МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

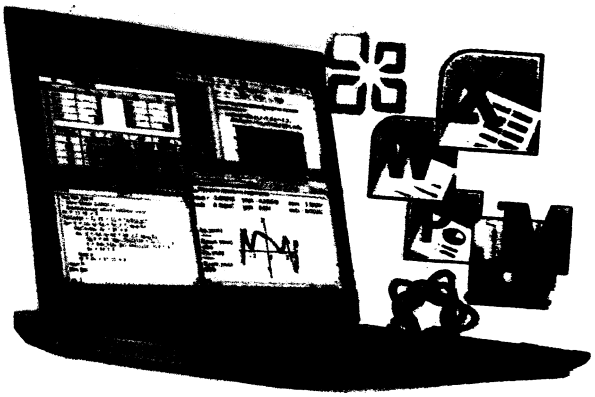
по дисциплине «Информатика» (II семестр)

для студентов специальности

«Машины и аппараты пищевых производств»

машиностроительного факультета

дневной формы обучения



Группа _____

Фамилия _____

Имя _____

Отчество _____

Вариант _____

м/тел. +375 (___) ____-___-___

Вариант № _____

Брест 2018

УДК 004

Практикум предназначен для студентов специальности 1 – 36 09 01 «Машины и аппараты пищевых производств» первого курса машиностроительного факультета, изучающих дисциплину «Информатика». В него входят задания для лабораторных работ и задания для их защиты. В теоретической части изложены методические рекомендации по работе с электронными таблицами MS EXCEL с поддержкой VBA и системой компьютерной алгебры MathCAD.

Составители: В.А. Кофанов, к.т.н., доцент
И.М. Гучко, ст. преподаватель
Л.К. Рамская, ст. преподаватель

Рецензенты: зав. кафедрой прикладной математики и технологии программирования БрГУ имени А.С. Пушкина, доцент, к.ф.-м.н. О.В. Матысик;

директор Брестского филиала ИООО EPAM Systems, доцент, к.ф.-м.н. С.А. Тузик.

Общие указания

Практикум предназначен для организации самостоятельной и лабораторной работы студентов в II семестре при изучении дисциплины «Информатика».

Перед началом работы необходимо привести информацию на титульном листе, то есть должны быть указаны данные ее владельца.

В специальном блоке «Условие» на странице с лабораторной работой переписывается условие задачи согласно варианта из электронного файла задания. При выполнении лабораторных работ результаты вычислений заносятся в отчет в том же виде, в котором они отображаются на экране монитора. Ведение записей выполняется четко и разборчиво шариковой ручкой. Неправильные (ошибочные) записи на страницах практикума необходимо исправлять с использованием корректирующих средств.

Каждая лабораторная работа считается выполненной только при наличии отметки преподавателя о ее защите, подтвержденной его подписью (личной печатью) на данной странице. Данные из этого листа служат основанием для допуска к экзамену по дисциплине.

Отметки о защите лабораторных работ

Наименование работы	***Дата***	***Защита***	***Примечание***
Лабораторная работа № 1			
Лабораторная работа № 2			
Лабораторная работа № 3			
Лабораторная работа № 4			
Лабораторная работа № 5			
Лабораторная работа № 6			

Допуск к экзамену: _____ _____

(дата) (подпись)

Методические указания к выполнению лабораторных работ

ЭЛЕКТРОННАЯ ТАБЛИЦА MICROSOFT EXCEL

Логическая функция ЕСЛИ (категория Логические)

Функция **«ЕСЛИ»** используется при проверке условий для значений и формул. Синтаксис этой функции: ЕСЛИ(*лог_выражение*; *значение_если_истина*; *значение_если_ложь*). Первый аргумент функции «ЕСЛИ» должен содержать логическое выражение. Второй и третий аргументы содержат выражение. В случае если логическое выражение истинно, то результат функции «ЕСЛИ» будет соответствовать ее второму аргументу. Если логическое выражение ложно, то результат функции «ЕСЛИ» будет соответствовать ее третьему аргументу. Пара примеров использования функции «ЕСЛИ» показана в таблице 1.

Таблица 1 – Примеры создания разветвляющихся функций с помощью встроенной функции «ЕСЛИ»

Разветвляющаяся функция	Формула в ячейке B1
$f(x)=\begin{cases}2\cdot x, & x<0\\ x^2, & x\geq 0\end{cases}$	A B 1 5 =ЕСЛИ(A1<0;2*A1;A1^2) да нет
$f(x)=\begin{cases}2\cdot x, & x<0\\ x^2, & 0\leq x\leq 3\\ e^{2\cdot x}, & x>3\end{cases}$	=ЕСЛИ(A1<0;2*A1;ЕСЛИ(A1<=3;A1^2;EXP(2*A1))) да да нет нет

Использование формы для ввода данных в таблицу

При вводе данных в таблицу Excel (пример такой таблицы представлен на рис. 1) можно использовать **форму** ввода данных – диалоговое окно, которое автоматически создается после определения заголовка списка.

С помощью формы можно также осуществлять поиск и редактирование записей, которые удовлетворяют простому или множественному критерию сравнения, в котором условия могут соответствовать логической операции **И**.

	A	B	C	D	E	F
1	***Наименование товара***	***Наименование магазина***	***Дата реализации***	***Кол-во***	***Цена***	***Выручка***
2	Газовая плита	ЦУМ	10-май-2010	5	100 000р.	500 000р.
3	Микроволновая печь	1000 мелочей	12-май-2010	8	150 000р.	1 200 000р.
35	Варочная панель	тд Немига	10-май-2010	1	150 000р.	150 000р.
36	Электрическая плита	ЦУМ	15-май-2010	9	150 000р.	1 350 000р.
37	Соковыжималка	ЦУМ	30-янв-2012	3	230 000р.	690 000р.

Рисунок 1 – Таблица исходных данных

Для вызова окна "Форма" (рис. 2) необходимо поместить указатель ячейки в любое место внутри списка и выбрать команду **Данные / Форма** (для Excel **2003**). В Excel **2007+** команды Форма нет на ленте, поэтому её надо добавить на панель быстрого доступа, для чего:

1. щелкнуть правой кнопкой мыши на панели быстрого доступа и выбрать в контекстном меню пункт **Настройка панели быстрого доступа.** На экране появится раздел ***Панель быстрого доступа*** диалогового окна "Параметры Excel".
2. в раскрывающемся списке **Выбрать команды из** необходимо выбрать **Команды не на ленте.** В списке ниже выбрать **Форма**, а затем нажать кнопку **Добавить**.
3. нажать кнопку **ОК**, чтобы закрыть диалоговое окно "Параметры Excel".

После этого панель быстрого доступа будет включать новый значок .

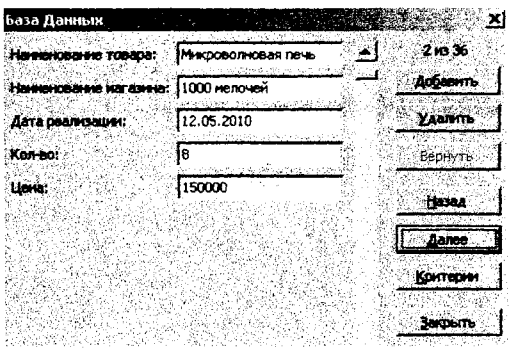


Рисунок 2 – Диалоговое окно **Форма**

Сортировка таблицы

Сортировка переупорядочивает строки таблицы на основе значений из одного столбца. Для реализации сортировки по выбранному столбцу или множественной сортировки необходимо воспользоваться диалоговым окном **Сортировка** (2003 – **Данные / Сортировка**; 2007+ – **Главная / Редактирование / Сортировка и фильтр / Настраиваемая сортировка**).

В диалоговом окне **Сортировка** надо указать, по какому столбцу будет идти сортировка и в каком порядке (возрастающем или убывающем). При задании множественной сортировки необходимо вначале указать столбец, по которому будет выполняться первоначальная сортировка, затем столбец, в котором сортировка будет выполняться в случае совпадения значений в предыдущем столбце и т.д.

Например, отсортируем таблицу по названию магазина в алфавитном порядке, а для одинаковых магазинов отсортируем их по выручке в возрастающем порядке. Для этого выделим любую ячейке в таблице, затем откроем и заполним диалоговое окно **Сортировка** как показано на рисунке 3.

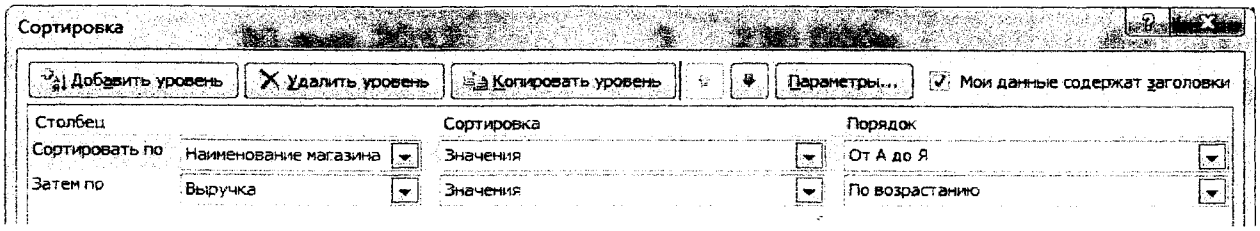


Рисунок 3 – Диалоговое окно **Сортировка** для выполнения множественной сортировки (2007+)

Результат множественной сортировки показан на рисунке 4.

	A	B	C	D	E	F
1	***Наименование товара***	***Наименование магазина***	***Дата реализации***	***Кол-во***	***Цена***	***Выручка***
2	Духовой шкаф	1000 мелочей	11-май-2010	2	150 000р.	300 000р.
3	Портативная газ/плита	1000 мелочей	15-май-2010	5	100 000р.	500 000р.
35	Духовой шкаф	ЦУМ	15-май-2010	9	110 000р.	990 000р.
36	Электрическая плита	ЦУМ	12-май-2010	9	140 000р.	1 260 000р.
37	Электрическая плита	ЦУМ	15-май-2010	9	150 000р.	1 350 000р.

Рисунок 4 – Результат множественной сортировки

Промежуточные итоги

С помощью команды **Промежуточные итоги** можно автоматически подсчитать промежуточные и общие итоги в списке для столбца таблицы. Перед выполнением этой команды необходимо обязательно выполнить сортировку по тому столбцу, по которому подводятся итоги.

Параметры промежуточных итогов задаются в диалоговом окне **Промежуточные итоги** (2003 – **Данные / Итоги**; 2007+ – **Данные / Структура / Промежуточные итоги**).

В случае *если необходимо определить сумму выручки по каждому магазину,* то предварительно выполняется сортировка как показано в предыдущем пункте. Затем для этой таблицы открывается диалоговое окно **Промежуточные итоги** (рис. 5). В диалоговом окне:

- ✓ в поле **При каждом изменении в** выбрать столбец для подсчета итогов – **Наименование магазина;**
- ✓ в поле **Операция** выбрать итоговую функцию для вычисления промежуточных итогов – **Сумма;**
- ✓ в поле **Добавить итоги по** установить флажок для каждого столбца, содержащего значения, по которым необходимо подвести итоги – **Выручка**.

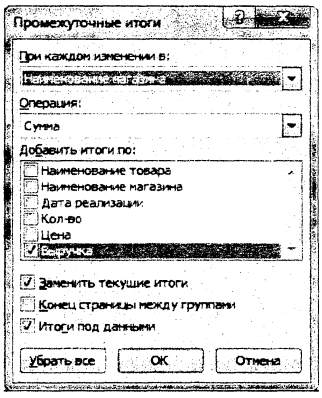


Рисунок 5 – Диалоговое окно **Промежуточные итоги**

При необходимости команду **Промежуточные итоги** можно использовать снова, чтобы добавить дополнительные строки итогов с использованием других функций. Во избежание перезаписи имеющихся итогов снимите флажок **Заменить текущие итоги**. Если наоборот необходимо заменить предыдущие итоги, то устанавливаем флажок **Заменить текущие итоги**.

Чтобы расположить итоговую строку под строкой данных, установить флажок **Итоги под данными**.

Результат выполнения настроек диалогового **Промежуточные итоги** показан на рисунке 6.

	A	B	C	D	E	F
1	Наименование товара	Наименование магазина	Дата реализации	Кол-во	Цена	Выручка
10		1000 мелочей Итог				5 920 000р.
11	Духовой шкаф	тд Граффи	14-май-2010	1	120 000р.	120 000р.
12	Портативная газ/плита	тд Граффи	12-май-2010	3	140 000р.	420 000р.
13	Газовая плита	тд Граффи	13-май-2010	4	110 000р.	440 000р.
14	Портативная газ/плита	тд Граффи	11-май-2010	5	150 000р.	750 000р.
15	Духовой шкаф	тд Граффи	13-май-2010	8	110 000р.	880 000р.
16	Газовая плита	тд Граффи	12-май-2010	6	150 000р.	900 000р.
17	Микроволновая печь	тд Граффи	12-май-2010	9	150 000р.	1 350 000р.
18		тд Граффи Итог				4 860 000р.
27		тд Немига Итог				5 110 000р.
41		ЦУМ Итог				9 540 000р.
42		Общий итог				25 430 000р.

Рисунок 6 – Результат подведения промежуточных итогов

Автофильтрация таблицы

Фильтрация таблицы означает отображение в таблице только тех строк, значения в которых удовлетворяют определенным условиям. Включить автофильтрацию можно выбрав соответствующую команду (2003 – **Данные** / **Фильтр** / **Автофильтр**; 2007+ – **Данные** / **Сортировка и фильтр** / **Фильтр**). Справа от заголовков (в шапке таблицы) столбцов появятся кнопки со стрелками автофильтра. Если щелкнуть стрелку автофильтра, отображается список различных вариантов фильтрации и сортировки по возрастанию и убыванию. При выполнении фильтрации стрелка активного автофильтра окрашивается в синий цвет.

Для того чтобы после фильтрации отобразились все записи необходимо нажать активную кнопку и выбрать команду (2003 – **Все**; 2007+ – **Снять фильтр с «...»**) или выполнить команду (2003 – **Данные** / **Фильтр** / **Отобразить все**; 2007+ – **Данные** / **Сортировка и фильтр** / **Очистить**), а для удаления автофильтра необходимо снять флажок команды **Данные** / **Фильтр** / **Автофильтр** (2003) либо отключить его **Данные** / **Сортировка и фильтр** / **Фильтр** (2007+).

На примере исходной таблицы (рис. 1) предположим, что необходимо *отобразить данные только по магазину ЦУМ.* Для этого необходимо открыть раскрывающийся список в заголовке столбца **Наименование магазина** и установить в нем сначала флажок напротив слова **ЦУМ**. При этом, на кнопке раскрытия списка заголовка столбца **Наименование магазина** появится значок, который означает, что таблица отфильтрована по значениям этого столбца (рис. 7).

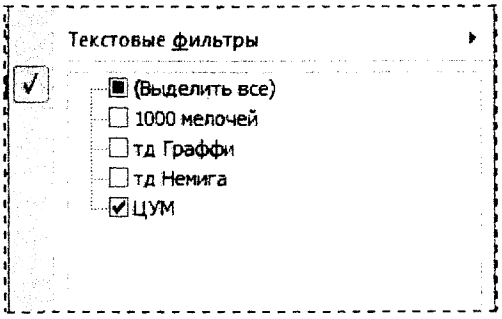


Рисунок 7 – Раскрывающийся список в заголовке столбца **Наименование магазина**

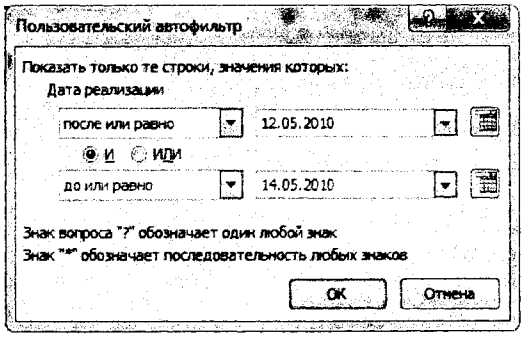


Рисунок 8 – Пользовательский фильтр в столбце **Дата реализации**

Большие возможности для фильтрации таблиц предоставляют находящиеся в раскрывающихся списках заголовков столбцов команды **Текстовые фильтры** (если в столбце записаны текстовые данные), **Числовые фильтры** (если в столбце хранятся числовые данные) и **Фильтры по дате** (если значения в столбце отформатированы одним из форматов даты или времени).

