовых объемов добычи нефти некоторыми компаниями. Для каждой компании найти максимальный период, в течение которого компания увеличивала добычу нефти, — определить с какого по какой год длился этот период.

1. **Формализация задачи.** Определим исходные и выходные данные.

Пусть учет ведется в таблице Excel (см. рис. 53). Задача может быть решена для любого периода, который мы выберем, кроме того, количество компаний может меняться.

	A	B	C	D	E	F	G
1		Компании					
2	Год	Компания1	Компания2	Компания3	Компания4	Компания5	Компания6
3	1994	5674,0013	3704,7639	7175,3899	9293,7407	9192,2666	9709,2502
4	1995	3101,474	5851,9547	9042,7259	3970,0919	4715,2318	3318,3081
5	1996	3226,661	4148,8998	4537,2784	3119,6326	4995,3001	5401,6236
6	1997	6875,454	5501,6022	5602,8626	5489,2117	9372,1427	6262,1235
7	1998	5983,1233	5127,3232	9829,9509	8646,6567	9938,6883	4793,8475
8	1999	9661,8244	3374,0654	7935,2702	8715,659	9807,5198	6264,2598
9	2000	5101,474	8251,442	5460,3717	8429,6091	3520,4016	4389,0194
10	2001	3448,4085	5508,4384	6409,3142	6578,5089	5614,185	9901,3031
11	2002	3284,9818	4615,0395	3034,8216	9483,0164	3702,2004	4796,8383
12	2003	8429,8227	7757,5304	8663,7471	8070,2841	3595,3856	3925,8705
13	2004	8293,0998	7385,6014	4215,5522	5833,5826	6866,2679	7980,5597
14							

Рисунок 53. Исходные данные

Целесообразно ввести исходные данные следующим образом: названия компаний объединить в одномерный динамический массив символьного типа Naz(), года в интересующий нас период объединить в числовой динамический массив a(), объемы добычи компаний по годам — в числовой динамический двумерный массив DN().

Выходные данные: MaxL — максимальный период добычи нефти, в течение которого компания увеличивала добычу нефти, fin — последний год максимального периода увеличения добычи, fin-MaxL — первый год максимального периода увеличения добычи нефти.

2. **Разработка алгоритма**.

Разберем способ решения поставленной задачи.

Может случиться, что ни у одной компании не обнаружиться ни одного периода прироста добычи. Следовательно, нужно ввести переменную-флажок (F), которая будет фиксировать факт наличия периода нарастания хотя бы у одной

компании. Ее следует обнулить перед началом вычислений и присвоить ей значение 1, как только выявится период прироста у какой-либо компании.

Обработку данных произведем с помощью вложенных циклических операторов с параметром.

Во внешнем цикле будем перебирать поочередно компании. Для каждой компании будем фиксировать период возрастания (их может оказаться несколько или не быть вообще) и находить среди них наиболее продолжительный, следовательно необходимы переменные: l — текущая длительность периода для данной компании, MaxL — максимальная длина периода увеличения объема добычи нефти для текущей компании. Этим переменным присвоим нулевые стартовые значения (в теле внешнего цикла). Далее просматриваем выпуски данной компании по годам во внутреннем цикле, отслеживая периоды прироста: прирост имеет место, если выпуски предыдущего года меньше выпуска текущего. Следовательно, нужно увеличить значение l на единицу и сравнить ее значение со значением переменной MaxL. Если текущий период оказался больше максимума, то необходимо переопределить значение переменной MaxL с фиксацией окончания данного периода в переменной . Как только период прироста заканчивается надо обнулить переменную l.смотрев для данной компании выпуск за все годы, будем выводить результат, но только в том случае, если у нее был хотя бы один период прироста (MaxL>0).

Завершив просмотр всех компаний, необходимо проверить состояние флажка.

Заметим, что максимальных периодов прироста у каждой компании может быть несколько. Будем предлагать пользователь самому определять какой период прироста ему нужен — первый или последний. Реализуем это с помощью добавления в пользовательскую форму элемента управления Флажок. Ввод данных будем выполнять с помощью элемента управления RefEdit.

3. Реализация алгоритма на ЭВМ.

Прежде чем приступить к написанию программного кода спроектируем пользовательскую форму и определим где будут находиться исходные данные и каким образом будет осуществляться вывод выходных данных.

