МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению курсовой работы
по дисциплине «Информатика»
для студентов технических специальностей
дневной формы обучения

Брест 2014

УДК 681

Методические рекомендации представляют собой руководство по выполнению курсовой работы по дисциплине «Информатика» для студентов второго курса технических специальностей дневной формы обучения.

В пособии приводятся: задание курсовой работы, теоретический материал и рекомендации к её выполнению в среде табличного процессора Microsoft Excel+VBA и СКМ MathCad, а также список литературы для самостоятельного изучения.

Пособие имеет целью оказать помощь студентам в выполнении курсовой работы по названной дисциплине.

Составители: Гучко И.М., ст. преподаватель
Кулешова А.М., ст. преподаватель
Рамская Л.К., ст. преподаватель

ОГЛАВЛЕНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Тема курсовой работы

Исследование функции одной переменной на заданном отрезке (на базе ЭТ Excel + VBA и СКМ MathCAD)

1.2. Задание к курсовой работе

Исследование функции произвести на отрезке [a; b] в табличном процессоре MS Excel, в среде программирования Visual Basic for Applications (VBA) и системе компьютерной математики (СКМ) MathCAD. Количество точек разбиения отрезка **n = 20**.

Этапы исследования:

1. Анализ функции. Область определения функции.
2. Построение графика функции.
3. Поиск нулей функции.
4. Поиск экстремумов функции.

Перечисленные выше этапы исследования реализовать в указанных средах:

Электронная таблица Excel:

- ✓ Построить таблицу значений функции одной переменной на интервале **[a; b]** с шагом **h**. Отделить интервалы, на которых находятся нули функции.
- ✓ Построить график заданной функции.
- ✓ Уточнить один из корней функции *методом последовательного табулирования* и *методом Ньютона* с точностью $\varepsilon = 0{,}00001$.
- ✓ Найти значения всех корней с помощью средств *«Поиск Решения»* и *«Подбор параметра»* с указанной точностью.
- ✓ Найти локальные экстремумы функции. Определить глобальные максимум и минимум функции.

Каждое задание выполнить на отдельном листе. Название листа должно соответствовать пункту задания.

На первом листе поместить кнопку для инициализации пользовательской формы VBA-приложения.

Среда программирования Visual Basic for Application:

Организовать пользовательскую многостраничную форму для представления курсовой работы, на отдельных вкладках которой с помощью процедур и функций, реализовать алгоритм исследования заданной функции ***f(x)*** и разместить:

- ✓ паспорт программы;
- ✓ таблицы значений функции и значений первой и второй производных;
- ✓ график функции;
- ✓ нули функции, рассчитанные в VBA и полученные в Excel (для сравнения);
- ✓ локальные и глобальные экстремумы, рассчитанные в VBA и полученные в Excel (для сравнения).

Система компьютерной математики MathCAD:

- ✓ Построить таблицу значений функции одной переменной на интервале **[a; b]** с шагом **h**, используя ранжированную переменную.
- ✓ Построить график заданной функции.
- ✓ Найти локальные экстремумы функции. Определить глобальные максимум и минимум функции.

Выполнение каждого пункта задания сопроводить подробными комментариями.

1.3. Выбор варианта задания

Вариант выполнения курсовой работы указывается преподавателем.

1.4. Требования к содержанию курсовой работы

Курсовая работа выполняется студентом самостоятельно с использованием табличного процессора Excel, языка программирования VBA и СКМ MathCAD.

Объём и содержание курсовой работы должны быть достаточными для проверки знаний студента по изучаемой дисциплине в объеме курса.

Расчетно-пояснительная записка к курсовой работе должна содержать:

1. Титульный лист.
2. Бланк задания.
3. Содержание.
4. Введение – общие сведения о решении поставленной задачи на ЭВМ (реферат).
5. Постановка задачи.
6. Описание используемых методов и приемов.
7. Блок-схемы и описание процедур VBA.
8. Список используемых операторов.
9. Литература.
10. Приложение:

- распечатки рабочих листов табличного процессора Excel (с выводом заголовков строк и столбцов, в верхнем колонтитуле по правому краю – ФИО и номер группы);
- распечатки вкладок пользовательской формы;
- распечатки VBA-кодов процедур;
- распечатки результатов работы в СКМ MathCAD с комментариями;
- электронная версия курсовой работы.

1.5. Требования к оформлению курсовой работы

Пояснительная записка и приложения оформляются в соответствии с требованиями ГОСТ 2.105-79 ЕСКД (Единая система конструкторской документации): Общие требования к текстовым документам.

Весь текст пояснительной записки пишется от руки на стандартных листах формата А4. Текст должен иметь поля: слева – 30 мм, справа – 10 мм, сверху – 15 мм, снизу – 20 мм. Средняя плотность текста по вертикали 28 - 30 строк на страницу.

Титульный лист оформляется на компьютере с использованием одного из редакторов текста. Остальные приложения распечатываются на принтере.

Текст курсовой работы разделяется на разделы и подразделы. Разделы нумеруются арабскими цифрами с точкой, например, 1., 2. и так далее. Если в разделе есть подразделы, то они нумеруются в пределах данного раздела (1.1., 1.2. ...). Заголовки разделов и подразделов записываются заглавными буквами. Точка в конце заголовка не ставится. Заголовки отделяются от текста одной свободной строкой.

Все страницы курсовой работы нумеруются, на первой странице номер не ставится. Нумерация страниц должна быть сквозная, включая приложения.

Формулы, используемые в пояснительной записке, записываются в общем виде в отдельной строке и нумеруются. Номер формулы записывается арабскими цифрами у правого края листа и заключается в круглые скобки. Ниже формулы приводится пояснение имеющихся в ней символов. Описание каждого символа приводится в отдельной строке. Пояснения к символам, кроме последнего, заканчиваются точкой с запятой, а пояснения к последнему символу – точкой.

Ссылки на формулы в тексте пояснительной записки указываются в виде их номеров в круглых скобках.

Рисунки выполняются на отдельных листах. Все рисунки, так же как и формулы, нумеруются в пределах раздела.