Для фильтрации данных можно использовать любое количество столбцов таблицы. Например, в исходной таблице (рис. 1) *можно использовать фильтр, когда в столбце* ***Наименование магазина*** *задан магазин* ***ЦУМ*** (рис. 7), *а в столбце* ***Дата реализации*** *– диапазон с 12.05.2010 по 14.05.2010* (рис. 8).

Результат применения автофильтра по двум столбцам таблицы показан на рисунке 9.

	A	B	C	D	E	F
1	Наименование товара	Наименование магазина	Дата реализации	Кол-во	Цена	Выручка
13	Варочная панель	ЦУМ	13-май-2010	6	110 000р.	660 000р.
17	Портативная газ/плита	ЦУМ	13-май-2010	2	150 000р.	300 000р.
20	Микроволновая печь	ЦУМ	12-май-2010	4	140 000р.	560 000р.
26	Варочная панель	ЦУМ	12-май-2010	1	120 000р.	120 000р.
29	Электрическая плита	ЦУМ	12-май-2010	9	140 000р.	1 260 000р.
31	Газовая плита	ЦУМ	12-май-2010	6	140 000р.	840 000р.

Рисунок 9 – Результат фильтрации по нескольким столбцам

Расширенный фильтр

Команда «Расширенный фильтр» используется для фильтрации по более сложным условиям отбора записей, чем автофильтр, например, по нескольким условиям отбора в одном столбце, по нескольким условиям отбора в нескольких столбцах или для отбора записей по условиям отбора с помощью формулы.

Перед применением команды «Расширенный фильтр» необходимо подготовить условия отбора, которые, как правило, располагаются в отдельных от таблицы ячейках.

Создание диапазона условий отбора следующее:

- ✓в выбранной свободной строке заполнить ячейки заголовками полей, по значениям которых будет происходить фильтрация;
- ✓строки ниже заполнить значениями, которые и будут являться критериями расширенного фильтра.

Например, критерий для расширенного фильтра, *когда в столбце* ***Наименование магазина*** *задан магазин* ***ЦУМ****, а в столбце* ***Дата реализации*** *– диапазон с 12.05.2010 по 14.05.2010,* будет выглядеть так, как показано на рисунке 10.

	A	B	C
40	***Наименование магазина***	***Дата реализации***	***Дата реализации***
41	ЦУМ	>=12.05.2010	<=14.05.2010

Рисунок 10 – Критерий для расширенного фильтра

После того как условие отбора подготовлено необходимо выполнить следующие действия:

- ✓выделить любую ячейку в таблице;
- ✓открыть окно **Расширенный фильтр** (2003 – **Данные** / **Фильтр** / **Расширенный фильтр**; 2007+ – **Данные** / **Сортировка и фильтр** / **Дополнительно**) – рисунок11;
- ✓установить переключатель **Обработка** в положение **Скопировать результаты в другое место**, если необходимо, чтобы Excel выводил результаты фильтрации рядом с таблицей и указать диапазон или ячейку для начала размещения результатов в поле **Поместить результат в диапазон**;
- ✓проверить исходный диапазон ячеек фильтруемой таблицы, диапазон должен содержать все ячейки списка с учетом ячеек заголовков столбцов;
- ✓указать диапазон критериев, он должен содержать все ячейки диапазона условий отбора (рис. 11) с учетом ячеек заголовков столбцов;
- ✓нажать на кнопку **ОК** для выполнения фильтрации. Начиная с ячейки А44, будут отображены результаты фильтрации (рис. 9).

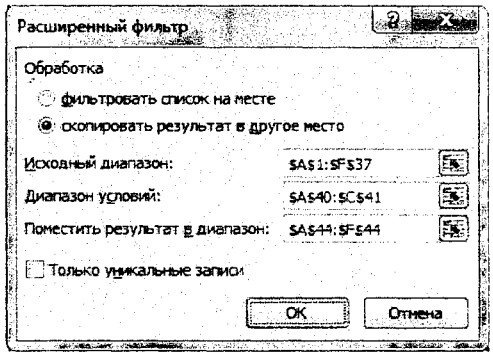


Рисунок 11 – Диалоговое окно **Расширенный фильтр**

При формировании критерия отбора расширенного фильтра возможны следующие варианты.

1. Критерий отбора содержит одно или несколько условий, накладываемых на один столбец (одно поле). Если критерий содержит несколько условий, то они связываются логической операцией **ИЛИ** (рисунок 12б).

2. Критерий отбора содержит несколько условий, накладываемых на несколько столбцов (полей) одновременно. Здесь возможны следующие варианты:

- ✓необходимо наложить несколько условий отбора на несколько столбцов, причем эти условия должны связываться логической операцией **ИЛИ**. Тогда условия отбора задаются в разных строках критерия (рис. 12в);
- ✓необходимо одновременно наложить несколько условий отбора на несколько полей, причем условия отбора должны быть связаны логической операцией **И**. Тогда все условия задаются в одной строке критерия (рис. 10, 12г);
- ✓необходимо несколько условий наложить на несколько полей, причем связываться они могут обеими логическими операциями **И** и **ИЛИ** (рис. 12а).

а)

Поле 1		***Поле 2***
усл. 1	и	усл. 2
или		
усл. 3	и	усл.4

б)

Поле 1
усл. 1
или
усл. 3

в)

Поле 1		***Поле 2***
усл. 1		
или		
		усл.4

г)

Поле 1		***Поле 2***
усл. 1	и	усл. 2

Рисунок 12 – Правила составления критериев для расширенного фильтра

3. Вычисляемый критерий. Условия отбора могут содержать формулу. Полученное в результате вычисления формулы значение будет участвовать в сравнении. Правила формирования вычисляемого критерия следующие:

- ✓ в диапазоне критерия нельзя указывать существующие в таблице имена полей. Следует ввести новое имя заголовка;

- ✓ при создании формул вычисляемых критериев следует использовать первую строку списка (не строку заголовков), т. е. первую ячейку в сравниваемом столбце;
- ✓ если в формуле используются ссылки на ячейки списка, они задаются как относительные;
- ✓ если в формуле используются ссылки на ячейки вне списка, они задаются как абсолютные;
- ✓ вычисляемые критерии можно сочетать с невычисляемыми;
- ✓ не следует обращать внимание на результат, выдаваемый формулой в области критерия (обычно ИСТИНА или ЛОЖЬ).

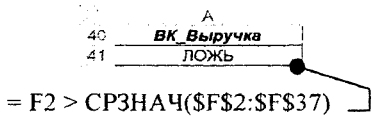


Рисунок 13 – Пример вычисляемого критерия

Для условия с вычисляемым критерием *приведем пример, в котором необходимо отыскать записи* (исходная таблица – рис. 1) *с выручкой выше средней по всему столбцу* (рис. 13).

Сводная таблица

Сводная таблица – это таблица итоговых данных, извлеченных или рассчитанных на основе информации, содержащейся на рабочем листе и организованной в виде таблицы.

Порядок создания сводных таблиц одинаков. Отличие только в ее настройках, которые зависят от конечной цели. Например, необходимо на основе исходных данных (рис. 1) сформировать сводную таблицу, *которая показала бы информацию о том, какова суммарная выручка от реализации каждого товара в каждом магазине и в качестве страницы в разрезе выбрать поле «Дата реализации».*

Для создания сводной таблицы в Excel 2003 необходимо выделить источник данных (таблицу с заголовками столбцов) и запустить инструмент **Мастер сводных таблиц**. Перед построением сводной таблицы из источника данных обязательно должны быть убраны промежуточные итоги и наложенные фильтры.

Запустить **Мастер сводных таблиц** можно путем выбора команды **Данные / Сводная таблица**. На экране появится первое диалоговое окно **Мастера сводных таблиц и диаграмм – шаг 1 из 3**.

- ✓ Шаг 1. Выбрать вариант создания таблицы на основе данных, находящихся в списке Microsoft Excel. Вид создаваемого отчета – сводная таблица.
- ✓ Шаг 2. Выбрать диапазон таблицы, если он не указан.
- ✓ Шаг 3. Выбрать будущее расположение сводной таблицы – новый лист (рис. 14). Однако перед тем как нажать кнопку **Готово**, надо нажать кнопку **Макет...** (рис. 15). Перетягиваем мышкой поле **Дата реализации** в отдельно стоящую область **Страница,** поле **Наименование товара** – в область **Строка**, поле **Наименование магазина** – в область **Столбец**, а поле **Выручка** – в поле **Данные**. Для поля **Выручка** автоматически выбирается тип операции – Сумма (т.к. формат данного поля – числовой).

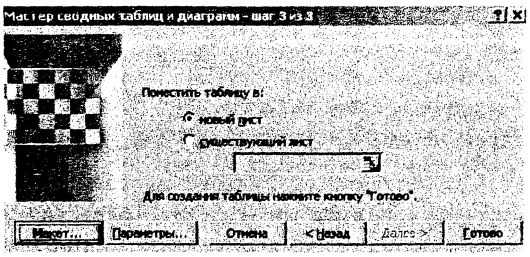


Рисунок 14 – Третий шаг построения сводной таблицы

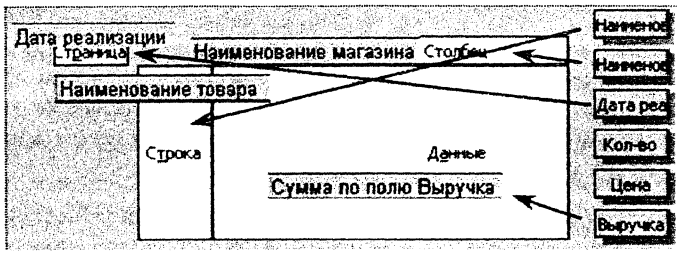


Рисунок 15 – Настройка областей сводной таблицы

По завершению третьего шага на новом листе получается результат, представленный на рисунке 16.

	A	B	C	D	E	F
1	Дата реализации	(Все)				
2						
3	Сумма по полю Выручка	Наименование магазина				
4	Наименование товара	1000 мелочей	тд Граффи	тд Немига	ЦУМ	Общий итог
5	Варочная панель			710000	1230000	1940000
6	Газовая плита		1340000	1350000	1340000	4030000
7	Духовой шкаф	1260000	1000000		990000	3250000
8	Микроволновая печь	2810000	1350000	450000	1400000	6010000
9	Портативная газ/плита	1850000	1170000	1350000	1280000	5650000
10	Соковыжималка				690000	690000
11	Электрическая плита			1250000	2610000	3860000
12	Общий итог	5920000	4860000	5110000	9540000	25430000

Рисунок 16 – Сводная таблица

Для создания сводной таблицы в Excel 2007+ необходимо выделить источник данных и выбрать команду **Вставка / Таблицы / Сводная таблица**. В диалоговом окне **Создание сводной таблицы** необходимо проверить правильность выбора источника данных и указать лист расположения сводной таблицы.

На новом листе сформируется пустой шаблон сводной таблицы, а справа окно для ее настройки (рис. 17).

Далее повторяются те же действия, что и Excel 2003. Перетягиваются мышкой поле **Наименование товара** в область **Названия строк**, поле **Наименование магазина** – в область **Наименование столбцов**, а поле **Выручка** – в поле **Значения**. При этом, в области **Значения** для поля **Выручка** автоматически устанавливается тип операции – **Сумма**. Если для поля **Выручка** нужна иная операция, тогда, щелкнув по полю **Сумма по полю Выручка**, выбирается команда **Параметры полей значений**. В открывшемся диалоговом окне выбирается из списка необходимая операция.

Вид и форма сводной таблицы изменяется на рабочем листе автоматически после выбора любого параметра в окне **Список полей сводной таблицы**.

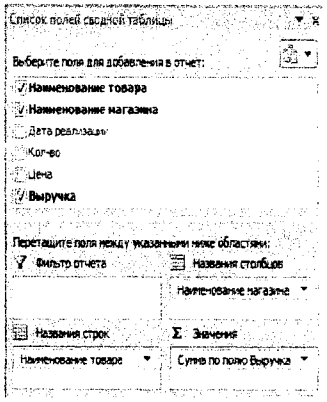


Рисунок 17 – Настройка полей сводной таблицы

ЯЗЫК ПРОГРАММИРОВАНИЯ VISUAL BASIC FOR APPLICATIONS В MICROSOFT EXCEL

Visual Basic for Applications (VBA) – язык программирования, встроенный в линейку приложений Microsoft Office и предназначенный для их автоматизации. Благодаря его сравнительной лёгкости освоения, офисные приложения могут создавать даже пользователи, не программирующие профессионально.

Обычно программный код VBA организуется в виде совокупности процедур (макросов), каждая из которых предназначена для решения определенной задачи. Наряду с процедурами, в языке VBA также могут быть созданы и собственные функции, которые затем будут использоваться в процедурах.

Безопасность макросов

Чтобы включить макросы в Excel 2003, необходимо выбрать пункт главного меню **Сервис / Макрос / Безопасность**. В открывшемся окне **Безопасность** установить переключатель **Уровень безопасности** в положение **Средняя** или **Низкая**.

В Excel 2007+ параметры безопасности макросов можно изменять так:

- ✓ На вкладке **Разработчик** в группе **Код** нажать кнопку **Безопасность макросов**. Если вкладка **Разработчик** не отображается, необходимо нажать **кнопку Microsoft Office**, щелкнуть **Параметры Excel**, а затем в категории **Основные** в разделе **Основные параметры работы с Excel** выбрать параметр **Показывать вкладку «Разработчик» на ленте**.
- ✓ В категории **Параметры макросов** в разделе **Параметры макросов** выбрать вариант **Отключить все макросы с уведомлением** (аналог Excel 2003 – **Средняя**) или **Включить все макросы** (аналог Excel 2003 – **Низкая**).

Если установлена опция **Средняя** (Excel 2003), то при открытии файла в диалоговом окне необходимо нажать кнопку **Не отключать макросы**.

Если установлена опция **Отключить все макросы с уведомлением** (Excel 2007+), то при открытии рабочих книг, содержащих макросы, макросы отключаются и Excel выводит предупреждение системы безопасности о том, что макрос отключен. Если есть необходимость включить макросы, то надо щелкнуть сначала на кнопке **Параметры** в строке предупреждения. Затем в открывшемся диалоговом окне **Параметры безопасности Microsoft Office** установить переключатель **Включить это содержимое** и щелкнуть на кнопке **ОК**.

Очень важно после этого закрыть и снова открыть данный файл. Без этого действия макросы будут по-прежнему отключены. Установленный уровень безопасности распространяется на все файлы Excel.

Открытие и сохранение макросов VBA

Для открытия редактора макросов VBA в Excel 2003 необходимо выбрать пункт меню **Сервис / Макрос / Редактор Visual Basic**, в Excel 2007+ – вкладку **Разработчик** в группе **Код** команду **Visual Basic**. В обеих системах эти действия можно заменить комбинацией клавиш **Alt+F11**.