Итак, учет ведется на рабочем листе **Данные** (см. рис.53; данные можно сгенерировать, например, с помощью формулы =3000+7000*СЛЧИС(); далее нужно выделить диапазон B3:G13; скопировать данные в буфер обмена, выполнить команду Правка/Специальная вставка, в диалоговом окне выбрать переключатель Значения).

Результат будем выводить в ячейки рабочего листа **Результат**.

Проверьте есть ли в рабочей книге рабочие листы Данные и Результат!

Создание пользовательской формы

3.1. Войдите в **редактор Visual Basic**.

3.2. Выберите команду **Insert/UserForm**. В редакторе Visual Basic появятся: окно с пользовательской формой и панель инструментов Панель элементов. Пользовательская форма должна быть спроектирована согласно рис. 54. Пользовательская форма имеет следующие элементы управления: три надписи, три поля ввода RefEdit, две кнопки, два элемента управления Флажок, две рамки.

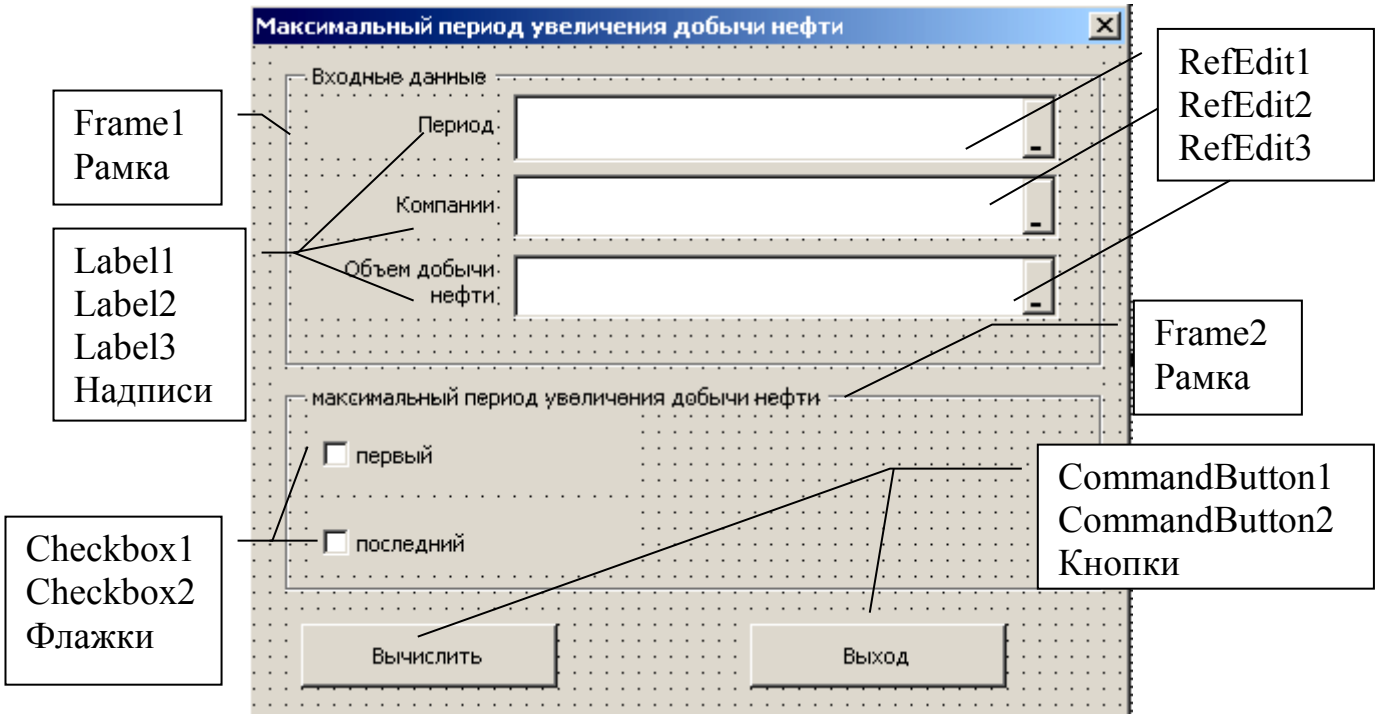


Рисунок 54.