Ориентация текста, таблиц и рисунков должна быть однотипной.

В тексте пояснительной записки разрешается использовать без расшифровки только общепринятые сокращения. Если используются нестандартные сокращения, то они должны быть расшифрованы в тексте при первом упоминании. Использовать сокращения отдельных слов не разрешается.

Схемы алгоритмов выполняются в соответствии с требованиями ГОСТ 19.003 80. Основные элементы схем алгоритмов приведены в таблице 1. Соотношение между геометрическими элементами схем установлены стандартом: размер ***a*** выбирается из ряда 10, 15, 20 мм. Допускается увеличивать размер ***a*** на число кратное ***5***. Размер ***b*** равен ***1,5a***, допускается устанавливать размер ***b*** равным ***2a***.

Список использованных источников составляется в алфавитном порядке или в порядке ссылок на источники в тексте. Номер источника проставляются в конце фразы или данных, заимствованных из соответствующего источника, и заключается в квадратные скобки, например [5].

Приложение должно начинаться с новой страницы. Если приложений несколько, то каждое из них должно начинаться с новой страницы. В верхнем правом углу пишется заглавными буквами слово «**ПРИЛОЖЕНИЕ**» и его номер, а в центре следующей строки пишется заглавными буквами название приложения.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Исследование функции на отрезке

2.1.1. 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. Этот процесс является обязательным этапом, предшествующим построению графика функции в табличном процессоре Excel.

2.1.2. Корни нелинейных уравнений (нули функции)

Нелинейными уравнениями называются уравнения вида

$$f(x) = 0, \qquad (1)$$

где функция $y = f(x)$ нелинейная:

– нелинейная алгебраическая функция вида $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$;

– трансцендентная функция (тригонометрическая, обратная тригонометрическая, логарифмическая, показательная или гиперболическая функция);

– функция, полученная комбинированием этих функций.

Решением нелинейного уравнения (1) называется такая точка x^*, которая путём подстановки в уравнение (1) превращает его в верное числовое равенство. Число x^* называют также нулём функции $f(x)$.

Процесс решения нелинейных уравнений предполагает реализацию двух этапов:

1) отделения корней нелинейных уравнений;

2) уточнения корней нелинейных уравнений.

Отделением корней нелинейного уравнения называется процесс выделения из области определения функции $y = f(x)$ отрезков **[a; b]**, в каждом из которых содержится *один и только один корень* уравнения $f(x) = 0$.

Отделение корней может быть выполнено следующими способами:

- ✓ *графическим методом:* строится график функции $y = f(x)$ на заданной области определения и выделяются отрезки, на которых функция меняет знак (т.е. точка пересечения графика с осью абсцисс является приближённым значением корня);
- ✓ *табличным способом (или методом табулирования):* строится таблица значений функции $y = f(x)$ на определённом промежутке изменения аргумента x (с некоторым, достаточно малым, шагом), и если окажется, что для соседних значений аргументов соответствующие значения функции имеют разные знаки, то корень уравнения $f(x) = 0$ находится между ними.

Если найден отрезок **[a; b]**, на котором имеется один корень функции, то задача нахождения отрезка **[a*; b*]**, содержащего корень и лежащего внутри отрезка **[a; b]**, такого, что его длина меньше заданной точности ε, называется ***задачей уточнения корня.***

Другими словами, уточнение значения корня с заданной точностью ε – это сужение границ отрезка **[a; b]** до длины, не превосходящей ε:

$$|b^* - a^*| \le \varepsilon. \quad (2)$$

При этом значением корня можно считать середину отрезка, то есть:

$$x^* = a^* + \frac{b^* - a^*}{2}. \quad (3)$$

На практике не всегда удаётся подобрать решение уравнения (1) точно. В этом случае решение находится с применением приближённых (итерационных) методов: метода простого табулирования, метода деления отрезка пополам (метод дихотомии), метода касательных (метод Ньютона) и др.

Итерационный процесс состоит в последовательном уточнении начального приближения x_0. Каждый такой шаг называется ***итерацией***. В результате итераций находится последовательность приближенных значений корня $x_1, x_2, ..., x_n$. Если эти значения с увеличением числа итераций ***n*** приближаются к истинному значению корня, то говорят, что итерационный процесс сходится.

Метод простого табулирования не имеет ограничений, но требует большого числа ручных операций.

Алгоритм реализации данного метода следующий:

- ✓ на отрезке **[a; b]** строится таблица значений функции с шагом $h = \frac{b-a}{n}$;
- ✓ по таблице определяется отрезок **[a1; b1]** $\subset$ **[a; b]**, на концах которого значения функции разных знаков;
- ✓ следующая таблица значений функции строится на отрезке **[a1; b1]** с шагом $h1 = \frac{b1-a1}{n}$;
- ✓ по таблице определяется отрезок **[a2; b2]** $\subset$ **[a1; b1]**, на концах которого значения функции разных знаков;
- ✓ процесс повторяется до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не станет меньше заданной точности ε, т.е. выполняется условие (2);
- ✓ тогда в качестве значения корня функции можно принять середину отрезка и окончательно рассчитать его по формуле (3).

Метод касательных (метод Ньютона)

Метод Ньютона или касательных заключается в том, что если x_n – некоторое приближение к корню x^* уравнения (1), то следующее приближение определяется как корень касательной к функции $f(x)$, проведенной в точке x_n.

Данный метод самый эффективный, поскольку обеспечивает сходимость за минимальное число шагов. Однако этот метод накладывает серьезные ограничения на вид функции, которая должна быть дважды дифференцируема.

Для поиска корня в этом методе составляется рекуррентная формула:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad (4)$$

Начальное приближение выбирается на одной из границ отрезка отделения корня. В качестве начального приближения x_0:

- ✓ выбирается граница ***b***, если

$$f'(x) * f''(x) > 0; \quad (5)$$

- ✓ выбирается граница ***a***, если

$$f'(x) * f''(x) < 0. \quad (6)$$

Алгоритм применения метода Ньютона:

- ✓ определяется начальное приближение x_0;
- ✓ составляется расчетная таблица с последовательным построением итераций по формулам Ньютона (4): для первого шага в качестве аргумента используется начальное приближение x_0, для последующих шагов – значение корня на предыдущем шаге, т.е. значение x_i;
- ✓ условием окончания процедуры уточнения корня является достижение функцией значения, меньше заданного.