Рабочие книги Excel 2003, содержащие макросы, сохраняются в файлах с расширением **XLS**, а рабочие книги Excel 2007+ – в файлах с расширением **XLSM**. Если такая рабочая книга Excel 2007+ сохраняется в формате **XLSX**, заданном по умолчанию, то Excel сохранит рабочую книгу без макросов. Если надо сохранить макросы, то нажимается **кнопка Microsoft Office**, выбирается пункт **Сохранить как** и в диалоговом окне **Сохранение документа** выбирается тип файла **Книга Excel с поддержкой макросов (*.xlsm)**.

Создание процедур и макросов

Перед тем как вводить код программы, необходимо вставить модуль в рабочую книгу. Если рабочая книга уже имеет лист модуля, его можно использовать для нового макроса.

Для вставки нового модуля, надо выполнить следующие действия:

✓ выбрать в окне **Project** рабочую книгу, которая используется в текущий момент;

✓ выбрать команду **Insert / Module**. В рабочей книге появится новый (пустой) модуль.

Каждый элемент в окне **Project** имеет определенное количество свойств. Для модификации свойств используется окно **Properties**. Чтобы отобразить окно **Properties**, выбирается команда **View / Properties Window** или нажимается клавиша **F4**. В окне **Properties** отображается список свойств выбранного элемента.

Для изменения имени модуля необходимо в окне **Project** выбрать нужный модуль, а затем в окне **Properties** в свойстве **Name** указать его новое имя. Чтобы удалить ненужный модуль, нужно щелкнуть по нему правой клавишей мыши в окне **Project** и в контекстном меню выбрать пункт **Remove Module...**. Далее в диалоговом окне нажать на кнопку **Нет**.

Процедура может быть двух типов: **подпрограммой** и **функцией**.

Процедура-подпрограмма – это нечто вроде новой команды, которая может быть выполнена либо пользователем, либо другим макросом. В рабочей книге Excel может содержаться произвольное число подпрограмм.

Подпрограммы всегда начинаются с ключевого слова **Sub**, после которого следует имя макроса (у каждого макроса должно быть уникальное имя), а затем – пара круглых скобок. (В этих скобках задаются аргументы, но если у подпрограммы нет аргументов, они остаются пустыми.) Оператор **End Sub** говорит об окончании подпрограммы. Строки, заключенные между этими двумя операторами, составляют тело процедуры или текст макроса.

Вторым типом процедуры VBA является **процедура-функция**. Функция всегда возвращает единственное значение (так же, как и обычная функция рабочей таблицы), которое внутри процедуры присваивается имени функции. Функции подобны подпрограммам, но начинаются ключевым словом **Function** и заканчиваются оператором **End Function**.

Таблица 2 – Пример создания процедур в VBA

Тип процедуры	Синтаксис	Пример

Функция	[Public | Private] **Function** *имя* ([*список_арг*]) [As *тип*]
	[*операторы*]
	имя = выражение
	End Function
		`Public Function Fun_F(x)`
		`'Процедура функции`
		
		`Fun_F = 2 * x + 3`
		`End Function`

Подпрограмма	[Public | Private] **Sub** *имя* ([*список_арг*])
	[*операторы*]
	End Sub
		`Public Sub Tab_F()`
		`'Процедура подпрограммы`
		
		`End Sub`

ЗАМЕЧАНИЕ: Служебные слова Private и Public задают область видимости процедур и функций:

✓ Private делает объект доступным только внутри данного модуля;

✓ Public делает объект доступным из другого модуля.

Примеры процедур, показанные в таблице 2, содержат комментарии (например, «'Процедура функции»). Комментарии – это заметки для того, кто составляет код процедуры, VBA их игнорирует. Строка комментариев начинается с апострофа. Комментарий можно поместить после любого оператора. Другими словами, если VBA встречает апостроф, он игнорирует остальной текст этой строки.

Создать шаблон процедуры типа **Sub** или **Function** можно автоматически:

✓ установить курсор в окне **Code**;

✓ выбрать пункт меню **Insert / Procedure...**;

✓ в появившемся диалоговом окне **Add Procedure** в поле **Name** указать имя процедуры (без пробелов и скобок), а в поле **Type** указать тип процедуры **Sub** (рис. 18) или **Function** (рис. 19).

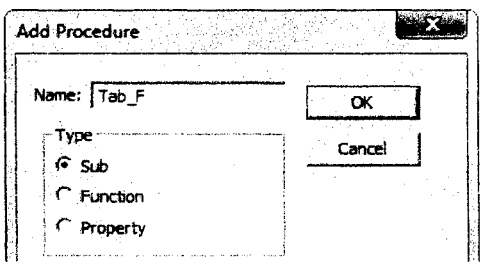


Рисунок 18 – Форма для создания процедуры **Sub**

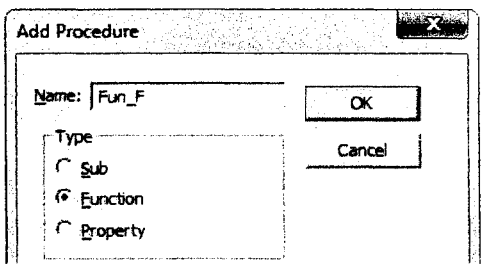


Рисунок 19 – Форма для создания процедуры **Function**

В процессе работы с MS Excel часто приходится многократно выполнять однотипные действия. В этом случае разумно создавать макросы с помощью макрорекордера (Macro Recorder), т.е. специального средства для записи макросов. При этом все необходимые действия записываются программой записи макроса и преобразуются автоматически в команды языка VBA. Работа макрорекордера во многом напоминает запись с помощью обычного магнитофона (диктофона). При этом перемещение курсора по ленте и рабочей книги не включается в записанные команды. Перед созданием макроса с помощью макрорекордера важно знать:

✓ Все шаги и команды, выполняемые макросом, должны быть спланированы перед записью или написанием макроса. Если при записи макроса была допущена ошибка, сделанные исправления также будут записаны.

✓ VBA хранит каждый записанный макрос в отдельном модуле, присоединенном к книге.

Порядок записи макроса:

1. Выбрать команду **Начать запись** (в Excel 2003 пункт меню **Сервис** / **Макрос**, в Excel 2007+ – вкладка **Разработчик** в группе **Код**).

2. В появившемся диалоговом окне ***Запись макроса***:

✓ в поле Имя макроса ввести имя для макроса, которое не должно начинаться с цифры, не должно иметь пробелов и символов пунктуации;

✓ в поле Сочетание клавиш для того чтобы запускать макрос с помощью сочетания клавиш, ввести букву (допускается использование сочетаний CTRL+ буква (для строчных букв) или CTRL+SHIFT+буква (для прописных букв), где буква – любая буквенная клавиша на клавиатуре);

✓ в поле Сохранить в: выбрать книгу, в которой требуется сохранить макрос (Личная книга макросов, Новая книга, Эта книга);

✓ в поле Описание можно, но необязательно, ввести необходимый текст для краткого описание макроса.

3. После нажатия кнопки ОК, Excel запустит макрорекордер и отобразит панель "Остановить запись".

4. Выполнить макрокоманды, которые нужно записать (для того чтобы прервать выполнение макроса, надо нажать кнопку ESC).

5. Нажать кнопку ■ Остановить запись на панели инструментов "Остановить запись".

Выполнение процедур и макросов

Перед использованием ***процедуры-функции*** обязательно нужно задать в скобках значения всех аргументов. Так, функцию VBA (Fun_F(x) в таблице 2) можно использовать:

✓ вызвав из другой процедуры VBA (например, y = Fun_F(4));

✓ ввести в ячейку рабочего листа точно так же, как и любую встроенную функцию Excel, только находящуюся в категории "Определенные пользователем" (рис. 20);

✓ вызвав из окна **Immediate** (см. ниже **Диалоговое окно Immediate**).

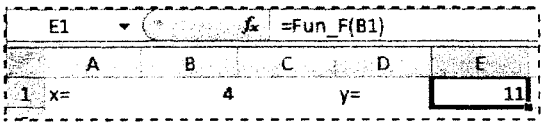


Рисунок 20 – Использование функции Fun_F(x) на рабочем листе

Процедуры-подпрограммы запускаются другими способами:

1. Запуск из рабочего листа Excel. В этом диалоговом окне **Макрос** выбрать из списка имя нужной подпрограммы и щелкнуть на кнопке **Выполнить**. Для открытия диалогового окна **Макрос** можно воспользоваться комбинацией клавиш **ALT + F8** или выполнить команду:

✓ Excel 2003: **Макросы...** из пункта главного меню **Сервис** / **Макрос**.

✓ Excel 2007+: **Макросы** на вкладке **Разработчик** в группе **Код**.

2. Запуск из окна **Code** редактора VBA. Поместить курсор где-нибудь в коде макроса и выполнить одно из предложенных действий:

✓ выбрать пункт главного меню **Run** / **Run Sub**;

✓ нажать на пиктограмму **Run Sub** панели инструментов **Standard** – ▶ ;

✓ нажать клавишу **F5**.

3. Вызов из другой процедуры VBA (например, Call Tab_F или просто Tab_F).

4. Запуск из окна **Immediate** (см. ниже **Диалоговое окно Immediate**).

5. Процедуры (в том числе и макросы) можно закрепить:

✓ за кнопкой на рабочем листе;

✓ за пиктограммой на имеющейся панели инструментов или на новой (для Excel 2003);

✓ за командой в любом пункте меню (для Excel 2003);

✓ в виде кнопок на ленте или панели быстрого доступа (Excel 2007+).

Пошаговое выполнение макросов

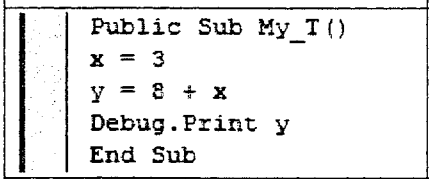
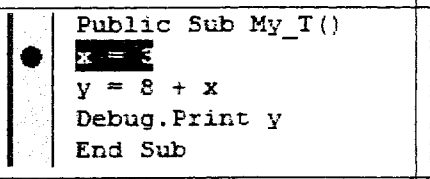
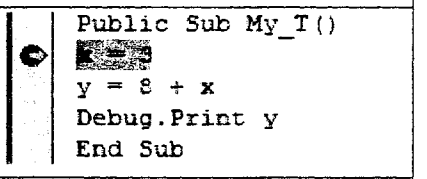
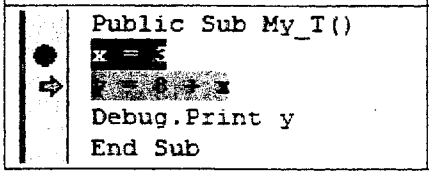
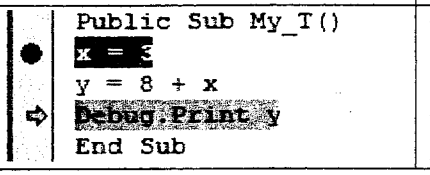
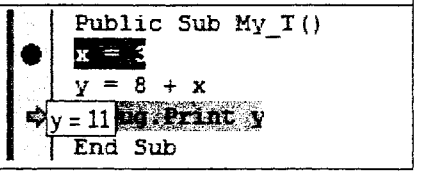
При отладке программы часто возникает необходимость проследить за тем, как выполняются отдельные операторы в отлаживаемой программе или же какие значения принимают те или иные переменные на различных этапах выполнения программы. Для достижения этих целей в редакторе VBA реализован режим прерывания.

Перевести программу в режим прерывания в некоторый, заранее известный момент ее выполнения можно с помощью установки **точки останова**. Чаще всего **точку останова** помещают в строке программного кода непосредственно перед оператором, который, по вашему предположению, является причиной возникновения ошибки. Как только по ходу выполнения программы достигается оператор, которому назначена **точка останова**, работа программы приостановится (шаг 3 таблицы 3). Теперь можно будет проверить текущие значения любых переменных (шаг 6 таблицы 3). Последующее нажатие **F8** позволяет выполнять процедуру по шагам, в порядке естественного выполнения операторов (шаги 4 и 5 таблицы 3). Если очередной оператор является вызовом процедуры типа **Sub** или **Function**, то применение **F8** приведет к открытию текста вызываемой процедуры в окне редактирования кода и позволит пройти эту процедуру по шагам, увидев результаты всех выполняемых в ней действий.

Для размещения или отмены точки останова в программном коде необходимо выполнить одно из следующих действий (шаг 2 таблицы 3):

- ✓ щелкнуть на серой полоске, расположенной в левой части окна редактирования кода, напротив требуемой строки программного кода;
- ✓ поместить курсор в нужную строку кода и нажать **F9**.

Таблица 3 – Элементы пошагового выполнения макроса

Шаг 1 – Создали процедуру My_T() типа Sub	Шаг 2 – Установили **точку останова** с помощью **F9**	Шаг 3 – Запустили процедуру с помощью **F5**
		
Шаг 4 – Перешли к следующему оператору с помощью F8	**Шаг 5 – Перешли к следующему оператору с помощью F8**	**Шаг 6 – Удерживаем указатель мыши над переменной *y***
		

Количество размещаемых точек останова неограниченно, поэтому в разных строках программы можно разместить столько точек останова, сколько сочтете необходимым. Единственное условие – нельзя размещать точки останова в строках комментариев и строках с операторами, которые VBA на самом деле не выполняет, например, в строках с объявлениями переменных.

В редакторе VBA реализован механизм автоматической подсказки, которая позволяет увидеть текущее значение любой переменной в программе. Для этого, когда программа находится в режиме прерывания, необходимо задержать на секунду указатель мыши на имени интересующей вас переменной, и на экране появится окно подсказки – небольшой прямоугольник с именем и текущим значением переменной в нем (шаг 6 таблицы 3).

При выполнении макросов могут возникнуть ошибки, отличные от синтаксических, – ошибки выполнения. Такие ошибки можно обнаружить только в процессе выполнения процедуры. Обычно такие ошибки выполнения приводят к остановке процедур VBA. При этом отображается диалоговое окно с сообщением, в котором указывается номер ошибки и краткое ее описание (рис. 21).

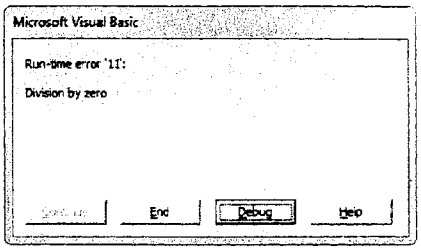


Рисунок 21 – Сообщение об ошибке

В этом окне кнопка **End** прекращает выполнение процедуры. Щелчок на кнопке **Debug** переводит выполняющуюся процедуру в режим прерывания, что позволяет в окне кода редактора VBA внести исправления в тот оператор, где возникла данная ошибка выполнения, а затем продолжить либо завершить (**Run / Reset** или на **Reset** панели инструментов) выполнение программы.

Диалоговое окно Immediate

Помимо средств прерывания и пошагового выполнения, отладчик редактора VBA предоставляет диалоговое окно **Immediate** (Окно немедленного выполнения команд), призванное дополнительно упростить процесс отладки программ и предоставить дополнительные инструменты поиска ошибок.

Чтобы открыть окно **Immediate**, необходимо выполнить одно из следующих действий:

- ✓ выбрать команду меню **View / Immediate Window**;
- ✓ щелкнуть на кнопке **Immediate Window** панели инструментов **Debug**;
- ✓ нажать **Ctrl+G**.

Окно **Immediate** редактора VBA позволяет контролировать значения переменных и выражений, проводить вычисления, изменять значения переменных и проверять результаты выполнения функций. Кроме того, в окне **Immediate** можно выполнять отдельные макросы. Так, для запуска процедуры типа **Sub** достаточно указать ее имя и через пробел значения аргументов при их наличии (макрос My_T( ) из таблицы 4). Для запуска процедуры типа **Function** в окне **Immediate** используется оператор **Print** либо знак «?» (макрос Fun_F(x) из таблицы 4). Этот оператор вводится в окне **Immediate**, затем указывается имя функции и в скобках значения аргументов. После завершения ввода нажимается клавиша **Enter**.