Элемент управления RefEdit аналогичен полю ввода, но позволяет вводить в него ссылку на диапазон выбором на рабочем листе. Данный элемент управления выглядит иначе, чем элемент управления выбора диапазона во встроенных диалоговых окнах Excel, однако работает точно также. Если пользователь щелкнет в правой части элемента управления, то диалоговое окно временно исчезнет, а на экране будет отображен небольшой указатель диапазона (именно так все происходит и при использовании встроенного диалогового окна Excel). Если во время проектирования формы элемент управления RefEdit отсутствует на панели инструментов необходимо выполнить следующие действия:

1. щелкнуть правой клавишей мыши на панели инструментов;
2. выбрать команду Additional Controls (Добавить компоненты);
3. в диалоговом окне Additional Controls в списке элементов управления отметить RefEdit.Ctrl и нажать кнопку OK. На панели инструментов появится значок  .

Значения свойств элементов управления приведены в таблице 12.

Таблица 12. *Значения Свойств элементов управления*

Объект	Значение
Значение Свойства Caption	
Frame1	Входные данные
Frame2	Максимальный период увеличения добычи нефти
Label1	Период
Label2	Компании
Label3	Объем добычи нефти
CheckBox1	первый
CheckBox2	последний
CommandButton1	Вычислить
CommandButton2	Выход
Значение Свойства Visible (видимости элемента управления)	
CommandButton1	True
CommandButton2	False

При запуске Формы будет видна только кнопка **Вычислить**. Кнопка **Выход** появится после выполнения событийной процедуры, связанной с CommandButton1. Значение свойства Visible объекта CommandButton2 определяется программно.