Метод половинного деления или дихотомии

Дихотомия – сопоставленность или противопоставленность двух частей целого.

Метод дихотомии состоит в делении пополам отрезка [a; b], на котором находится корень уравнения (1). Затем анализируется изменение знака функции на половинных отрезках, и одна из границ отрезка [a; b] переносится в его середину, причем та, со стороны которой функция на половине отрезка знака не меняет. Далее процесс повторяется. Итерации прекращаются при выполнении одного из условий: либо длина интервала [a; b] становится меньше заданной погрешности нахождения корня ε, либо значение функции сравнимо с погрешностью расчетов. Данный метод не требует дифференцируемости функции и легко реализуется на ЭВМ.

Алгоритм применения метода дихотомии:

- ✓ отрезок [a; b] делится пополам точкой $c = \frac{a+b}{2}$;
- ✓ если $f(c) \neq 0$, то возможны два случая: $f(x)$ меняет знак на отрезке [a; c] или $f(x)$ меняет знак на отрезке [c; b];
- ✓ выбираем тот отрезок, на котором функция меняет знак: если $f(x)$ меняет знак на отрезке [a; c], то b = c; если $f(x)$ меняет знак на отрезке [c; b], то a = c;
- ✓ процесс повторяется до тех пор, пока длина отрезка [a; b] не станет меньше заданной точности ε; т.е. выполняется условие (3);
- ✓ тогда в качестве значения корня функции можно принять середину отрезка и окончательно рассчитать по формуле $\frac{a+b}{2}$.

2.1.3. Экстремумы функции

Точка экстремума – это точка, в которой достигается максимальное или минимальное значение функции в некоторой области определения ее аргументов.

Точка x_0 называется **точкой локального максимума** функции $f(x)$, если существует такая окрестность этой точки, что для всех x из этой окрестности выполняется неравенство: $f(x) \le f(x_0)$.

Точка x_0 называется **точкой локального минимума** функции $f(x)$, если существует такая окрестность этой точки, что для всех x из этой окрестности $f(x) \ge f(x_0)$.

Первое достаточное условие экстремума

Пусть для функции $y = f(x)$ выполнены следующие условия:

1) функция непрерывна в окрестности точки x_0;

2) первая производная $f'(x_0) = 0$ или $f'(x_0)$ не существует;

3) производная $f'(x_0)$ при переходе через точку x_0 меняет свой знак.

Тогда в точке $x = x_0$ функция $y = f(x)$ имеет экстремум, причем это:

- ✓ *минимум*, если при переходе через точку x_0 производная меняет свой знак с минуса на плюс;
- ✓ *максимум*, если при переходе через точку x_0 производная меняет свой знак с плюса на минус.

Если производная $f'(x)$ при переходе через точку x_0 не меняет знак, то экстремума в точке $x = x_0$ нет.

Таким образом, для того чтобы исследовать функцию $y = f(x)$ на экстремум, необходимо:

- ✓ найти производную функции $f'(x)$;
- ✓ найти критические точки, то есть такие значения x, в которых $f'(x) = 0$ или $f'(x)$ не существует;
- ✓ исследовать знак производной слева и справа от каждой критической точки;
- ✓ найти значение функции в точках экстремума.

Второе достаточное условие экстремума

Пусть для функции $y = f(x)$ выполнены следующие условия:

1) она непрерывна в окрестности точки x_0;

2) первая производная $f'(x) = 0$ в точке x_0;

3) $f''(x) \ne 0$ в точке x_0.

Тогда в точке x_0 достигается экстремум, причем, если $f''(x) > 0$, то в точке $x = x_0$ функция $y = f(x)$ имеет *минимум*; если $f''(x) < 0$, то в точке $x = x_0$ функция $y = f(x)$ достигает *максимума*.

Т.о., для того чтобы исследовать функцию $y = f(x)$ на экстремум, необходимо:

- ✓ найти первую и вторую производные функции – $f'(x)$ и $f''(x)$;
- ✓ найти критические точки, то есть такие значения x, в которых $f'(x) = 0$ или $f'(x)$ не существует;
- ✓ исследовать знак второй производной каждой критической точки;
- ✓ найти значение функции в экстремальных точках.

Наибольшее значение функции $f(x)$ в некоторой области определения ее аргументов называют ***глобальным максимумом***, а её наименьшее значение – ***глобальным минимумом***.

Чтобы найти глобальные экстремумы функции $f(x)$ на отрезке **[a; b]**, на котором она непрерывна, надо:

- ✓ вычислить значения функции в критических точках отрезка;
- ✓ вычислить значения функции в граничных точках отрезка, т.е. $f(a)$ и $f(b)$;
- ✓ из всех полученных значений выбрать наименьшее и наибольшее.

Для нахождения экстремумов функции одной переменной можно использовать численные методы, такие как метод половинного деления, метод золотого сечения и др.

Метод золотого сечения

Пусть задана функция $f(x)$ на отрезке [a; b]. Для нахождения экстремумов функции на заданном отрезке рассматриваемый отрезок делится в пропорции золотого сечения в обоих направлениях, то есть выбирается точка $x1$, такая что:

$$\frac{b-a}{b-x1} = \frac{b-x1}{x1-a} = t = 1.618, \tag{7}$$

где t – пропорция золотого сечения.

И точка $x2$, зеркальная точке $x1$ относительно середины отрезка [a; b]:

$$x2 = b - (x1 - a). \tag{8}$$

Точка $x1$ является золотым сечением отрезка [a; b], если отношение длины всего отрезка $(b-a)$ к длине большей части $(b-x1)$ равно отношению длины части отрезка к длине $(x1-a)$ меньшей части.