Таблица 4 – Результаты выполнения макросов в окне **Immediate**

Макрос	**Function** Fun_F(x) из таблицы 2	**Sub** My_T( ) из таблицы 3
Результат выполнения в окне **Immediate**	Immediate ?Fun_F(4) 11	Immediate My_T 11

При необходимости можно направить в окно **Immediate** вывод промежуточных значений переменных и выражений (макрос My_T( ) из таблицы 4) непосредственно при выполнении программы. Для этого надо разместить в подходящих строках программы операторы, вызывающие метод **Debug.Print** (шаг 1 таблицы 3).

Ввод-вывод данных на рабочий лист

VBA – это объектно-ориентированный язык программирования. Это означает, что основными его элементами являются объекты – рабочие книги (**Workbook**), рабочие листы (**Worksheet**), ячейки или диапазоны ячеек (**Range**, **Cells**), диаграммы (**Chart**) и др. Эти объекты расположены в иерархическом порядке.

Набор однотипных объектов называется коллекцией. Например, коллекция всех объектов рабочих листов (**Worksheet**) называется **Worksheets**. Обратиться к отдельному объекту коллекции можно, используя порядковый номер или ссылки. Например, если в рабочей книге есть три рабочих листа: **Лист1**, **Лист2** и **Лист3**, то к объекту коллекции рабочих листов **Лист2** можно обратиться так – **Worksheets("Лист2")**.

Для выбора объекта, с которым предстоит дальнейшая работа, необходимо использовать метод **Select**. Метод – это действие, примененное к объекту. Так, например, чтобы выбрать рабочий лист **Лист2** (сделать его активным), необходимо записать в макросе следующую команду – **Worksheets("Лист2").Select**.

Для обращения к определенной ячейке рабочего листа необходимо использовать свойства **Range**(“*адрес ячейки*”) или **Cells**(*номер строки ячейки, номер столбца ячейки*). С помощью этих свойств можно передавать значения переменной в ячейку и из ячейки – переменной. Примеры приведены в таблице 5.

Таблица 5 – Примеры использования свойств **Range** и **Cells**

Условие:	Присвоить переменной f значение из ячейки **C2** рабочего листа	Вывести на рабочий лист в ячейку **B3** результат выражения 2^2-1
A B C 1 2 4 3 3	f = Range(“C2”) или f = Cells(2,3)	Range(“B3”) = 2^2 - 1 или Cells(3,2) = 2^2 - 1
Результат:	f = 4	В ячейке B3 число 3

При использовании в выражении без указания координат свойство **Cells** объекта **Worksheet** определяет диапазон, включающий все ячейки данного рабочего листа. Совместное действие **Cells** и метода **Clear** позволяет очистить содержимое всех ячеек активного рабочего листа. Например, **Cells.Clear**.

Диалоговые окна

В VBA есть две функции – **InputBox** и **MsgBox**, которые позволяют отображать простые диалоговые окна. Эти диалоговые окна можно видоизменить несколькими способами, но, конечно, они не могут содержать всех тех опций, которые доступны в созданных пользователем диалоговых окнах.

В краткой форме функция VBA **InputBox**, предназначенная для ввода значений переменным, имеет следующий синтаксис:

InputBox(*сообщение* [,*заголовок*] [,*по_умолчанию*])

Назначение аргументов:

- ✓ *сообщение* – текст, отображаемый в окне (обязательный аргумент);
- ✓ *заголовок* – текст, который появляется в строке заголовка окна (необязательный аргумент);
- ✓ *по_умолчанию* – значение, отображаемое в окне ввода по умолчанию (необязательный аргумент).

В следующем примере функция **InputBox** предлагает пользователю ввести значение переменной *x*.

```
x = InputBox("введите значение x", "ввод значений")
```

При выполнении этого оператора VBA Excel выводит на экран диалоговое окно, показанное на рисунке 22. Следует заметить, что в данном примере использованы только два первых аргумента, параметр *по_умолчанию* не указан. Когда пользователь введет некоторое значение и щелкнет на кнопке ОК, это значение будет присвоено переменной *x*. Функция **InputBox** всегда возвращает строку (текст), поэтому может возникнуть необходимость в преобразовании результата в число. Для этого необходимо воспользоваться функцией **Val**, которая преобразует текст в число:

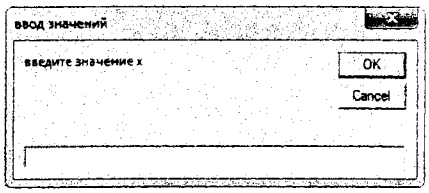


Рисунок 22 – Диалоговое окно функции **InputBox**

```
x = Val(InputBox("введите значение x", "ввод значений"))
```

Если в качестве параметра *по_умолчанию* задать значение 3.5 (т.е. x=3.5), то код будет выглядеть так:

```
x = Val(InputBox("введите значение x", "ввод значений", "3.5"))
```

Функция VBA **MsgBox** – это весьма удобное средство для того, чтобы отобразить на экране информацию и попросить пользователя сделать выбор, щелкнув на одной из предложенных кнопок. Эта функция используется для того, чтобы вывести значение переменной. Краткий синтаксис функции **MsgBox** выглядит следующим образом:

MsgBox(*сообщение* [,*кнопки*] [,*заголовок*])

Описание аргументов:

- ✓ *сообщение* – текст, отображаемый в окне сообщения (обязательный аргумент);
- ✓ *кнопки* – коды кнопок, которые отобразятся в окне сообщения (необязательный аргумент);
- ✓ *заголовок* – текст, который появляется в строке заголовка окна сообщения (необязательный аргумент).

Функцию **MsgBox** можно вызывать в виде отдельного оператора, не присваивая ее результат какой-нибудь переменной. Если функция вызывается самостоятельно, то аргументы не нужно заключать в круглые скобки.

В следующем примере функция **MsgBox** выводит на экран сообщение о переменной *x* = 3.5 (рис. 23):

```
MsgBox "значение переменной x = " + Str(x)
```

Здесь в тексте *сообщения* использованы знак «+» для соединения (склеивания) двух частей текста и функция **Str**, которая преобразует числовое значение переменной *x* в текст.

Настраивать окна сообщений можно с помощью соответствующих встроенных констант VBA, которые используются в качестве аргументов **MsgBox**.

Встроенные константы VBA, отвечающие за тип пиктограммы в диалоговом окне сообщений, перечислены в таблице 6.

Таблица 6 – Константы в **MsgBox**

Константа	*Знак*
vbCritical	
vbQuestion	
vbExclamation	
vbInformation	

В следующем примере в операторе **MsgBox** заданы все три аргумента:

```
MsgBox "значение переменной" + vbCr + "x = " + Str(x), vbInformation, "Вывод результата"
```

Результат приведенного оператора отображен на рисунке 24. В первом аргументе *сообщение* использована встроенная константа **vbCr**, которая позволяет разбить текст на две части и перенести вторую часть на следующую строку. Если в качестве аргумента используется текст, то он заключается в кавычки.

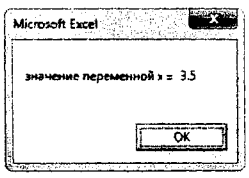


Рисунок 23 – Диалоговое окно оператора **MsgBox**

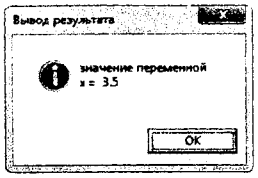


Рисунок 24 – Диалоговое окно оператора **MsgBox**

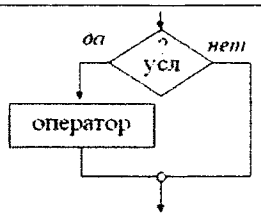
Оператор IF

Оператор **If...Then** – один из важнейших управляющих операторов. Он организует выполнение заданного блока операторов, если при вычислении указанного условного выражения будет получено значение ИСТИНА, и не делает ничего, если будет получено значение ЛОЖЬ. Такая конструкция **If...Then** записывается в одной строке, и завершающий оператор **End If** для нее не требуется (таблица 7).

Однако в операторе **If...Then** может выполняться не только один оператор, но и блок последовательно следующих операторов. В этом случае проверяемое условие и выполняемые операторы записываются в отдельных строках. Если условное выражение имеет значение ИСТИНА, выполняемая группа операторов начинается с первого оператора, стоящего после ключевого слова **Then**, и включает все последующие операторы, вплоть до оператора завершения **End If** (таблица 7).

Таблица 7 – Примеры краткой формы разветвляющей структуры

Блок-схема	*Оператор*	*однострочный*	*многострочный*

	**Синтаксис**	If *условие* Then *оператор*	If *условие* Then
			операторы
			End If

	Пример	If x > 5 Then MsgBox "x больше 5"	If x > 5 Then
			MsgBox "x больше 5"
			End If

Для более сложных ситуаций, когда на основании некоторого условия необходимо выбрать одну из двух различных последовательностей операторов, используется оператор **If...Then...Else**.

Как и оператор **If...Then**, конструкция **If...Then...Else** может быть однострочной (таблица 8) и многострочной (таблица 9).

Таблица 8 – Пример полной формы разветвляющей структуры (однострочный оператор **If**)

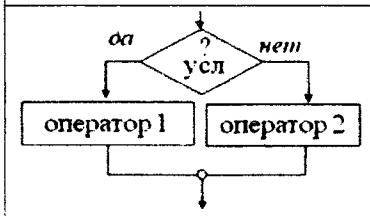
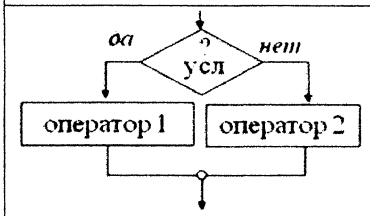
Блок-схема	*Оператор*	*однострочный*
	**Синтаксис**	If *условие* Then *оператор 1* Else *оператор 2*
	Пример	If x > 5 Then MsgBox "x больше 5" Else MsgBox "x не больше 5"

Таблица 9 – Пример полной формы разветвляющей структуры (многострочный оператор **If**)

Блок-схема	*Оператор*	*Синтаксис*	*Пример*

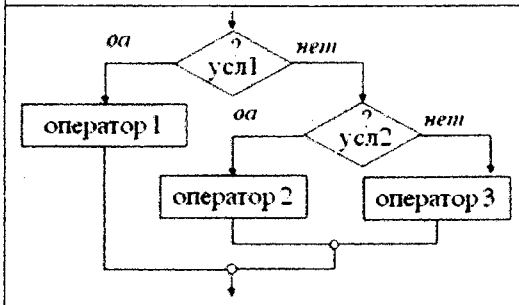
	многострочный	If *условие* Then
		оператор 1
		Else
		оператор 2
		End If
			If x > 5 Then
			MsgBox "x больше 5"
			Else
			MsgBox "x не больше 5"
			End If

Часто при написании программ возникают такие ситуации, когда приходится проверять не одно, а два или больше условий. В этом случае можно поместить один оператор **If...Then** внутри другого оператора **If...Then** (или оператор **If...Then...Else** внутри оператора **If...Then...Else**), что называется вложением операторов.

Вложенные операторы следует использовать тогда, когда для принятия решения необходимо проверить дополнительное условие, но при этом первое условие должно оказаться истинным. Если же первое условие не выполнено, то будут выполняться операторы, следующие за ключевым словом **Else**.

Таблица 10 – Пример многострочного оператора **If** с вложенным оператором **If**

Блок-схема	*Синтаксис*	*Пример*

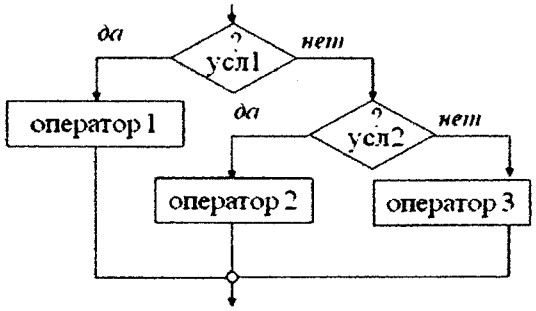
	If *условие 1* Then
	оператор 1
	Else
	If *условие 2* Then
	оператор 2
	[Else
	оператор 3]
	End If
	End If
		If x > 5 Then
		MsgBox "x больше 5"
		Else
		If x < 5 Then
		MsgBox "x меньше 5"
		Else
		MsgBox "x равно 5"
		End If
		End If

Эквивалентом вложенных операторов является оператор **If...Then...Elself**, в котором используется ключевое слово **ElseIf**. С помощью ключевого слова **ElseIf** в одном условном операторе можно проверить несколько условий. Синтаксис этого условного оператора выглядит так, как показано в таблице 11.

Ключевое слово **Else** является необязательным, но если оно все же присутствует, то должно быть записано в конструкции оператора последним.

Таблица 11 – Пример многострочного расширенного оператора **If**

Блок-схема	*Синтаксис*	*Пример*

	If *условие 1* Then
	оператор 1
	ElseIf *условие 2* Then
	оператор 2
	[Else
	оператор 3]
	End If
		If x > 5 Then
		MsgBox "x больше 5"
		ElseIf x < 5 Then
		MsgBox "x меньше 5"
		Else
		MsgBox "x равно 5"
		End If

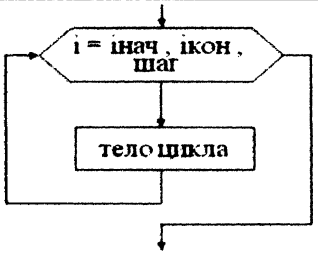
Ключевое слово **Elself** может повторяться в конструкции сколько угодно раз. При этом выполнение заданных условий проверяется строго последовательно, в том порядке, в каком они записаны. И как только истинное условие (ИСТИНА) будет найдено, выполняется соответствующий ему блок операторов, после чего выполнение данного условного оператора прекращается.

Операторы цикла For...Next, Do...Loop и While...Wend

Под циклом мы будем понимать такой оператор или последовательность операторов, которые по ходу выполнения программы могут при необходимости выполняться многократно.

Оператор **For...Next** относится к операторам создания циклов простейшего типа. В программе на языке VBA с его помощью блок операторов выполняется заданное количество раз, известное до начала выполнения цикла. Пример такого оператора приведен в таблице 12.

Таблица 12 – Пример оператора цикла **For...Next**

Блок-схема	*Синтаксис*	*Пример*	*Результат*
	For *i* = *iнач* To *iкон* [Step *шаг*] *тело цикла* Next *i*	For i = 0 To 10 Step 2 Debug.Print i ^ 2 Next i	Immediate 0 4 16 36 64 100

В таблице 12 *i* – любая числовая переменная, значение которой меняется в начале каждого цикла (проходе). Параметры *iнач* и *iкон* – это числовые выражения, задающие начальное и конечное значения переменной цикла *i*.

Числовая переменная *шаг* задает приращение, на которое увеличивается переменная цикла *i* при каждом проходе. Вся фраза с ключевым словом **Step** необязательна. При ее отсутствии интерпретатор VBA (по умолчанию) увеличивает переменную цикла *i* на единицу при каждом выполнении *тела цикла*.

Операторы циклов **Do...Loop** (в отличие от операторов **For...Next**) выполняются не заданное количество раз, а сколь угодно долго – до тех пор, пока не будет выполнено некоторое логическое условие. Язык VBA включает два варианта организации такого цикла: цикл с предусловием и цикл с постусловием.

В цикле с предусловием логическое условие проверяется до начала выполнения цикла. В этом случае, если после оператора цикла **Do** указано ключевое слово **While**, то тело цикла повторяется до тех пор, пока логическое условие, стоящее после **While**, будет истинно (таблица 13). Если указано ключевое слово **Until**, то тело цикла выполняется до тех пор, пока логическое условие будет ложно (таблица 14).