Приведенные ниже процедуры связать с элементами пользовательской формы. Комментарии можно не вводить в код программы. Они приведены для объяснения логики программы.

```
'Изменение базового индекса массива
Option Base 1

Private Sub CommandButton1_Click()
'Объявление массивов как динамических
Dim a() As Single
Dim Naz() As String
Dim DN() As Single
'Оператор On Error производит перехват ошибки,
'переводит управление на метку ErrorHandler
'в случае возникновения ошибки
'Пока программа не отлажена, 10 строку программы
'необходимо представить как комментарий
'10 On Error GoTo ErrorHandler
```

```
m = 0
r = RefEdit1.Value
'Создаем ссылку на объект Range,
'в котором содержатся ячейки с годом
Set rng1 = Range(r)
'Используем оператор For Each для перебора
'ячеек диапазона,
'записываем значения ячеек, содержащих год, в массив a()
'Переменной m присваиваем значение размера массива и
'меняем размерность массива a()
For Each c In rng1
  m = m + 1
  ReDim Preserve a(m) As Single
  a(m) = c
Next
n = 0
r = RefEdit2.Value
'Создаем ссылку на объект Range,
'содержащий ячейки с названиями компаний
Set rng2 = Range(r)
'Используем оператор For Each
'для перебора ячеек диапазона,
'записываем значения ячеек, 'содержащих год, в массив Naz()
'Переменной n присваиваем значение размера массива и
'меняем размерность массива Naz()
For Each c In rng2
  n = n + 1
  ReDim Preserve Naz(n) As String
  Naz(n) = c
Next
'Меняем размерность массива DN()
ReDim DN(m, n) As Single
r = RefEdit3.Value
'Создаем ссылку на объект Range,
'содержащий ячейки со значениями
'объема добычи нефти за m лет в n компаниях
Set rng3 = Range(r)
'Проверка размерности диапазона  rng3
If rng3.Rows.Count = m And rng3.Columns.Count = n Then
'Записываем значения ячеек, содержащихся в rng3,
'в массив DN(m,n)
  For i = 1 To m
  For j = 1 To n
```

```
    DN(i, j) = rng3.Cells(i, j)
  Next
  Next
'Проверяем выбран ли один из элементов
'управления Флажок
  If CheckBox1 = False And CheckBox2 = False Then
    MsgBox "Поднимите флажок:какой максимальный " _
& "период непрерывного увеличения добычи нефти " _
& "нужно найти-первый или последний."
    Exit Sub
  End If
'переменная f будет фиксировать факт наличия периода
'нарастания хотя бы у одной компании
'k — счетчик строк на рабочем листе при выводе данных
  f = 0
  k = 1
'j —счетчик внешнего цикла для перебора компаний
'i — счетчик внутреннего цикла для перебора номера года
'l — текущая длительность периода для данной компании
'MaxL — максимальная длина периода увеличения
'добычи нефти для текущей компании.
  For j = 1 To n
   l = 0
   MaxL = 0
   For i = 2 To m
   If DN(i, j) > DN(i - 1, j) Then
      l = l + 1: f = 1
   Else
      l = 0
   End If
'fl — переменная типа Boolean,
'принимающая значения либо ложь либо истина.
'В зависимости от выбора флажка проверяется истинность
'различных неравенств.
   If CheckBox1 = True Then
         fl = (l > MaxL)
         st1 = "Первый"
   Else
        fl = (l >= MaxL)
        st1 = "Последний"
   End If
'Если длина текущего периода оказалась больше чем MaxL,
'переопределяем значение переменной MaxL и вычисляем
```

```
'последний год периода возрастания добычи
       If fl Then MaxL = l: fin = a(m) - m + i
  Next
   If MaxL > 0 Then
'Выводим результаты в рабочий лист,
'если нашелся период возрастания
'в столбец В рабочего листа выводим начало периода возрастания
     With Sheets("Результат")
        .Cells(k + 2, 1) = Naz(j)
        .Cells(k + 2, 2) = fin - MaxL
        .Cells(k + 2, 3) = fin
     End With
     k = k + 1
   End If
 Next
 If f = 0 Then MsgBox "Таких компаний нет"
Else
  MsgBox " Неправильно определены диапазоны"
  Exit Sub
End If
'Управление видимостью элементов управления
CommandButton1.Visible = False
CommandButton2.Visible = True
Sheets("Результат").Activate
'Создание названия и заголовка таблицы результатов
 st2 = rng1.Cells(1, 1)
 st3 = rng1.Cells(m, 1)
 st = st1 & "максимальный период увеличения добычи " _
 & "нефти за" & st2 & "-" & st3 & "г."
With Sheets("Результат")
    .Cells(1, 1) = st
    .Cells(2, 1) = "Имя компании"
    .Cells(2, 2) = "Начало периода"
    .Cells(2, 3) = "Конец периода"
End With
Exit Sub
'Обработка ошибок
'Окно сообщения формируется в зависимости от кода ошибки
ErrorHandler:
Select Case Err.Number
  Case 13
   MsgBox "Неправильно введены числовые данные", 48
   Exit Sub
```

```
    Case 56
     MsgBox "Ошибка ввода/вывода", 56
     Exit Sub
    Case 1004
     MsgBox "Заполните правильно адреса диапазонов", 1004
     Exit Sub
    Case 9
    MsgBox "Создайте лист Результат", 9
    Unload UserForm1
    Case else
    MsgBox "Непредвиденная ошибка"
  End Select
End Sub

Private Sub CommandButton2_Click()
'Выгрузка формы из оперативной памяти
Unload UserForm1
End Sub

Private Sub UserForm_Initialize()
'При открытии формы очищаем диапазон,
'содержащий таблицу результатов
Sheets("Результат").Range("A:C").Clear
End Sub
```

3.5. Выполнить команду **Debug/Compile VBA Project**.

4. Процесс создания диалогового окна и процедур, связанных с ним, завершен. **Не забудьте удалить символ «'» в начале 10 строки.** Для того чтобы проверить, как работает созданная программа, нажмите кнопку F5. Создайте различные ситуации, приводящие к вызову оператора **On Error**.

5. Создайте на рабочем листе кнопку для запуска разработанной формы.

Примеры вопросов теста по программированию

1. Определить какую задачу можно решить с помощью процедуры PRY.

```
Public Sub PRY()
   kol = 0: m = 5000000000
   For i = 1 To 10
      x = InputBox("Введите число")
      If x = m Then kol = kol + 1
      If  m>x Then
                m = x
                kol = 1
      End If
   Next
   MsgBox "m=" & CStr(m) & " Таких чисел " & CStr(kol)
End Sub
```

1) Определить максимальное значение и количество чисел равных максимуму в последовательности вводимых чисел.
2) Определить минимальное значение в последовательности вводимых чисел и максимальный индекс числа равного минимуму.
3) Определить максимальное значение в последовательности вводимых чисел и максимальный индекс числа равного максимуму.
4) Определить минимальное значение и количество чисел равных минимуму в последовательности вводимых чисел.
5) Нет правильного варианта ответа.