Аналогично, точка $x2$, симметричная точке $x1$ относительно середины отрезка [a; b], является вторым золотым сечением этого отрезка. Т.к. точки $x1$ и $x2$ расположены симметрично относительно середины отрезка [a; b], то можно записать формулы:

$$x1 = b - \frac{(b-a)}{t}, \quad x2 = a + \frac{(b-a)}{t}. \tag{9}$$

По определению золотого сечения, учитывая выражение зеркальности $(b-x1) = (x2-a)$, соотношение (7) переписывается в виде:

$$t = \frac{b-x1}{x1-a} = \frac{x2-a}{x1-a}. \tag{10}$$

То есть точка $x1$ делит отрезок [a; x2] в отношении золотого сечения. Аналогично $x2$ делит отрезок [x1; b] в той же пропорции. Это свойство и используется для построения итерационного процесса.

Алгоритм реализации метода золотого сечения:

- ✓ на первой итерации заданный [a; b] отрезок делится двумя симметричными относительно его центра точками $x1$ и $x2$ по формулам (9);
- ✓ рассчитываются значения функции в этих точках: $y1 = f(x1)$ и $y2 = f(x2)$;
- ✓ отбрасывается тот из концов отрезка, к которому среди двух вновь поставленных точек ближе оказалась та, значение в которой максимально (для случая поиска минимума):

  если $y1 < y2$, то $b = x2$, $x2 = x1$, $x1 = a + (b - x2)$, $y2 = y1$, $y1 = f(x1)$,

  иначе $a = x_1$, $x_1 = x_2$, $x_2 = b - (x_1 - a)$, $y_1 = y_2$;
- ✓ на следующей итерации в силу показанного выше свойства золотого сечения уже ищется всего одна новая точка;
- ✓ процедура продолжается до тех пор, пока не будет достигнута заданная точность.

2.2. Основы программирования

2.2.1. Алгоритмизация и программирование

Каждая программа проходит от момента своего появления до списания определенный жизненный цикл. Основными этапами этого цикла являются: содержательная постановка задачи, математическая постановка задачи, выбор метода решения, разработка математической модели, разработка схемы алгоритма, написание программы на одном из языков программирования, отладка программы, сдача программы в эксплуатацию и научно-техническое сопровождение программы.

Содержательная постановка задачи заключается в формулировке задачи на естественном языке.

Математическая постановка задачи сводится к точному описанию исходных данных, условий задачи и целей ее решения с использованием математических выражений в общем виде.

Математическая модель задачи представляет собой совокупность математических выражений, описывающих данную задачу.

Метод решения задачи выбирается из известных методов. При отсутствии известных методов, разрабатывается свой, оригинальный метод решения. После выбора метода решения осуществляется формализация задачи, или, иными словами, описание исходных данных и условий задачи в виде, удобном для ввода в ЭВМ, неформализованные условия задачи представляются в математической форме.

Схема алгоритма является одним из способов наглядного представления алгоритма с помощью специальных элементов, предусмотренных ЕСКД. Некоторые из этих элементов приведены в таблице 1.

Алгоритм – точное, однозначное предписание последовательности действий (операций), приводящее к решению задач данного класса за конечное число шагов или заданное время.

Написание программы на одном из языков программирования заключается в описании схемы алгоритма с помощью операторов и команд соответствующего языка высокого уровня.

Отладка программы заключается в проверке правильности функционирования алгоритма решения задачи с помощью контрольных примеров, результаты решения которых заведомо известны, и в устранении обнаруженных синтаксических и логических ошибок.

Научно-техническое сопровождение программы предусматривает контроль за результатами работы программы и устранение ошибок, обнаруженных в процессе эксплуатации, доработке программы и ее совершенствовании в соответствии с требованиями заказчика.

При разработке программы необходимо руководствоваться принципами структурного программирования. К ним относятся нисходящая разработка, модульное проектирование и использование базовых структур.

При реализации принципа нисходящей разработки программа разрабатывается в следующей последовательности:

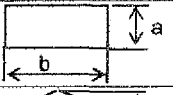
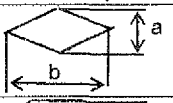
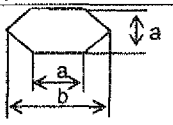
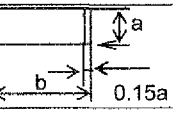
1. На основе анализа задача разбивается на подзадачи, выделяются уровни и подуровни, составляется иерархическая структура программы;

2. Для каждого уровня определяется:

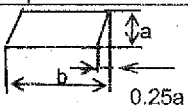
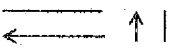
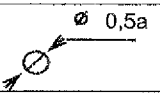
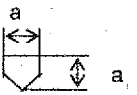
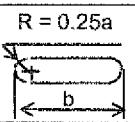
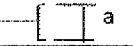
- ✓ метод решения и разрабатывается математическая модель задачи;
- ✓ определяются основные блоки программы и разрабатывается укрупненная схема алгоритма;
- ✓ определяются входные и выходные переменные, общие для данного уровня;
- ✓ разрабатываются схемы алгоритмов для реализации основных блоков программы, определяются входные и выходные переменные соответствующего уровня.

Этот процесс продолжается, пока схема алгоритма не будет доведена до базовых структур соответствующего языка программирования. Базовые элементы схем алгоритмов для всех языков программирования в основном совпадают. Отличия могут заключаться только в форматах используемых операторов. Основные базовые элементы схем алгоритмов приведены в таблице 1.

Таблица 1

Наименование	*Обозначение*	*Функция*
Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды, или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы)
Предопределенный процесс		Использование созданных ранее или отдельно описанных алгоритмов или программ

Продолжение таблицы 1

Ввод – вывод		Преобразование данных в форму, пригодную для обработки(ввод) или отображения результатов обработки (вывод)
Линии потока		Указание на последовательность связи между символами. Можно без стрелки, если линия направлена слева направо или сверху вниз, со стрелкой в остальных случаях
Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы в пределах листа
Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы на разных листах
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементами схемы с пояснениями

2.2.2. *Программирование в VBA*

Общие сведения о языке программирования

Visual Basic для приложений (VBA) является инструментальным средством разработки приложений в среде основных компонентов Office: Word, Excel, PowerPoint, Access, FrontPage и Outlook. По замыслу разработчиков он может использоваться именно как средство разработки приложений, а не только в качестве инструмента настройки пользовательского интерфейса и редактирования макросов.