Таблица 13 – Примеры операторов цикла **Do...Loop (While...Wend)** с предусловием ИСТИНА

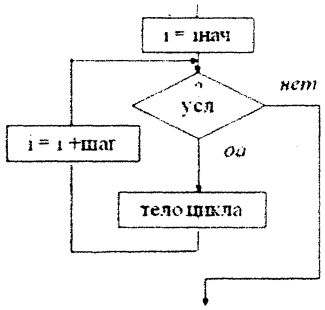
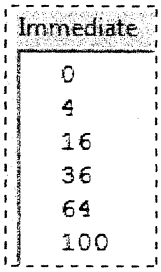
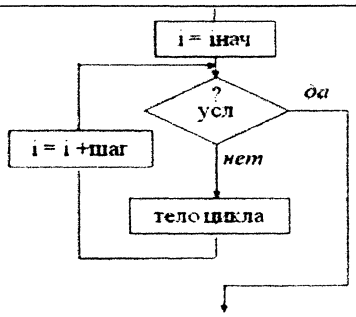
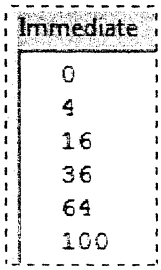
Блок-схема	*Синтаксис*	*Пример*	*Результат*
	*i = iнач* Do While *условие* *тело цикла* *i = i + шаг* Loop --- *i = iнач* While *условие* *тело цикла* *i = i + шаг* Wend	i = 0 Do While i <= 10 Debug.Print i ^ 2 i = i + 2 Loop --- i = 0 While i <= 10 Debug.Print i ^ 2 i = i + 2 Wend	

Таблица 14 – Примеры операторов цикла **Do...Loop** с предусловием ЛОЖЬ

Блок-схема	*Синтаксис*	*Пример*	*Результат*
	*i = iнач* Do Until *условие* *тело цикла* *i = i + шаг* Loop	i = 0 Do Until i > 10 Debug.Print i ^ 2 i = i + 2 Loop	

На практике конструкция оператора **Do While...Loop** наиболее популярна, поэтому в VBA ее часто заменяют оператором цикла **While...Wend**, синтаксис и пример которого приведен в таблице 13.

В цикле с постусловием логическое условие проверяется после выполнения цикла. В этом случае, если после оператора цикла **Loop** указано ключевое слово **While**, то тело цикла повторяется до тех пор, пока логическое условие, стоящее после **While**, будет истинно (таблица 15). Если указано ключевое слово **Until**, то тело цикла выполняется до тех пор, пока логическое условие будет ложно (таблица 16).

Таблица 15 – Примеры операторов цикла **Do...Loop** с постусловием ИСТИНА

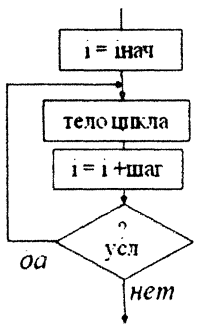
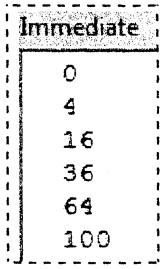
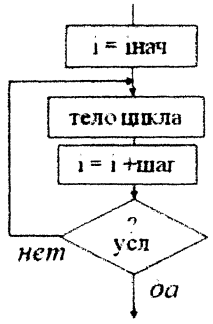
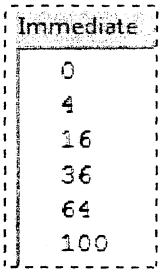
Блок-схема	*Синтаксис*	*Пример*	*Результат*
	*i = iнач* Do *тело цикла* *i = i + шаг* Loop While *условие*	i = 0 Do Debug.Print i ^ 2 i = i + 2 Loop While i <= 10	

Таблица 16 – Примеры операторов цикла **Do...Loop** с постусловием ЛОЖЬ

Блок-схема	*Синтаксис*	*Пример*	*Результат*
	*i = iнач* Do *тело цикла* *i = i + шаг* Loop Until *условие*	i = 0 Do Debug.Print i ^ 2 i = i + 2 Loop Until i > 10	

Создание формы

Пользовательские диалоговые окна создаются в редакторе VBA как экранные формы. Такая экранная форма – форма пользователя – состоит из окна формы (это то самое окно, которое отображается на экране) и нескольких специфических элементов управления, размещенных в этом окне. Все формы, как и любые элементы управления в форме, – это полноценные объекты VBA.

Для создания в проекте VBA новой формы пользователя необходимо активизировать этот проект в окне проектов и либо выбрать команду меню **Insert** / **UserForm** редактора VBA, либо щелкнуть правой кнопкой мыши в окне **Project** и выбрать команду **Insert** / **UserForm** в раскрывшемся контекстном меню. При этом новая форма (т.е. заготовка новой формы – рис. 25) появится в специально созданном для нее окне. Рядом с формой будет отображена панель элементов управления **Toolbox** – специальная панель инструментов, содержащая кнопки, с помощью которых соответствующие элементы управления можно разместить в создаваемой форме.

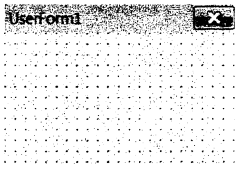


Рисунок 25 – Заготовка новой формы

По умолчанию формам пользователя (т.е. соответствующим объектам) автоматически присваиваются имена **UserForm1**, **UserForm2** и т.д. Имя формы пользователя можно изменить на более понятное, отражающее назначение формы. Для этого достаточно выбрать требуемую форму в окне проектов, а затем в окне свойств **Properties** изменить значение ее свойства **Name**. Если окно свойств не отображено в окне редактора VBA, предварительно надо нажать клавишу **F4** для вывода этого окна на экран, после чего изменить значение свойства **Name** формы.

Для отображения формы на экране используется метод **Show** объекта **UserForm**. Синтаксис вызова метода следующий: **ИмяФормы.Show**. Здесь обозначение **ИмяФормы** представляет имя объекта требуемой формы, содержащееся в свойстве **Name**. Ниже приведен пример процедуры типа **Sub** (она должна находиться в обычном модуле VBA, а не в модуле экранной формы), которая отображает на экране форму **Form1**.

```
Sub Vyvod_form()
   Form1.Show
End Sub
```

Для закрытия окна используется команда **Upload Me** либо **End**, которая обычно записывается в *событие* **Click** элемента управления **CommandButton**.

Событие – это то, что происходит, когда пользователь воздействует на элемент управления экранной формы. Например, щелчок на командной кнопке инициирует событие **Click**, ассоциированное с данной кнопкой. Программное приложение должно иметь процедуры, которые бы выполнялись при наступлении того или иного события. Такие процедуры часто называют *процедурами обработки событий.* Процедуры обработки событий носят имена, в которых название элемента управления объединено с названием события с помощью символа подчеркивания. Например, процедура, которая выполняется после щелчка на кнопке **MyButton**, называется **MyButton_Click**. Если эта кнопка предназначена для закрытия формы **Form1**, то процедура обработки события **Click** для нее будет выглядеть так:

```
Private Sub MyButton_Click()
   Unload Form1
End Sub
```

Для формирования такой процедуры можно выполнить двойной щелчок по кнопке **MyButton**, размещенной на форме **Form1**.

Для перемещения между конструктором формы и процедурами обработки событий элементов управления на панели инструментов окна **Project** слева находятся две кнопки **View Code** и **View Object**

Элементы управления

Элементы управления, которые можно вставить в экранную форму, находятся на панели инструментов **Toolbox**. Чтобы вставить какой-либо элемент управления в экранную форму, щелкните на соответствующем инструменте панели **Toolbox**, затем щелкните на экранной форме. Можно также сначала щелкнуть на нужном инструменте панели **Toolbox**, а затем протащить указатель мыши по экранной форме, указывая размер элемента управления.

Если выделить на форме один из ее элементов управления, то в окне **Properties** будут представлены свойства данного элемента управления. В этом окне слева указываются имена свойств, а справа от каждого имени – его текущее значение.

Таблица 17 – Основные свойства часто используемых элементов управления

Объект	*Свойство*	*Описание*
Label	Caption	Содержит текст, отображаемый в элементе управления
TextBox	Value	Введенный в поле текст
CheckBox	Caption Value	Надпись, отображаемая рядом с данным элементом управления Выбор элемента управления
OptionButton	Caption Value	Надпись, отображаемая рядом с данным элементом управления Выбор элемента управления
Frame	Caption	Надпись, отображаемая вверху элемента управления
CommandButton	Caption	Надпись, отображаемая в центре элемента управления
Image	Picture	Путь к графическому файлу

Свойства элементов управления можно изменять как в процессе разработки экранной формы – вручную, внося изменения в окно свойств элемента управления, так и во время выполнения экранной формы, когда она уже выведена на экран для работы с пользователем, – программно. В последнем случае для изменения свойств элементов управления надо использовать соответствующие операторы VBA. Например:

Имя_объекта.Свойство = Значение

Так команда **TextA.Value=Str(3.5)** поместит в элемент управления **TextBox** с именем **TextA** текст **3.5**.

СИСТЕМА КОМПЬЮТЕРНОЙ АЛГЕБРЫ MATHCAD

Определение нулей и экстремумов функции

Для нахождения нулей функции в MathCAD используется функция **root( )**:

root(функция, начальное приближение[, начало отрезка, конец отрезка]).

Перед началом решения желательно построить график функции, чтобы проверить, есть ли корни, т.е. пересекает ли график ось абсцисс. Начальное приближение лучше всего выбрать по графику поближе к значению корня.

В функции root( ) достаточно использовать два первых параметра. В этом случае необходимо предварительно задавать начальное приближение.

Если указывать в функции root( ) все четыре параметра, тогда начальное приближение предварительно можно не задавать. Необходимо помнить, что внутри указанного интервала должен быть только один нуль функции.

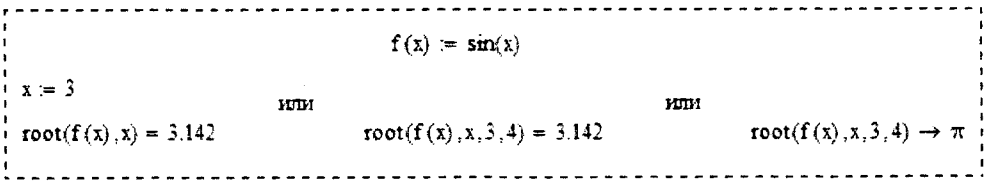


Рисунок 26 – Пример использования функции **root( )**

Для нахождения локальных экстремумов в MathCAD используются функции **maximize( )** и **minimize( )**:

maximize(имя функции, аргументы),
minimize(имя функции, аргументы).

Использование функций maximize( ) и minimize( ) аналогично функции find( ). Ключевое слово Given можно опустить – оно необходимо только при наличии ограничений (рис. 27).

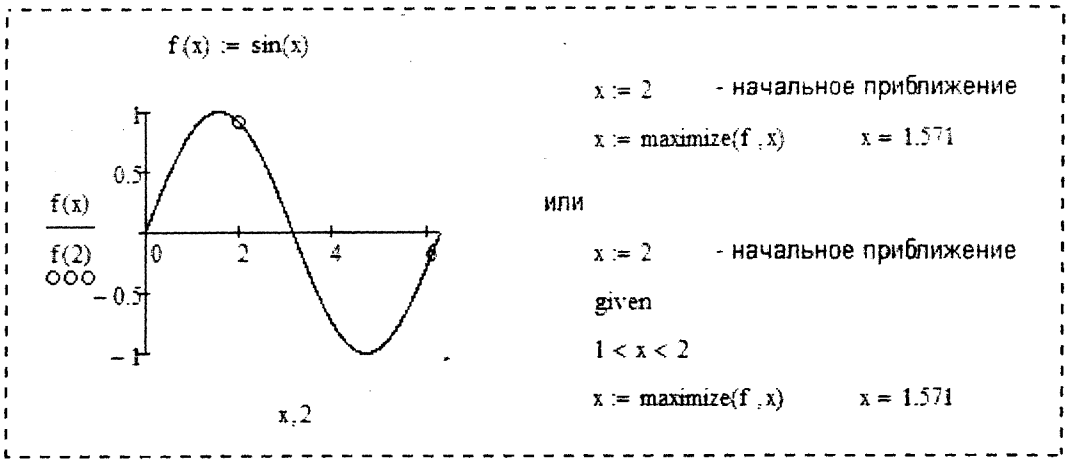


Рисунок 27 – Пример использования функции **maximize( )**

Элементы программирования

Элементы программирования в MathCAD служат для создания программных модулей, которые содержат в себе множество строк и фактически являются составными выражениями.

Программный модуль состоит из названия, следующего за ним знака присвоения значения и необходимых выражений в правой части, записанных в столбик и объединенных слева вертикальной чертой.

Для создания программного модуля надо:

- ✓ ввести имя программного модуля и, если это необходимо, в скобках указать входные переменные;
- ✓ ввести оператор присваивания «:=»;
- ✓ щелкнуть на панели **Программирование** по кнопке **Add Line** (или клавишу <]>) столько раз, сколько строк должна содержать программа;
- ✓ в появившиеся маркеры ввести нужные операторы, лишние маркеры удалить.

Чтобы создать недостающие маркеры, надо поставить уголок курсора ввода в конец строки, после которой нужно будет вводить новую строку. Клавишей пробела следует выделить полностью всю строку и нажать кнопку **Добавить строку**.

Присваивание значений переменным и константам в программном модуле производится с помощью программного оператора присваивания «←», который вводится с панели **Программирование** нажатием кнопки «←». При создании программы, когда этот знак приходится использовать часто, полезно пользоваться клавишей <{>.

Любой программный модуль представляет собой сочетание обычных математических выражений с операторами условия и цикла.

Условный оператор if

Действие условного оператора **if** состоит из двух частей. Сначала проверяется условие, записанное справа от оператора **if.** Если оно верно, выполняется выражение, стоящее слева от **if,** если не верно, происходит переход к следующей строке программы. Она может содержать новое условие или быть обычным выражением.

Часто встречается условие с двумя вариантами действия: «Если..., то ..., иначе...». Для слова «иначе» на панели **Программирование** имеется оператор **otherwise** (иначе), который вводится так же, как **if.**

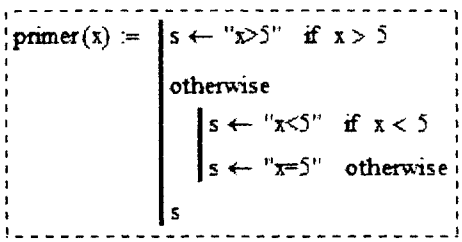


Рисунок 28 – Реализация в MathCAD примера из таблицы 10

На практике условие «если-то-иначе» необходимо вводить в таком порядке:

- ✓ в создаваемой программе установить курсор на свободное место ввода, где должен появиться условный оператор;
- ✓ на панели **Программирование** щелкнуть на кнопке **if.** В программе появится шаблон оператора с двумя маркерами;
- ✓ в правый маркер ввести условие. Можно пользоваться при этом логическими операторами, вводя их с панели **Булевы операторы**;

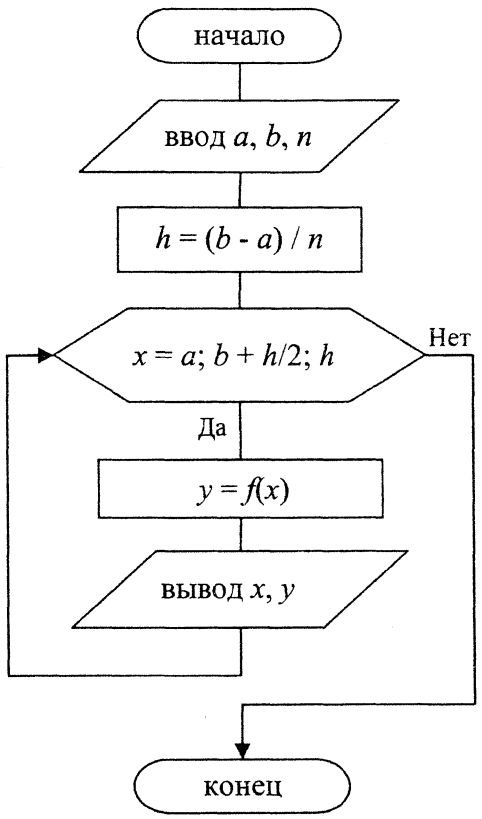
- ✓ слева от оператора **if** ввести выражение, которое должно выполняться, если условие верно. Если при выполнении условия должно выполняться сразу несколько выражений, надо иметь несколько маркеров. Установить курсор ввода на маркер слева от **if** и нажать **Add Line** столько раз, сколько строк надо ввести. Обратить внимание на то, что при этом изменяется вид условного оператора. Столбик маркеров появится не слева, а под оператором **if**;
- ✓ на следующем свободном маркере (на следующей строке) пишется выражение, выполняемое «иначе», если условие не выполняется;
- ✓ выделяется курсором выражение так, чтобы уголок курсора ввода был в конце выражения, и на панели **Программирование** нажимается **otherwise**.