2. В программе, записанной на объектно-ориентированном языке программирования Visual Basic, определить, что является свойством объекта.

```
Private Sub CommandButton1_Click()
    Label1.Visible = True
    Sheets(1).Activate
End Sub
```

1)CommandButton1_Click 2)Range("A1") 3)True 4)Activate 5)Visible

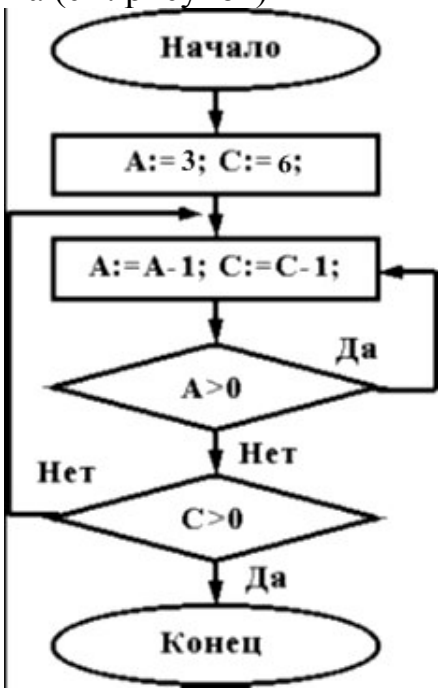
3. После выполнения фрагмента программы

```
A = Array(-6, 2, 4, 0, -1, 3)
For n = 1 To 3
    For m = n + 1 To 4
        If A(n) < A(m) Then
            c = A(n): A(n) = A(m): A(m) = c
        End If
    Next
Next
```

порядок элементов в массиве А будет следующий

1) -6, -1, 0, 2, 3, 4 2) 4, 3, 2, 0, -1, -6 3) -6, 4, 2, 0, -1, 3
4) 3, -1, 0, 2, 4, -6 5) 3, 4, 2, 0, -1, -6

4. Определить выходные значения переменных A и C после выполнения алгоритма (см. рисунок)



1) -2,3 2) 0, 3 3) -1, 2 4) 1,4 5) зацикливание

5. В какой строке приведенного фрагмента программного кода содержится ошибка?

```
10  Dim a(50) As Integer
20  For i = 1 To 50
30     a(i) = Range(i, 1)
40  Next
```

6. Фрагмент программы, в котором значения двумерного массива задаются с помощью вложенных циклов, записан в VB. Определите, какое значение приобретет элемент массива A[2,3].

```
For n=1 to 5
     For k=1 to 5
          If  n>=k and n+2<5 Then A(n,k)=n+k^2 Else
     A(n,k)=n*k^2
     Next k
Next n
```

1)-5 2)0 3)18 4)11

Контрольная работа

Контрольная работа выполняется в Excel с использованием средств VBA. Рабочая книга: Контрольная_работа_ФамилияСтудента. Формат имени рабочих листов: Задача<номер задачи>.

1. Запишите на VBA следующие арифметические выражения. Напишите программы для вычисления значений арифметических выражений (для ввода и вывода значений переменных используйте диалоговые окна).

1. $\dfrac{a+\sin^3 b^2}{\left|\cos 25+ctg\,60\right|}\log_2 x^2$

2. $\dfrac{(x^8+8^x)^{\frac{1}{8}}(a^2-(b+\sqrt[3]{ctgx})}{3-\dfrac{4}{\sqrt{x}-\sin^2 a^3}}$

3. $\dfrac{xyz-3{,}3\left|x-\sqrt[4]{|y|}\right|}{10^7+\ln\sin^2 x}$

4. $\dfrac{(b+\cos^2 y^4)(ab+tg(x+\sqrt[3]{y^2})}{\cos 11^0+\left|ctg(y)\right|}$

5. $\dfrac{\sqrt{\left|\dfrac{xb}{a}\right|+\cos^2(x+b)^3}}{\dfrac{x^2(x+1)}{b}-\sin^2(x+a)}$

6. $\dfrac{\sin^3(x^2+a)^2-\sqrt[3]{\dfrac{x}{b}}}{\dfrac{x^2}{a}+\cos^3(x+b)}$

7. $\dfrac{tg(t)+\left|\sqrt[3]{3\sin(t)}\right|}{\cos(bt\sin^2 t)+2tgt}$

8. $btg^2 x-\dfrac{a}{\sin^2\dfrac{x}{a}}+\dfrac{ae^{-\sqrt{a}}}{\cos(\dfrac{bx}{a})}$

9. $\dfrac{\cos^3(x^2+b)^2-\sqrt[5]{\dfrac{x}{b}}}{x+e^{3^{(x+b)}}}$

10. $\dfrac{\sqrt{\dfrac{\cos^2(x+b)^3+x^4}{|a|}}}{e^{x^2}-\sin^2(x+a)}$

2. Запишите на VBA следующие логические выражения. Напишите программы для вычисления значений логических выражений.