Язык программирования VBA является диалектом одного из самых популярных и мощных универсальных языков программирования Visual Basic (VB). Основное различие между ними формулируется следующим образом: проекты VBA выполняются только с помощью приложения, поддерживающего VBA, в то время как VB позволяет создавать полностью автономные приложения. С другой стороны, синтаксис языков VBA и VB практически одинаков. Оба языка имеют почти одинаковые интегрированные среды разработки.

Общие принципы объектно-ориентированного программирования

VBA является объектно-ориентированным языком, предоставляющим возможности визуального программирования, содержит иерархию объектов, каждому из которых соответствует свой набор методов и свойств. Объекты представляют собой фундаментальные «строительные» блоки – почти все, что делается в среде VBA, включает модификацию объектов.

Объект – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его свойства, и некоторые методы

для управления объектом. В VBA имеется много встроенных объектов, например, *Range* (диапазон ячеек или одна ячейка), *Cell* (ячейка), *Worksheet* (рабочий лист), *DialogSheet* (диалоговое окно).

Каждый объект обладает некоторыми характеристиками или ***свойствами***. Например, свойством объекта может быть *шрифт*, используемый для отображения информации в ячейке (объекте) рабочего листа.

Объект содержит также список методов, которые к нему применимы.

Методы – это то, что можно делать с объектом. Например, можно вывести диалоговое окно на экране или убрать его.

Большинство объектов принадлежит к группе подобных объектов. Эти группы называются ***наборами***. Например, все рабочие листы рабочей книги образуют набор, называемый *WorkSheets*.

Кроме того, для каждого объекта определен ряд возможных ***событий***. Каждое событие представляет собой извещение, которое генерируется в результате действия пользователя или изменением состояния приложения или системы. Например, *щелчок или двойной щелчок мыши, нажатие клавиши, перетаскивание объекта и т.д.*

Установка значений свойств объекта: **Объект.Свойство = Выражение**

ПРИМЕРЫ:

В ячейку A1 записывается число 2,1:

Range("A1").Value = 2.1

В ячейку A2 вставляется формула:

Range("A2").Formula = "=СУММ(A1:C1)"

Свойству Row (номер строки) элемента управления MSFlexGrid (с именем **Grid***) присваивается значение переменной i:*

Grid.Row = i

Чтение свойств объекта: **Переменная = Объект.Свойство**

ПРИМЕРЫ:

Переменной **x** *присваивается значение из ячейки B4:*

x = Range("B4").Value *или* **x = Cells(4,2).Value**

Переменной E присваивается содержимое элемента управления TextBox с именем Text1:

E = Text1.Text

Использование методов: **Объект.Метод**

ПРИМЕРЫ:

Очистить диапазон ячеек A1:C5:

Range("A1:C5").Clear

Выделить ячейку D6:

Cells(6,4).Select

Показать форму:

UserForm.Show

Скрыть форму:

UserForm.Hide

Общие принципы построения VBA-программы

Программный код VBA состоит из следующих «строительных блоков»:

Ключевые слова – это элементы языка программирования, *например, If, Else, Sub, Select Case...*

Выражения – это комбинации ключевых слов, операторов, элементов и констант, результатами которых являются строка, число или логическое значение.

Переменные – используются для сохранения значений величин, изменяющихся в процессе выполнения программы. Каждая переменная имеет имя, по которому к ней обращаются. Присвоение значения для переменной осуществляется посредством оператора присваивания. В левой части оператора присваивания – имя переменной, а в правой – значение или выражение:

*X = 200 * 0.8 / 70*

Y = "Петров Семен Иванович"

Z = X / 80 + 30

Константа – поименованный элемент, сохраняющий постоянное значение в течение выполнения программы.

Литерал – простое выражение, определяющее значение простого типа (числовой литерал, строковый литерал и литерал даты).

Операторы – используются для объединения простых выражений в сложные (могут оперировать литералами).

Комментарии – служат для краткого описания по тексту программы:

' комментарии

Процедура – это отдельная единица программного кода VBA, которую можно вызвать по имени для выполнения определенных действий. Процедура может иметь параметры и в результате выполнения последовательности инструкций изменять их значения. Любая процедура содержит один или несколько операторов. Существует два вида процедур:

- ✓ Подпрограммы (**Sub**) – выполняют один или несколько операторов и не возвращают значение как функция (пример – текст макроса).
- ✓ Функции (**Function**) – либо вычисляют какое-либо значение и возвращают результат вычисления, либо тестируют что-либо, возвращая в результате – ИСТИНА или ЛОЖЬ.

Модуль – это именованная единица, состоящая из одной или нескольких процедур и раздела описаний, в котором объявляются переменные, константы и пользовательские типы данных, а также устанавливаются параметры компилятора (рис. 1).

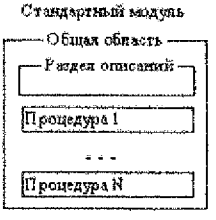


Рисунок 1

Форма – это объект, с помощью которого можно создавать диалоговые окна для взаимодействия с пользователем.

Проект – включает в себя все модули, формы и связанные с приложением объекты, относящиеся к конкретному документу (в MS Excel – это рабочая книга), причём проект сохраняется вместе с самим этим документом.

Правила объявления констант и переменных

Объявление констант:

[Private | Public] Const <Имя_конст> As <тип = значение>

Объявление переменных:

Dim <Имя переменной> [As <Тип данных>]

Тип данных (*data type*) – это термин, относящийся к определённым видам данных, которые VBA сохраняет и которыми может манипулировать. Как и большинство других систем программирования, VBA разделяет обрабатываемые данные на числа, даты текст и др. (таблица 2).