ЧИСЛЕННЫЕ МЕТОДЫ

Табулирование функции

Табулирование функции – это вычисление значений функции при изменении аргумента от некоторого начального значения до некоторого конечного значения с определенным шагом. Именно так составляются таблицы значений функций, отсюда и название – табулирование. Необходимость в табулировании возникает при решении достаточно широкого круга задач. Например, путем табулирования можно отделить (локализовать) корни уравнения, т.е. найти такие отрезки, на концах которых функция имеет разные знаки. Необходимость в табулировании возникает также при построении графиков функции и т.д.

Блок-схема алгоритма табулирования:



Программная реализация табулирования функции $y(x) = \cos^2(x) - \sin(x^2 + 2x - 3)$ в VBA:

```
Sub Tab_f()
 'функция y(x) определена в процедуре-функции Fun_y()
 a = 0
 b = 3
 n = 10

 h = (b - a) / n
 Range("A1") = "a = ": Range("B1") = a
 Range("A2") = "b = ": Range("B2") = b
 Range("A3") = "n = ": Range("B3") = n
 Range("A4") = "h = ": Range("B4") = h

 Range("A6") = "x": Range("B6") = "y"
 rw = 7
 For x = a To b + h / 2 Step h
         y = Fun_y(x)
         Cells(rw, 1) = x: Cells(rw, 2) = y
         rw = rw + 1
 Next x
End Sub
```

Результат выполнения в Excel:

	A	B
1	a =	0
2	b =	3
3	n =	10
4	h =	0,3
5		
6	x	y
7	0	1,14112
8	0,3	1,651673
9	0,6	1,672637
10	0,9	0,766587
11	1,2	-0,61334
12	1,5	-0,77307
13	1,8	0,69462
14	2,1	0,878349
15	2,4	-0,41335
16	2,7	1,07947
17	3	1,516658

Метод дихотомии (уточнение нуля функции)

Полагаем, что отделение корней произведено и на интервале [a, b] расположен один корень функции *f(x)*, который необходимо уточнить с погрешностью **ε**. Одним из методов уточнения корня на отрезке является метод половинного деления (метод дихотомии). Суть метода заключается в последовательном делении выбранного отрезка на две равные части до тех пор, пока одновременно не выполнятся оба условия:

$|f(c)| \le \varepsilon$ и $|b-a| \le \varepsilon$,

где c – середина отрезка [a, b], c = (b – a) / 2; ε – точность вычисления (задается пользователем).

На каждом этапе деления та часть, на которой нет нуля функции, отбрасывается. Графически такую ситуацию можно представить так:

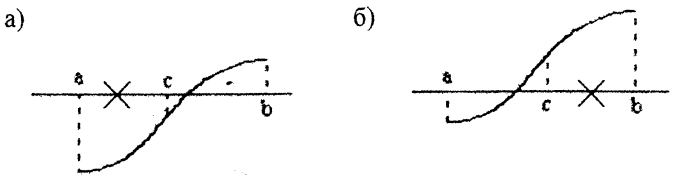
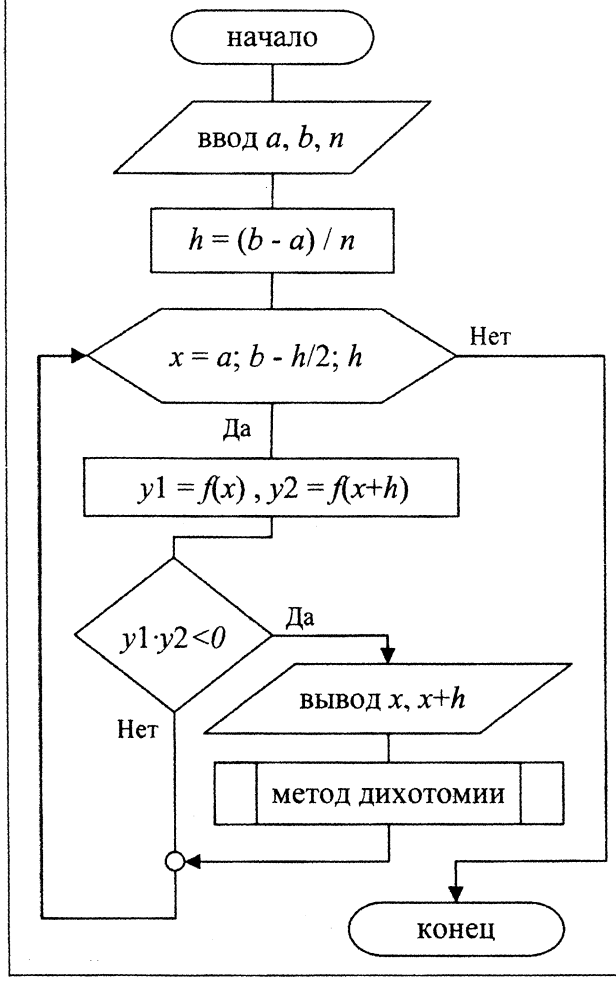
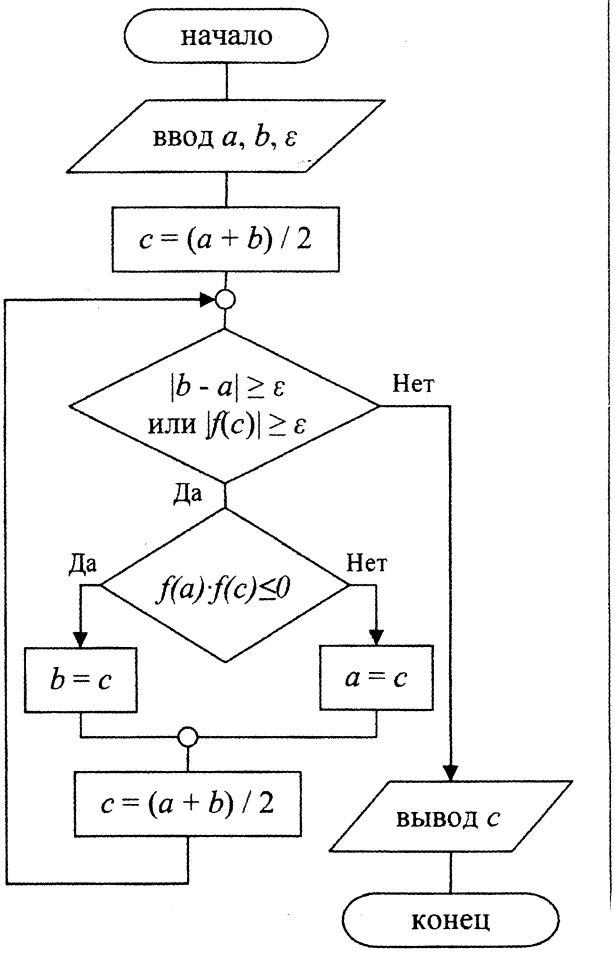


Рисунок 29 – Графическая интерпретация метода дихотомии

Аналитически, процедура «отбрасывания» происходит следующим образом:

$$\begin{cases} a = c, \text{ если } f(a)\cdot f(c) > 0 \text{ - рисунок 31(а)} \\ b = c, \text{ если } f(a)\cdot f(c) \le 0 \text{ - рисунок 31(б)} \end{cases}$$

Блок-схема алгоритма отделения отрезков, содержащих нули функции:	Блок-схема алгоритма метода дихотомии:
	

ЗАПИСЬ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ В EXCEL, VBA И MATHCAD

Математические константы

мат. запись	*Excel*	*MathCAD*	*VBA*
число π	ПИ()	π	4·Atn(1)
число e^1	EXP(1)	e^1	Exp(1)

Математические функции

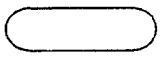
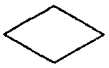
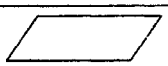
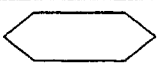
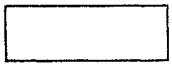
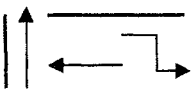
мат. запись	*Excel*	*MathCAD*	*VBA*	*мат. запись*	*Excel*	*MathCAD*	*VBA*
|x|	ABS(x)	|x|	Abs(x)	$\sqrt{x}$	КОРЕНЬ(x)	$\sqrt{x}$	Sqr(x)
lg x	LOG10(x)	log(x)	Log(x) / Log(10)	e^x	EXP(x)	e^x	Exp(x)
$\log_n x$	LOG(x;n)	log(x,n)	Log(x) / Log(n)	ln x	LN(x)	ln(x)	Log(x)
ctg x	1/ TAN(x)	cot(x)	1/ Tan(x)	cos x	COS(x)	cos(x)	Cos(x)
	COT(x)*			sin x	SIN(x)	sin(x)	Sin(x)
arctg x	ATAN(x)	atan(x)	Atn(x)	tg x	TAN(x)	tan(x)	Tan(x)
arcctg x	ПИ()/2–ATAN(x)	acot(x)	4·Atn(1)/2 - Atn(x)	arccos x	ACOS(x)	acos(x)	–
	ACOT(x)*			arcsin x	ASIN(x)	asin(x)	–

Примечание: * начиная с версии Excel 2013

Арифметические операции

мат. запись	*Excel*	*MathCAD*	*VBA*
$-a^n$	–(a^n)	$-a^n$	–(a^n)
$\sin^2 x$	SIN(x)^2	$\sin(x)^2$	Sin(x)^2
$\cos^2 x^3$	COS(x^3)^2	$\cos(x^3)^2$	Cos(x^3)^2
$\sqrt[n]{a}$	a^(1/n)	$\sqrt[n]{a}$	a^(1/n)
$\frac{a+b}{c \cdot d}$	(a+b)/(c*d)	$\frac{a+b}{c \cdot d}$	(a+b)/(c*d)

ОСНОВНЫЕ ЭЛЕМЕНТЫ БЛОК-СХЕМ

	Начало или конец алгоритма		Проверка условия
	Ввод или вывод информации		Организация циклов с параметрами
	Обработка данных (вычисление)		Порядок выполнения действий

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

Лабораторная работа № 1

ТЕМА. *Работа со списками в ЭТ Excel.*

Задание:

1. Открыть файл с исходной базой данных (далее – БД) **СписокЛР1.xls** и сохранить его на рабочий диск R:\Информатика\2 семестр\...

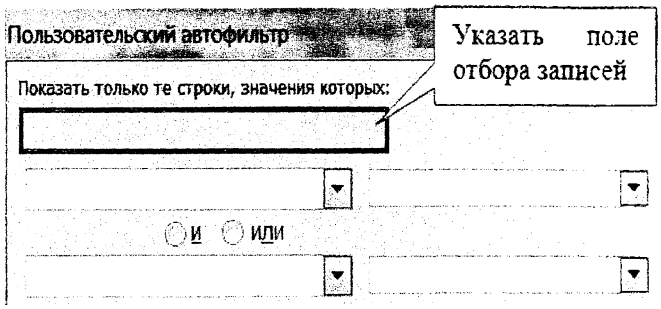
2. На рабочем листе **«Исходный_список»** добавить в список 5 новых записей с произвольными данными с помощью ***формы***.

3. Скопировать в текущей рабочей книге рабочий лист «Исходный_список» 6 раз. Каждому листу присвоить новое имя: ***Сортировка, Автофильтр1, Автофильтр2, Расш_фильтр, Итоги, Сводная таблица***.

4. На рабочем листе **«Сортировка»** отсортировать список по указанным полям.

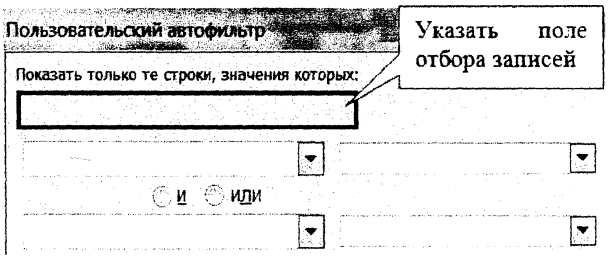
5. На рабочем листе **«Автофильтр1»**, используя возможности Автофильтра, выбрать записи по критериям для одного поля таблицы БД.

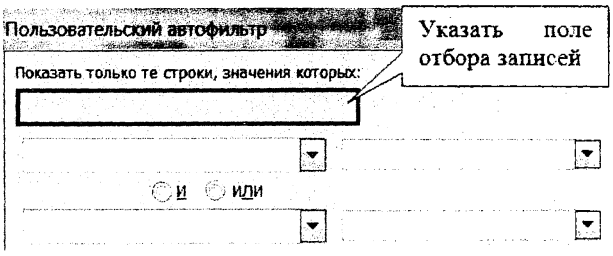
Условие: *Выбрать перевозки*



6. На рабочем листе **«Автофильтр2»**, используя возможности Автофильтра, выбрать записи по критериям для нескольких полей таблицы БД.

Условие: *Выбрать* _____





7. На рабочем листе **«Расш_фильтр»** (в отдельных диапазонах) используя возможности расширенного фильтра, сделать выборки:

7.1. по критериям из п. 5 (автофильтра1)

7.2. по критериям из п. 6 (автофильтра2)

7.3. по вычисляемому критерию

Условие: *Выбрать* _____

_____

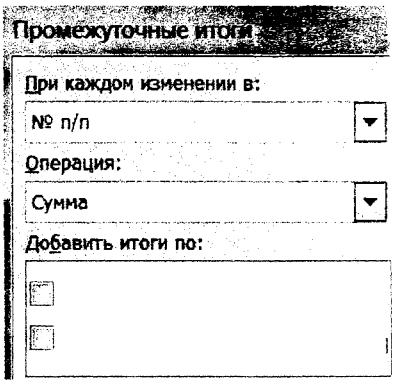
Вычисляемый критерий:_____ = _____.
(ячейка) (формула)

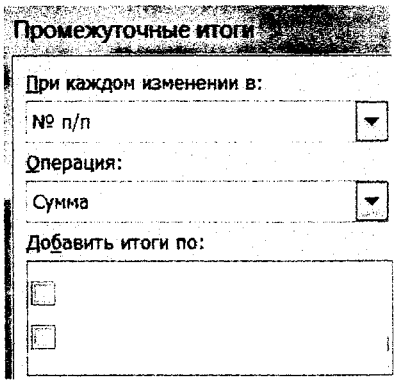
8. На рабочем листе **«Итоги»** подвести промежуточные итоги.

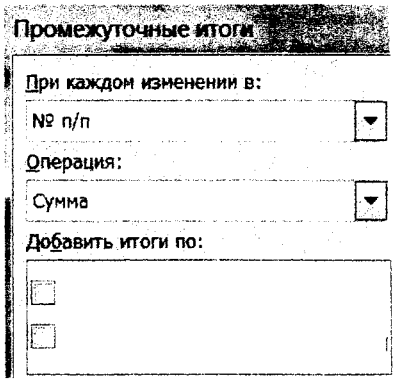
Условие: *Подсчитать* _____

_____

_____







9. На рабочем листе **«Сводная таблица»** построить сводную таблицу.