1. (2 *2 = 4 или 3 * 3 = 10) $\Rightarrow$ (2 * 2 = 5 или 3 * 3 = 9)

2. (2 * 2 = 4 и 3 * 3 = 10) $\Leftrightarrow$ (2 * 2 = 5 и 3 * 3 = 9)

3. (2 * 2 = 4 или 3 * 3 = 10) $\Rightarrow$ (2 * 2 = 5 и 3 * 3 = 9)

4. (2 * 2 = 4 и 3 * 3 = 10) $\Leftrightarrow$ (2 * 2 = 5 или 3 * 3 = 9)

5. (2 * 2 = 4 или 3 * 3 = 10) $\Rightarrow$ (2 * 2 = 5 или 3 * 3 = 9)

6. (2 * 2 = 4 и 3 * 3 = 10) $\Leftrightarrow$ (2 * 2 = 5 и 3 * 3 = 9)

7. (2 * 2 > 4 или 3 * 3 = 9) $\rightarrow$ (2 * 2 = 5 и 3 * 3 = 9)

8. (2 * 2 ≤ 4 и 3 * 3 = 9) ⇔ (2 * 2 = 5 или 3 * 3 = 9)

9. (2 * 2 = 4 и 3 * 3 = 9) ⇒ (2 * 2 = 5 или 3 * 3 ≤ 9)

10. (2 * 2 = 5 или 3 * 3 = 9) ⇔ (2 * 2 = 4 и 3 * 3 ≥ 9)

3. Создать программу для вычисления значения функции g.

1. $g = \begin{cases} \dfrac{1+X^2}{\sqrt{1+X^4}}, X \le 0, \\ 2X + \dfrac{\sin^2}{2+X}, X > 0, \end{cases}$

2. $g = \begin{cases} 3\sin(x) - \cos^2(x), x \le 0, \\ 3\sqrt{1+x^2}, x \ge 1. \end{cases}$

3. $g = \begin{cases} \dfrac{3x^2}{1+x^2}, x \le 0 \\ \sqrt{1+\dfrac{2x}{1+x^2}}, X > 0, \end{cases}$

4. $g = \begin{cases} \dfrac{3+\sin^2(2x)}{1+\cos^2(x)}, x \le 0, \\ 2\sqrt{1+2x}, x > 0, \end{cases}$

5. $g = \begin{cases} \sqrt{1+x^2}, x \le 0, \\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0,2x}}}, x > 0, \end{cases}$

6. $g = \begin{cases} \sqrt{1+|x|}, x \le 0, \\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}, x > 0, \end{cases}$

7. $g = \begin{cases} \dfrac{\sqrt{1+|x|}}{2+|x|}, x \le 0, \\ \dfrac{1+x}{2+\cos^3(x)}, x > 0, \end{cases}$

8. $g = \begin{cases} \sqrt[3]{1+x^2}, x \le 0, \\ \sin^2 x + \dfrac{1+x}{1+\cos^2(x)}, x > 0, \end{cases}$

9. $g = \begin{cases} \dfrac{\sqrt{1+|x|}}{2+|x|}, x \le 0, \\ \dfrac{1+x}{2+\cos^3(x)}, x > 0, \end{cases}$

10. $g = \begin{cases} \sqrt{1+2x^2-\sin^2(x)}, x \le 0, \\ \dfrac{2+x}{\sqrt[3]{2+e^{-0,1X}}}, x > 0, \end{cases}$

4. Создать функцию пользователя для начисления премии. Премия агенту по продажам начисляется по следующему правилу: если продукции продано меньше чем на C_1 руб., то премия составляет i_1% от стоимости реализованной продукции; если продукции продано не меньше, чем на C_1 руб., но меньше, чем на C_2 руб., то премия составляет i_2%, и т.д.; если продукции продано не меньше, чем на C_k руб., то премия составляет i_{k+1}%.