Таблица 2

Тип данных	*Хранимые значения*
Boolean	Логическая величина
Byte	Целое положительное число
Integer	Целое число
Long	Целое число двойной длины
Single	Число с плавающей точкой
Double	Число с плавающей точкой двойной точности
Currency	Число с фиксированной точкой
Date	Дата
String	Строковое значение

ЗАМЕЧАНИЯ:

а) Указание типа данных в инструкции описания не является обязательным. Если тип данных не указан, то по умолчанию переменная получит тип Variant.

б) В языке VB действуют следующие соглашения на ***ИМЕНА*** *процедур, переменных и констант, которые:*

- ✓ *могут иметь не более 255 символов, первые три – должны быть буквами;*
- ✓ *могут включать буквы, цифры и символы подчеркивания;*
- ✓ *не должны включать знаки препинания («.», «,»), пробелы, а также символы «!», «@», «$», « #»;*
- ✓ *не должны совпадать с ключевыми словами языка Visual Basic.*

с) Переменные объявленные в процедуре, можно использовать только внутри этой процедуры. Если переменная должна быть доступна во всех процедурах одного модуля, она объявляется в области описания модуля. Здесь также можно применять оператор **Dim**. *Наибольшую область действия имеет переменная, объявленная глобальной (с префиксом* **Public**) *в области описания модуля.*

ПРИМЕРЫ:

- ✓ переменные **X** и **Y** описываются с типом данных *Variant*, а переменная **Z** с типом данных *Integer*:

Dim X, Y, Z As Integer

- ✓ создается переменная **X** и указывается для нее текстовый (строковый) тип данных:

Public X As String

Создание и выполнение процедуры в VBA

Порядок создания процедуры:

1) открываем *редактор VB*: ***п. м. Сервис → Макрос → Редактор Visual Basic;***

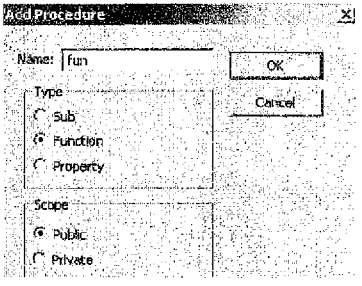
2) создаём новый *модуль* для текущей книги Excel – ***п. м. Insert → Modul;***

3) создаём новую *процедуру* – ***п. м. Insert → Procedure →*** в появившемся диалоговом окне ***Add Procedure*** (рис. 2) выполняем следующее:

- ✓ задаём ***имя функции*** (в поле **Name**);
- ✓ выбираем тип процедуры (в группе переключателей **Type**) из возможных вариантов: **Sub** *(подпрограмма)*, **Function** *(функция)*, **Property** *(свойство)*;
- ✓ задаём область видимости (в группе переключателей **Scope**) – ***Privat, Public.***

После выполнения пункта 3 в окне ***Code Window*** (окно просмотра исходного кода VBA) появятся соответствующие ключевые слова (рис. 2).

для функции

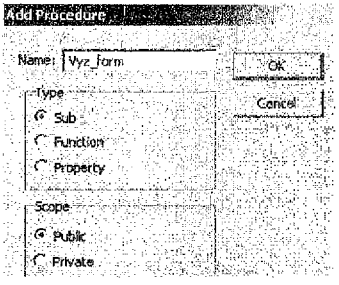


Public Function fun()

...

End Function

для подпрограммы



Public Sub Vyz_form()

...

End Sub

Рисунок 2

4) выполняем «отладку» кода, для чего вызываем команду ***п. м. Debug → Compile VBAProject;***

5) сохраняем модуль вместе с файлом Excel – ***п. м. File → Save.***

ЗАМЕЧАНИЯ:

а) чтобы вернутся в Excel, не закрывая редактор VBA, набрать ALT+F11.

б) служебные слова Private и Public задают область видимости процедур:

- ✓ ***Private*** *указывает, что процедура доступна для других процедур только того модуля, в котором она объявлена;*
- ✓ ***Public*** *указывает, что процедура доступна для всех других процедур во всех модулях проекта;*

с) для удаления модуля необходимо его выделить в дереве проекта, вызвать команду ***п. м. File → Remove*** *и ответить «Нет» в появившемся окне сообщения (рис. 3).*

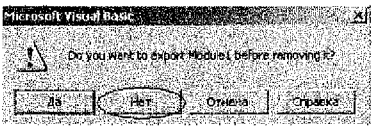


Рисунок 3

Структура процедуры:

для функции:

[Private | Public] Function <Имя>(<Аргументы>)<Описание функции>
<Тело процедуры>
End Function

для подпрограммы:

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

ЗАМЕЧАНИЕ: *в* ***теле процедуры*** *задается текст программы.*

Для выполнения *процедур-подпрограмм* используются следующие способы:

1 способ: находясь в редакторе VB – клавиша F5 или команда ***п. м. Run → RunSub***;

2 способ: находясь в рабочей книге Excel – ***п. м. Макрос → Макросы*** *→ выбрать имя процедуры* → **Выполнить** (рис. 4);

3 способ: в рабочей книге Excel *закрепить процедуру за* ***командной кнопкой***.

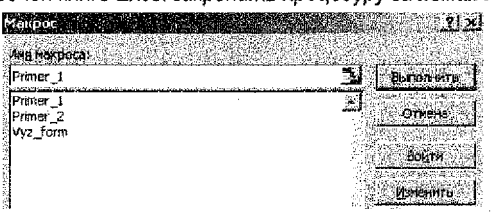


Рисунок 4

Для вызова *процедуры-функции* в ячейке рабочего листа вызывается *Мастер функций* командой ***п. м. Вставка → Функция***, в появившемся окне (рис. 5) из категории *Определенные пользователем* выбирается необходимая функция и на следующем шаге заполняются данными поля аргументов.

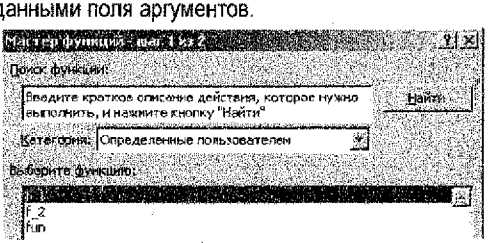


Рисунок 5

Оператор условного перехода If...Then...Else

VBA-оператор ***If...Then...Else*** позволяет задавать выполнение одной или нескольких альтернативных групп инструкций в зависимости от определённого условия. Данный оператор имеет однострочный и многострочный синтаксис.