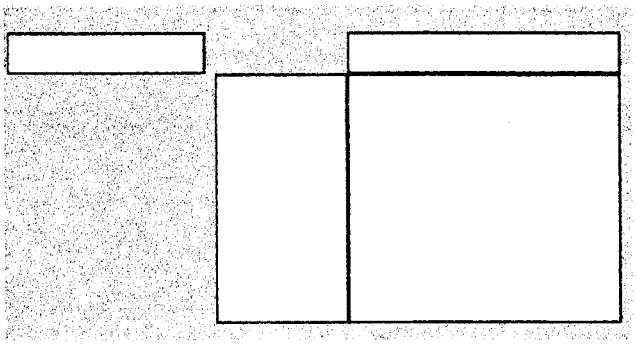
Условие:
Страница в разрезе:_____

Макет: _____

_____

_____

_____



Работа выполнена верно: (дата) _____ (подпись)_____

Задания на защиту лабораторной работы № 1

Задание 1. Выбрать записи по заданным критериям используя возможности

☐ Автофильтра,
☐ Расширенного фильтра.

Условие: _____

_____

_____

_____

Вычисляемый критерий: _____ = _____.
(ячейка) (формула)

выдано: _____ _____
(дата) (подпись)

Задание 2. Подвести промежуточные итоги.

Условие: _____

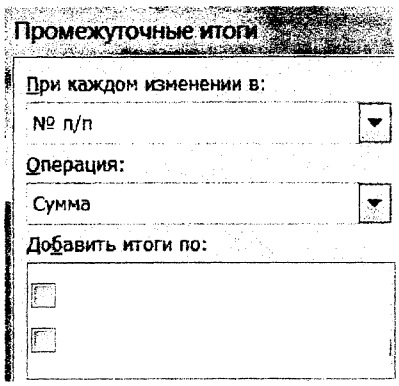
_____

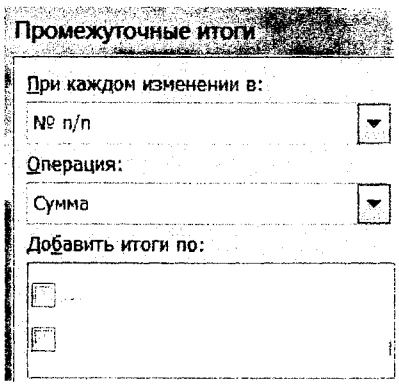
_____

_____

_____

_____





выдано: _____ _____
(дата) (подпись)

Задание 3. Построить сводную таблицу.

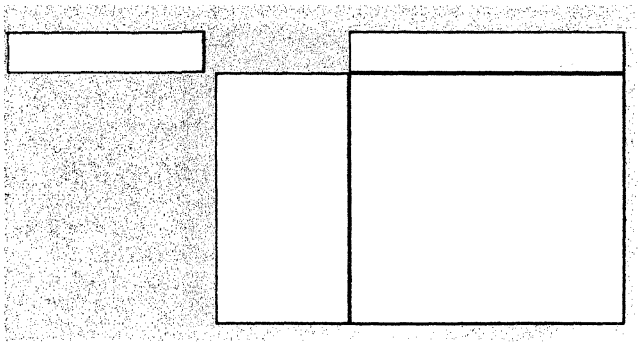
Условие:
Страница в разрезе: _____

Макет: _____

_____

_____

_____



выдано: _____ _____
(дата) (подпись)

Лабораторная работа № 2

ТЕМА. *Основы работы в VBA. Создание процедур с линейной структурой. Работа с макросами.*

ЧАСТЬ I. Процедуры-функции.

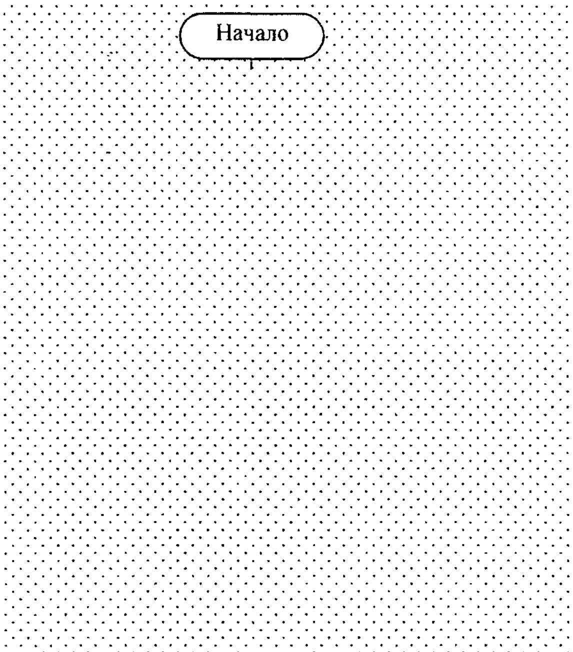
Задание:

1.1. Разработать линейную программу в виде процедуры-функции для вычисления значения выражения, зависящего от произвольных значений переменных ***a, b, c.*** Разбить выражение на простейшие части. Составить блок-схему алгоритма.

Условие:

Public Function

End Function



1.2. Вычислить в Excel значение полученной пользовательской функции, задав значения входным переменным в любых ячейках рабочего листа.

1.3. Получить значение заданного выражения обычной формулой в Excel.

	Параметры:	**a=**		Результат выражения (функция VBA):	
		b=			
		c=		Результат выражения (формула Excel):	

Замечание: Десятичные дроби переписываются с тремя знаками после запятой.

Формула Excel:______ = ____.
(ячейка) (формула)

1.4. Вычислить заданное выражение в СКМ MathCAD.

Ответ = ______________

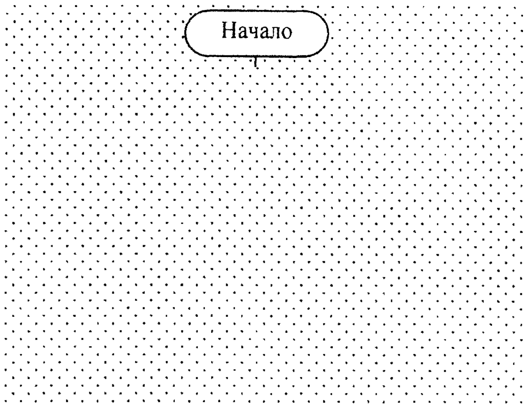
ЧАСТЬ II. Процедуры-подпрограммы.

Задание: Разработать линейный алгоритм в виде процедуры-подпрограммы для вычисления геометрической формулы. Составить блок-схему алгоритма. Исходные данные ввести в окно InputBox, а результат вывести в окно MsgBox. Закрепить выполнение процедуры за указанным элементом интерфейса.

Условие:	*Расчетная формула:*

Public Sub

End Sub



Значение исходных данных:	*Результат:*

ЧАСТЬ III. Макросы.

Задание: Записать макрос с помощью макрорекордера для форматирования ячейки: изменения размера и начертания шрифта, обрамления и заполнения любым цветом фона (желательно бледным оттенком). Закрепить макрос за комбинацией клавиш. Выполнить макрос для форматирования ячеек со значениями параметров и результатами вычислений выражения в ЧАСТИ I лабораторной работы.

Работа выполнена верно: (дата) ___________ (подпись)_______________

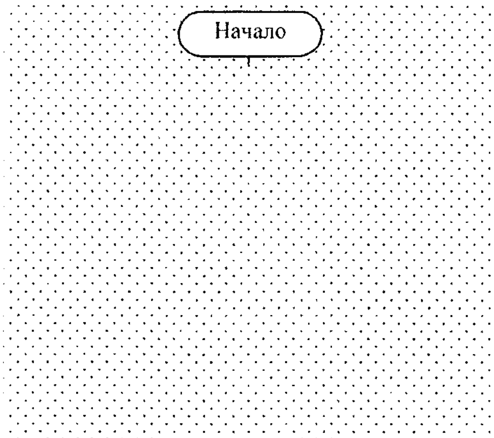
Задания на защиту лабораторной работы № 2

Задание 1. Разработать линейную программу в виде процедуры-функции для вычисления значения выражения. Составить блок-схему алгоритма.

Условие:

Public Function

End Function



выдано: __________ ________________
(дата) (подпись)

Задание 2. Разработать процедуру-подпрограмму для вычисления формулы. Составить блок-схему алгоритма. Исходные данные ввести в окно InputBox, а результат вывести в окно MsgBox. Закрепить выполнение процедуры за указанным элементом интерфейса.

Условие:	***Расчетная формула:***

Public Sub

End Sub

Значение исходных данных:

Результат:

выдано: __________ ________________
(дата) (подпись)

Лабораторная работа № 3

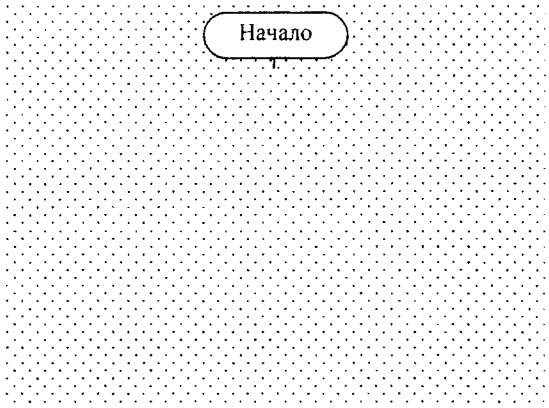
ТЕМА. ***Создание процедур с разветвляющейся структурой в Excel + VBA.***

ЧАСТЬ I. Простой многострочный оператор условного перехода.

Задание:

1.1. Создать процедуру-подпрограмму для решения квадратного уравнения ____$\mathbf{x^2}$ + ____**x** + ____ **= 0** с использованием простого многострочного оператора условного перехода **If…End If**. Составить блок-схему алгоритма. Коэффициенты уравнения считать с ячеек рабочего листа, полученные значения корней отобра-

зить в отдельных ячейках листа. Закрепить полученную процедуру за кнопкой на рабочем листе.



Public Sub

____

____

____

____

____

____

____

____

End Sub

1.2. Получить значения корней уравнения в Excel с помощью функции ***ЕСЛИ()*** – в случае отрицательного значения дискриминанта вывести в результирующих ячейках текст «нет корней».

1.3. Проверить значения корней уравнения в Excel с помощью средства «Подбор параметра».

	A	B	C	D	F	G	H
1	Коэффициенты:			Дискриминант:		D=	
2	a=						
3	b=			Корни (функция ЕСЛИ):			
4	c=			x_1 =		x_2 =	
5							
6		Корни ("Подбор параметра"):					
7		x_1 =			x_2 =		

<u>*Замечание:*</u> Десятичные дроби переписываются с тремя знаками после запятой.

H1 = ________________________________.
(формула)

F4 = ________________________________
(формула)

H4 = ________________________________
(формула)

D7 = ____________________, H7 = ____________________
(формула) (формула)

1.4. Выполнить решение уравнения в СКМ MathCAD с помощью функции ***root()***.

Листинг решения (в MathCad):

ЧАСТЬ II. Простой однострочный оператор условного перехода.

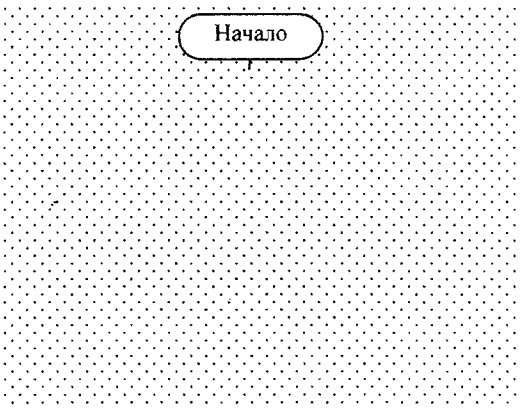
2.1. Создать процедуру-функцию для вычисления значений разветвляющейся функции (с двумя ветвями) при двух разных значениях аргумента (попадающих в разные интервалы) с использованием простого однострочного оператора условного перехода **If**. Составить блок-схема алгоритма.

Условие: $f(x) = \begin{cases} \\ \\ \\ \end{cases}$

Public Function

_____
_____
_____
_____
_____
_____
_____
_____

End Function



2.2. Получить значения функции в Excel с помощью функции ***ЕСЛИ( )***.

		x	***f(x) - функция ЕСЛИ***	***f(x) - пользов. функция***
	на 1-ой ветке			
	на 2-ой ветке			

Замечание: Десятичные дроби переписываются с тремя знаками после запятой.

Формула с ЕСЛИ:

_____ (ячейка) = _____ (формула).

_____

Формула с пользовательской функцией: _____ (ячейка) = _____ (формула).

2.3. Вычислить значения функции в СКМ MathCAD с помощью функции ***if( )***.

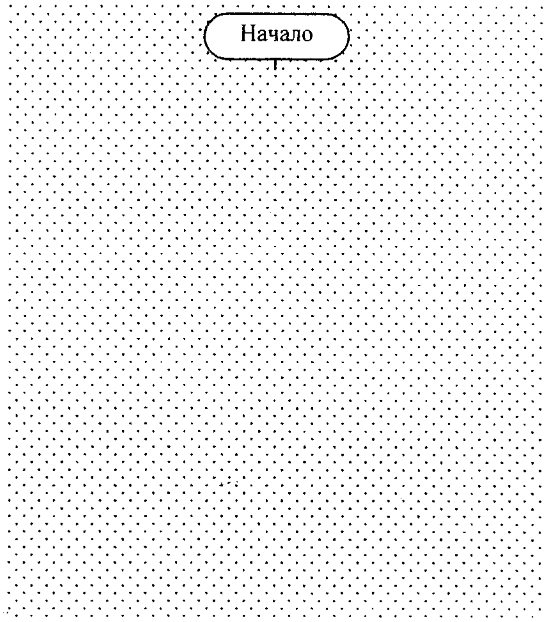
Листинг решения (в MathCad):

ЧАСТЬ III. Расширенный многострочный оператор условного перехода.

3.1. Создать процедуру-функцию для вычисления значений разветвляющейся функции (с тремя ветвями) при трёх разных значениях аргумента (попадающих в разные интервалы) с использованием расширенного многострочного оператора условного перехода **If...End If**. Составить блок-схему алгоритма.

Условие: $f(x) = \begin{cases} \\ \\ \end{cases}$

Public Function



End Function

3.2. Получить значения функции в Excel с помощью функции ***ЕСЛИ( )***.

		x	***f(x) - функция ЕСЛИ***	***f(x) - пользов. функция***
	на 1-ой ветке			
	на 2-ой ветке			
	на 3-ей ветке			

Замечание: Десятичные дроби переписываются с тремя знаками после запятой.

Формула с ЕСЛИ:

______ = ______________________________________.
(ячейка) (формула)

Формула с пользовательской функцией: ______ = ____________________.
(ячейка) (формула)

3.3. Вычислить значения функции в СКМ MathCAD с помощью ***программного модуля.***

Листинг решения (в MathCad):

Работа выполнена верно: (дата) _____ (подпись)_____

Задания на защиту лабораторной работы № 3

Задание 1. Найти корни квадратного уравнения ____x^2 + ____x + ____ = 0 в СКМ MathCAD с помощью функции ***root( ).***

Листинг решения (в MathCad):	

выдано: _____ _____
(дата) (подпись)

Задание 2. Создать пользовательскую функцию для вычисления значений функции при разных значениях аргумента (попадающих в разные интервалы) с использованием: ☐ простого однострочного оператора **If;** ☐ простого многострочного оператора **If;** ☐ расширенного многострочного оператора **If**

Условие:

$$f(x)=\begin{cases} \\ \\ \\ \\ \\ \\ \\ \end{cases}$$

Public Function

_____
_____
_____
_____
_____
_____
_____
_____

End Function

		x	*f(x)*
	на 1-ой ветке		
	на 2-ой ветке		
	на 3-ей ветке		

_____
(формула)

Замечание: Десятичные дроби переписываются с тремя знаками после запятой.