Используя созданную функцию пользователя, создать таблицу начисления премии для некоторого агента за n месяцев.

Сделать отчет: создать документ, содержащий постановку задачи, блок-схему решения задачи, листинг программы.

Вариант	n	C_1	C_2	C_3	i_1	i_2	i_3	i_4
1	12	10000	20000		1	1,5	2,3	
2	11	20000	40000		2	3,0	4,5	
3	10	30000	60000	120000	3	4,5	6,8	7,4
4	9	40000	80 000	160000	4	6,0	9,0	9,9
5	12	50000	100000	200 000	5	7,5	11,3	12,4
6	11	60000	120000	240 000	6	9,0	13,5	14,9
7	10	70000	140000	280 000	1	1,5	2,3	2,5
8	9	80000	160000	320 000	2	3,0	4,5	5,0
9	12	90000	180000	360 000	3	4,5	6,8	7,4
0	11	100000	200 000		4	6,0	9,0	

5. Сгенерировать с помощью формулы $f = a + (b - a) * СЛЧИС()$ на рабочем листе доход фирмы в некоторых условных единицах, где a = -500, b=600 за 2000, 2001... 2009. Номер года должен содержаться в столбце A, доход — в столбце B.

Составить программу для решения следующей задачи.

Вариант	Задача
1	Найти сколько лет фирма имела убытки. Определить средний убыток, если он был.
2	Вычислить сумму убытков фирмы. Определить максимальный убыток, если он был.
3	Найти сколько лет фирма имела прибыль. Определить максимальную прибыль, если она была
4	Вычислить сумму прибыли фирмы. Определить минимальную прибыль, если она была.
5	Вычислить суммы прибылей и убытков фирмы. Когда прибыль была максимальной?
6	Вычислить сумму убытков фирмы. Когда убытки были максимальными?
7	Вычислить суммы прибылей и убытков за первые семь лет работы. Найти наибольший доход за этот период.
8	Вычислить сумму убытков фирмы и определить, сколько лет фирма несла убытки? В каком году убыток был максимальным?
9	Вычислить средние арифметические всех прибылей и убытков фирмы. В каком году фирма имела наименьшую прибыль?
0	Вычислить средний доход фирмы. В каком году доход был наибольшими?

Сделать отчет: создать документ, содержащий постановку задачи, блок-схему решения задачи, листинг программы, распечатку формы.

6. Создать пользовательскую форму, имеющую следующие элементы управления: список, два переключателя, рамку, метку, текстовое поле, две кнопки. Построить диалог согласно таблице. Список заполнить программно по формуле $f = A + (B - A) * rnd()$

Вариант	A	B	Задача при выборе первого переключателя	Задача при выборе второго переключателя
1	-4	4	Найти количество выбранных в списке чисел	Найти среднее геометрическое выбранных в списке чисел
2	1	9	Найти количество отрицательных чисел среди выбранных в списке	Найти сумму квадратов отклонений выбранных в списке чисел от среднего
3	-1	5	Найти количество положительных чисел среди выбранных в списке	Найти среднее арифметическое положительных чисел среди выбранных в списке
4	-4	8	Найти количество отрицательных чисел среди выбранных в списке	Найти среднее арифметическое отрицательных чисел среди выбранных в списке
5	-3	7	Найти среднее геометрическое выбранных в списке чисел	Найти количество выбранных в списке чисел
6	5	10	Найти сумму квадратов выбранных чисел	Найти произведение выбранных чисел
7	-6	12	Найти количество положительных чисел среди выбранных в списке	Найти сумму квадратов отклонений выбранных в списке чисел от среднего
8	-2	5	Найти первый положительный элемент среди выбранных в списке	Найти среднее арифметическое положительных чисел среди выбранных в списке
9	-4	7	Найти первый отрицательный элемент среди выбранных в списке	Найти среднее арифметическое отрицательных чисел среди выбранных в списке
10	3	9	Найти сумму квадратов отклонений выбранных	Найти количество выбранных чисел в списке

			чисел в списке от среднего	

Сделать отчет: создать документ, содержащий постановку задачи, блок-схему решения задачи, листинг программы, распечатку формы.

7. Реализовать решение задачи в среде VBA. Ввод и вывод данных выполнять в ячейки рабочего листа.