Синтаксис **простого однострочного** условного оператора:

IF <*условие*> **THEN** <*операторы*>

*При выполнении оператора **IF** проверяется **условие** и, если оно истинно, то выполняется действие, указанное после оператора **THEN**. Если выражение ложно, то управление передается на оператор, следующий за оператором **IF**.*

Синтаксис **простого расширенного однострочного** условного оператора:

IF <*условие*> **THEN** <*операторы1*> **ELSE** <*операторы2*>

*При выполнении оператора **IF**, если условие истинно, выполняются операторы, указанные после оператора **THEN**, в ином случае выполняются операторы, следующие за оператором **ELSE**. После выполнения соответствующей группы операторов управление передается на оператор, следующий за оператором **IF**.*

ЗАМЕЧАНИЕ: после операторов ***THEN*** и ***ELSE*** может быть указано несколько операторов, разделенных *двоеточием*. Однако число операторов ограничено длиной строки.

Формат и блок-схема многострочного простого условного оператора указаны на рис. 6. Формат и блок-схема многострочного расширенного условного оператора указаны на рис. 7.

IF <*условие*> **THEN**
    <*первая группа операторов*>
ELSE
    <*вторая группа операторов*>
END IF

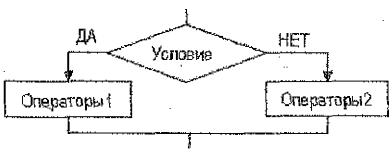


Рисунок 6

IF <*условие*> **THEN**
    <*первая группа операторов*>
ELSEIF <условие> **THEN**
    <*вторая группа операторов*>
ELSE
    <*третья группа операторов*>
END IF

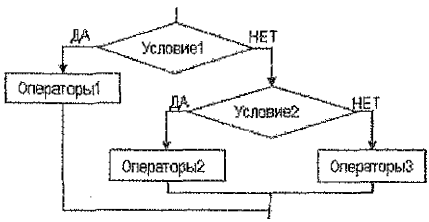


Рисунок 7

Достоинство многострочного **IF** состоит в том, что число операторов в группах не ограничено.

Оператор цикла For...Next

Цикл является одной из базовых алгоритмических конструкции и представляет собой последовательность операторов, которая выполняется многократно до тех пор, пока выполняется некоторое условие. Сама последовательность повторяющихся действий называется телом цикла. Циклы VBA могут быть организованы с использованием специального оператора ***For...Next***.

Синтаксис оператора:

For *i* = *Начальное значение* **To** *Конечное значение* [**Step** *Шаг*]

Тело цикла

Next *i*

Переменная *i* является параметром цикла или счётчиком выполнения команд тела цикла. При первом выполнении тела цикла значение параметра *i* равно *Начальное значение*. Каждый раз перед выполнением тела цикла текущее значение параметра *i* сравнивается с конечным значением. После каждого выполнения тела цикла параметр *i* увеличивается на заданный *Шаг*. Как только *i* превысит *Конечное значение*, выполнение цикла прекращается и осуществляется переход на следующий после цикла ***For*** оператор программы. Формат и блок-схема оператора цикла ***For...Next*** указаны на рисунке 8.

FOR *i* = *N1* **TO** *N2* **STEP** *h*

< *тело цикла* >

NEXT *i*

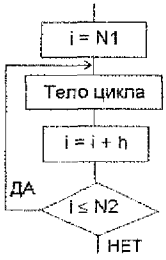


Рисунок 8

Массивы

Массив – это структурированный тип данных, конечная упорядоченная совокупность данных одного типа, доступ к которым осуществляется по индексу (порядковому номеру). Массивы бывают нескольких типов:

- ✓ **одномерные** – это пронумерованный список значений, где каждый элемент этого списка имеет свой уникальный номер, т.е. неповторяющийся индекс;
- ✓ **многомерные** – это более сложная переменная, которая содержит в себе набор списков (в VBA допускается описание массивов, имеющих до 60 размерностей, но, как правило, используются ***двухмерные*** массивы). В VBA двухмерный массив можно представить в виде строк и столбцов, т.е. в виде матрицы.

Массив должен быть объявлен до его использования в программе, и его объявление зависит от того, какой массив: статический или динамический.

Статический массив – это массив с переменным размером, который в свою очередь определяется количеством элементов:

Dim *имя_массива(размерность массива)* [**As** *тип данных массива*]

ПРИМЕРЫ:

- ✓ одномерный статический массив из 11-ти элементов: **Dim A(10)**
- ✓ одномерный статический строковый массив, включающий 10 элементов:

Dim Array_Str(1 To 10) As String

- ✓ двумерный статический массив целых чисел, включающий 6 * 8 = 48 элементов:

Dim Array_Mult(0 To 5, 0 To 7) As Integer

Динамический массив – это массив с переменным размером, т.е. количество элементов может изменяться во время выполнения программы:

Dim *имя_массива()* **[As** *тип данных массива***]**

Для задания размеров динамического массива во время выполнения программы используется следующая инструкция:

ReDim *имя_массива(размер)*

ПРИМЕР:

✓ объявление целочисленного динамического массива:

Dim MyAr() As Integer

✓ изменение массива на двумерный с количеством элементов 3 * 10 = 30:

ReDim MyAr(2, 9)

ЗАМЕЧАНИЯ:

а) по умолчанию нумерация элементов массива начинается с нуля. Для изменения индексации с нуля на единицу используется инструкция **Option Base N** (где N может принимать значения 0 и 1), которая должна располагаться в самом начале модуля VBA;

б) в случае отсутствия явного описания типа данных массива, он будет представлен как тип *Variant*.