выдано: _____ _____
(дата) (подпись)

Задание 3. Вычислить значения разветвляющейся функции в СКМ MathCAD
☐ с помощью функции ***if( );*** ☐ с помощью программного блока.

Условие:

$$f(x) = \begin{cases} \\ \\ \\ \end{cases}$$

Листинг решения (в MathCad):

выдано: _____ _____
(дата) (подпись)

Лабораторная работа № 4

ТЕМА. ***Создание процедур с циклической структурой в Excel + VBA. Исследование функции одной переменной на отрезке.***

ЧАСТЬ I. Табулирование функции.

Задание:

1.1. Определить функцию ***y(x)***, вычислить её первую производную ***y1(x)*** и вторую производную ***y2(x)*** в СКМ MathCad.

1.2. Построить функции ***y(x)*** и ***y1(x)*** на одном графике на отрезке [*a*, *b*].

Условие:

y(x) = _____ a =____ b =____

Листинг решения (в MathCad):

тип диаграммы – «точечная» с гладкими кривыми
и маркерами
название диаграммы не указывать
название горизонтальной оси – «x»
название вертикальной оси – «y(x)»
легенду не добавлять
ориентировочный размер диаграммы – 9x16 см.

1.3. На отдельном рабочем листе EXCEL с названием **«Табулирование»** построить таблицу значений функции ***y*(*x*)**. Создать для этого процедуру-подпрограмму с использованием оператора цикла со счетчиком **For … Next**. Закрепить выполнение процедуры за командой на плавающей панели инструментов «Исследование функции». Составить блок-схему алгоритма.

Public Function

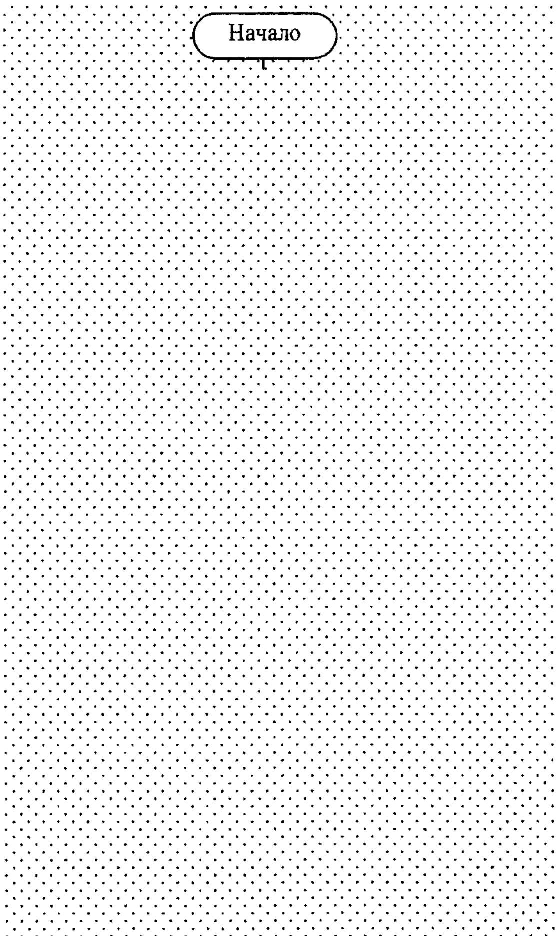
____

End Function

Public Sub

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

End Sub

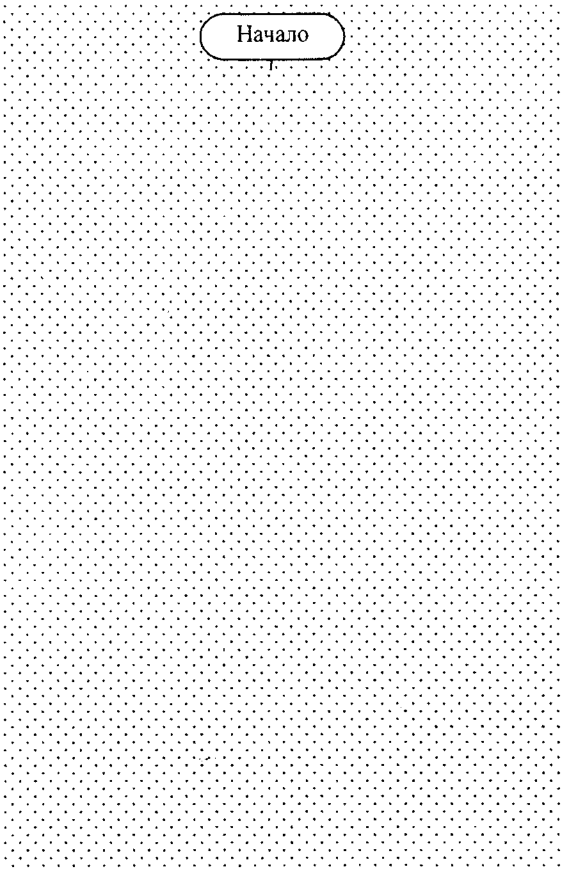


ЧАСТЬ II. Нахождение нулей функции.

2.1. На отдельном рабочем листе **«Корни»** сформировать таблицу для отделения отрезков, на которых находятся нули функции $y(x)$. Создать для этого процедуру-подпрограмму и закрепить ее за кнопкой на плавающей панели инструментов «Исследование функции». Составить блок-схему алгоритма.

Public Sub

End Sub



2.2. На рабочем листе **«Корни»** последовательно уточнить корни функции $y(x)$ на отделённых отрезках методом дихотомии (деления отрезка пополам) с заданной точностью ε = _________. Результаты вывести в сформированную в п. 2.1. таблицу отделенных отрезков. Для этого создать процедуру-подпрограмму, используя оператор цикла с условием. Закрепить процедуру за кнопкой на плавающей панели инструментов «Исследование функции». Составить блок-схему алгоритма.

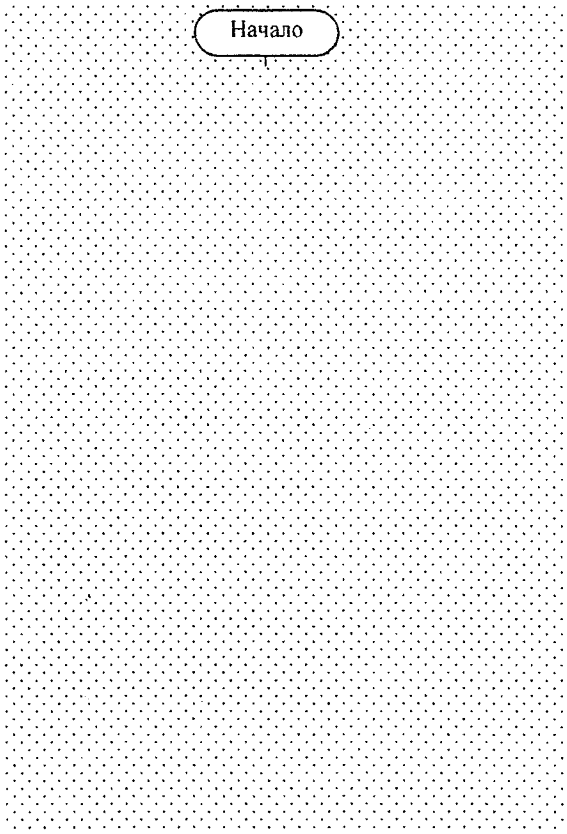
Замечание: Десятичные дроби переписываются с пятью знаками после запятой.

2.3. Найти нули функции $y(x)$ в СКМ MathCAD с помощью функции ***root( ).***

Листинг решения
(в MathCad):

Public Sub

End Sub



ЧАСТЬ III. Нахождение экстремумов функции.

3.1. На отдельном рабочем листе **«Экстремумы»** создать таблицу для отображения отрезков, содержащих локальные экстремумы функции $y(x)$ и сами точки максимума и минимума. Для этого создать процедуру-подпрограмму и закрепить ее за кнопкой на плавающей панели инструментов «Исследование функции».

Для количества разбиений отрезка n = _________ результат следующий:

Замечание: Десятичные дроби переписываются с пятью знаками после запятой.

3.2. Найти экстремумы функции $y(x)$ в СКМ MathCAD с помощью вычислительного блока ***Given… Maximize/Minimize***.

Листинг решения (в MathCad):

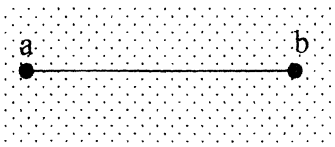
3.3. Найти экстремумы функции $y(x)$ в СКМ MathCAD с помощью производной.

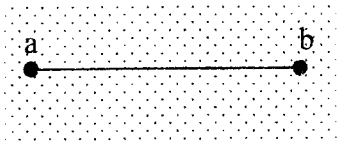
Листинг решения (в MathCad):

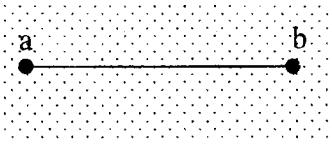
Работа выполнена верно: (дата) _____ (подпись)_____

Задания на защиту лабораторной работы № 4

Задание 1. Проиллюстрировать результаты первых трех шагов выполнения программы, реализующей уточнение _____ нуля функции из лабораторной работы.







выдано: _____ _____
(дата) (подпись)

Задание 2. Определить результаты первых трех шагов метода дихотомии для _____ нуля функции из лабораторной работы.

шаг 1	a =	s =	b =
шаг 2	a =	s =	b =
шаг 3	a =	s =	b =

выдано: _____ _____
(дата) (подпись)

Задание 3. Для функции $f(x)$=_____на отрезке [a =_____; b =_____]:

построить:
☐ график функции

найти:
☐ нули функции

найти:
☐ локальные минимумы функции
☐ локальные максимумы функции
с помощью
☐ блока *Given... Maximize/Minimize*
☐ производной

Листинг решения (в MathCad):

выдано: _____ _____
(дата) (подпись)

Лабораторная работа № 5

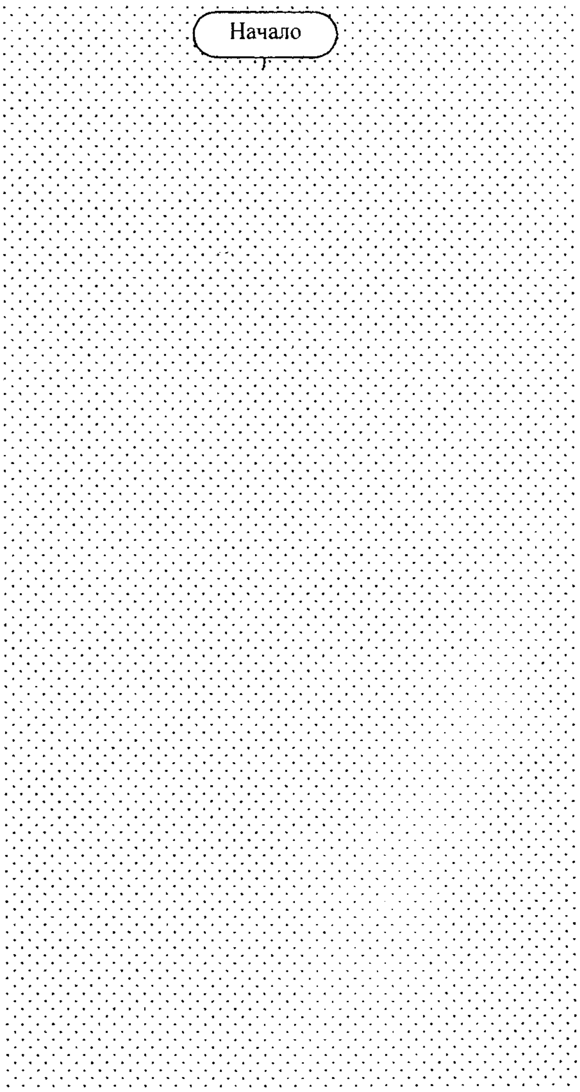
ТЕМА. *Работа с массивами в VBA.*

Задание:

1. В редакторе VBA разработать процедуры для обработки **одномерного** массива с количеством элементов 10. Ввод элементов организовать с помощью окна InputBox, а вывод массива и результат его обработки – на рабочий лист с комментариями. Для этого создать процедуру-подпрограмму с использованием оператора цикла со счетчиком **For … Next**. Закрепить процедуру за кнопкой на рабочем листе.

Public Sub

End Sub



2. В СКМ MathCAD, используя возможности системы, сформировать выражения (или программный блок) для решения задачи из п. 1.

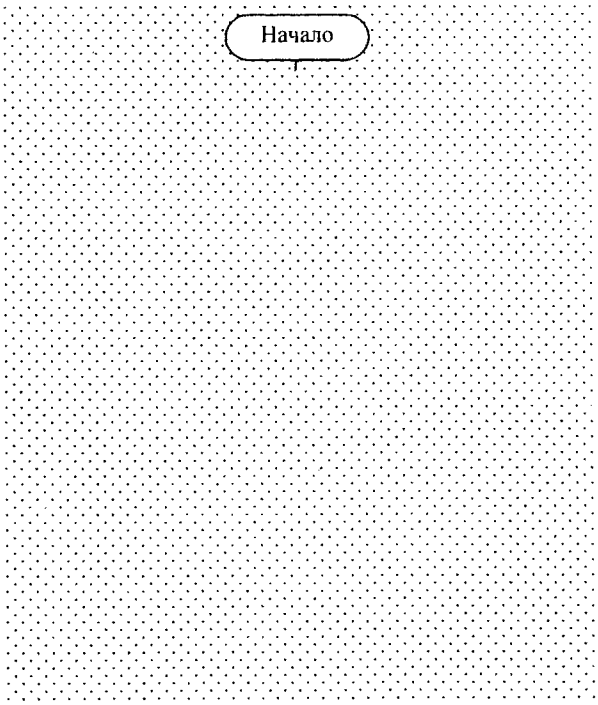
Листинг решения (в MathCad):

3. В редакторе VBA разработать процедуры для обработки **двухмерного** массива (3 × 3). Ввод элементов организовать с помощью окна InputBox, а вывод массива и результат его обработки – на рабочий лист с комментариями.

Для этого создать процедуру-подпрограмму с использованием оператора цикла со счетчиком **For … Next**.
Закрепить процедуру за кнопкой на рабочем листе.

Часть процедуры и блок-схема только алгоритма для обработки массива (без алгоритма ввода-вывода):

Public Sub
. . .
. . .
End Sub



Работа выполнена верно: (дата) _____ (подпись)_____

ОГЛАВЛЕНИЕ

Учебное издание

Составители:
Кофанов Валерий Анатольевич
Гучко Ирина Михайловна
Рамская Людмила Константиновна

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «Информатика» (II семестр)

для студентов специальности

«Машины и аппараты пищевых производств»

машиностроительного факультета

дневной формы обучения

Ответственный за выпуск: Кофанов В.А.
Редактор: Боровикова Е.А.
Компьютерная вёрстка: Кофанов В.А.
Корректор: Никитчик Е.В.

Подписано в печать 21.02.2018 г. Формат 60x84 $^{1}/_{8}$. Бумага «Performer».
Гарнитура «Times New Roman». Усл. печ. л. 5,58. Уч. изд. л. 6,0. Заказ № 233. Тираж 25 экз.
Отпечатано на ризографе учреждения образования «Брестский государственный технический университет». 224017, г. Брест, ул. Московская, 267.