Вариант	Задача
1	В каждой строке матрицы A(n,m), имеющей нечетный номер, найти максимальный элемент.
2	В матрице A(n,m) найти номер последнего из ее столбцов, имеющего равное количество положительных и отрицательных элементов. Если таких столбцов нет, то вывести 0.
3	В каждой строке матрицы A(n,m), имеющей четный номер, найти минимальный элемент.
4	В каждой строке матрицы A(n,m) найти количество элементов меньше среднего арифметического всех элементов этой строки.
5	В каждом столбце матрицы A(n,m), имеющем нечетный номер, найти максимальный элемент.
6	В каждом столбце матрицы A(n,m) найти количество элементов меньше среднего арифметического всех элементов этого столбца..
7	В каждом столбце матрицы A(n,m), имеющем четный номер, найти минимальный элемент.
8	В матрице A(n,m) найти максимальный среди минимальных элементов ее столбцов.
9	В каждой строке матрицы A(n,m), найти произведение суммы положительных и суммы отрицательных элементов.
0	В матрице A(n,m) найти минимальный среди максимальных элементов ее строк.

8. * Выполнить формализацию задачи. Разработать алгоритм решения. Написать и отладить программу, реализующую алгоритм. Входные и выходные данные хранить в ячейках различных рабочих листов Excel. Спроектировать пользовательский интерфейс с использованием элементов управления: RefEdit, кнопка, рамка, метка. Остальные элементы управления использовать по необходимости. Запрограммировать перехват ошибок. Исходные данные сгенерировать с помощью датчика случайных чисел. Сделать отчет.

Вариант	Задача
1	Известен поквартальный выпуск продукции для A цехов. Вывести среднее арифметическое выпуска по всем цехам и определить цеха, выпустившие продукции больше среднего значения по всем

Вариант	Задача
	цехам.
2	Известны годовые объемы добычи нефти каждой из A компаний за последние B лет. Для каждой компании определить года, в течение которых было добыто максимальное количество нефти.
3	Известны очки, полученные каждым из B спортсменов-многоборцев в каждом из A видов соревнований. Для каждого спортсмена определить, в каких видах соревнований он получил результат не хуже прочих и каков этот результат.
4	Известна сумма, на которую в течении B дней каждый из A филиалов фирмы продал товар. Определить лидеров продаж за весь зарегистрированный период (первые три места).
5	Ежедневно в течение апреля измеряли уровень шума в каждом из B районов города. Найти все районы, в которых наблюдался максимальный уровень, число месяца, когда наблюдался этот уровень.
6	Известны объемы выпуска продукции каждым из A филиалов предприятия за каждый год из последних B. Найти номера филиалов, ежегодно увеличивающих объем выпуска продукции.
7	Каждому из C опрошенных граждан предлагали выбрать вероятного кандидата в президенты из предложенного списка A политиков. Выбор фиксировался следующим образом: рядом с номером выбранного кандидата ставилась 1, рядом с невыбранным 0. Определить кандидатов, имеющих наибольший и наименьший рейтинг.
8	Известны результаты встреч на шахматном турнире для C шахматистов. Результаты встречи каждой пары участников турнира оценивался следующим образом: 2 - победа, 1 - ничья, 0 - поражение. Определить номера участников турнира, набравших наибольшее число очков.
9	Известны результаты встреч на шахматном турнире для C шахматистов. Результаты встречи каждой пары участников турнира оценивался следующим образом: 2 - победа, 1 - ничья, 0 - поражение. Определить сумму очков, которую набрал шахматист с заданным номером, а также номера других участников, набравших такое же количество очков.
0	Известен объем сбыта продукции для A компаний за B лет. Определить доля какой компании уменьшилась на рынке за зарегистрированный период.

Литература

1. Гарнаев, А.Ю. Excel, VBA, Internet в экономике и финансах / А.Ю. Гарнаев. - М.: БХВ-Петербург, 2005. - 826 с.
2. Кузьменко, В. Г. VBA. Эффективное использование / В.Г. Кузьменко. - М.: Бином-Пресс, 2012. - 624 с.
3. Уокенбах, Джон Excel 2013. Профессиональное программирование на VBA / Джон Уокенбах. - М.: Вильямс, 2014. - 960 с.

Приложение 1

Отображение вкладки Разработчик в Microsoft Office 2016

1. На вкладке файл выберите Параметры> Настройка ленты.
2. В разделе Настройка ленты в списке Основные вкладки установите флажок Разработчик.