Окно диалога InputBox

В режиме диалога данные можно ввести в программу с помощью функции ***InputBox***. Данная функция используется тогда, когда необходимо ввести только краткую информацию и имеет следующий синтаксис:

<Переменная> = InputBox(prompt [,title] [,default] [,xpos] [,ypos] [,helpfile], context])

Назначение параметров функции (рис. 9):

Prompt – определяет текст, отображаемый в диалоговом окне как приглашение («Введите значение переменной X»).

Title – отвечает за надпись заголовка («Ввод данных»). Если этот параметр не указан, то отображается название приложения.

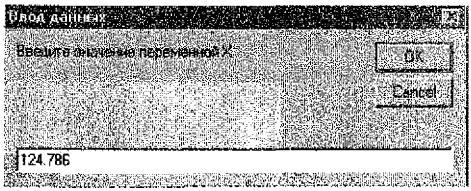


Рисунок 9

Default – определяет значение по умолчанию, отображаемое в строке ввода «124.786».

XPos и ***YPos*** используются совместно и указывают координаты верхнего левого угла окна. По умолчанию окно отображается посередине экрана.

Функция ***InputBox*** имеет еще два необязательных параметра ***HelpFile*** и ***Context***, которые позволяют открывать определенные файлы справочной системы.

При нажатии кнопки **OK**, функция ***InputBox*** возвращает строку, введенную пользователем. При нажатии кнопки **Cancel** возвращается пустая строка.

Пользовательские формы

Форма – это элемент проекта VBA, с помощью которого можно создавать диалоговые окна для взаимодействия с пользователем. Пользовательские окна обеспечивают уникальный интерфейс, наилучшим образом приспособленный для решения конкретных задач, стоящих перед пользователем. С точки зрения VBA форма представляет собой объект ***UserForm***. Как и любой объект, она имеет свои свойства, методы и события.

В VBA новая форма добавляется выбором команды из ***п. м. Insert → UserForm*** (рис. 10).

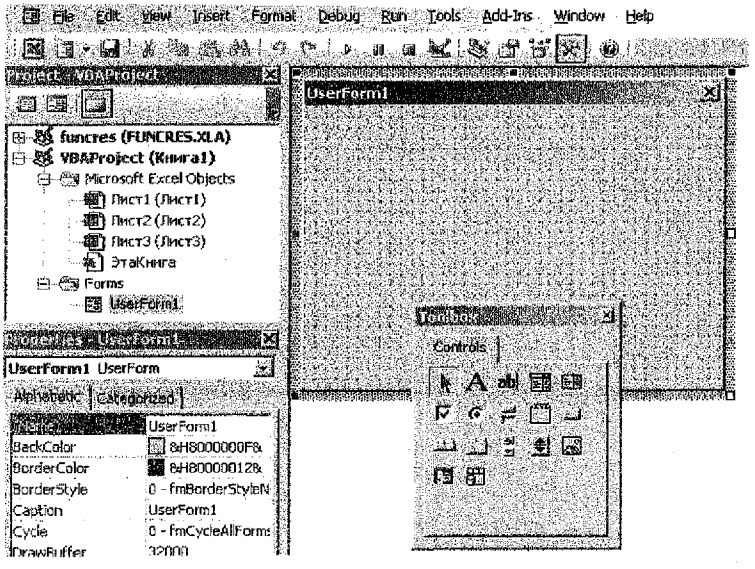


Рисунок 10

Форма состоит из двух частей:

- ✓ модуль формы – это модуль, в котором реализуется поведение формы (открывается командой из ***п. м. View → Code***);
- ✓ собственно форма – это диалоговое окно, в котором можно размещать элементы управления (ЭУ).

ЗАМЕЧАНИЕ: в проекте может быть как одна, так и несколько форм, размеры которых можно изменять при помощи маркеров изменения размеров.

Наиболее часто используемыми **свойствами** формы являются:

- ✓ ***Name*** – имя пользовательской формы;
- ✓ ***Caption*** – текст, отображаемый в строке заголовка формы.

Методы формы:

- ✓ ***Show*** – отображает форму на экране;
- ✓ ***Hide*** – скрывает форму;
- ✓ ***Move*** – изменяет местоположение и размер формы.

События формы:

- ✓ ***Initilize*** – происходит во время конфигурирования формы, но до её загрузки;
- ✓ ***Activate, Deactivate*** – происходят при активации и деактивации формы.

При работе с формами особое место занимают **операторы**, которые управляют процессами отображения окна на экране и закрытием окна:

- ✓ ***Load*** – загружает окно;
- ✓ ***Unload*** – выгружает форму с экрана и из памяти;
- ✓ ***End*** – завершает выполнение кода.

В VBA имеется широкий набор встроенных элементов управления, которые конструируются с помощью панели элементов управления ***Toolbox,*** появляющейся автоматически при создании новой формы. Отобразить данную панель можно также выбором соответствующей команды из ***п. м. View***, либо нажатием кнопки панели инструментов **Standart**. Все кнопки панели ***Toolbox***, за исключением кнопки **Выбор объектов**, служат для создания элементов управления, таких как кнопка, надпись, текстовое поле и др. (таблица. 4).

Таблица 4

Элемент управления		***Назначение***
	CommandButton (кнопка)	Когда пользователь щелкает по кнопке, выполняется VBA-процедура, закрепленная за данным ЭУ
A	Label (надпись)	Позволяет создавать заголовки ЭУ, которые не имеют собственных встроенных заголовков
ab|	TextBox (текстовое поле)	Окно редактируемого текста свободной формы для ввода данных (может быть одно- или многострочным)
	OptionButton (переключатель)	Используется для выбора между положениями «включено/выключено» или «истина/ложь» (как правило, объединяются вместе при помощи рамки для создания группы переключателей)
	CheckBox (флажок)	Используется для выбора вариантов, которые не являются взаимоисключающими
	ListBox (список)	Отображает список значений, из которых пользователь может сделать выбор
	ComboBox (поле со списком)	Объединяет окно редактирования и окно списка
	ScrollBar (полоса прокрутки)	Позволяет выбирать линейное значение аналогично тому, как это можно сделать при помощи счетчика
	SpinButton (счетчик)	Предназначен для ввода последовательной величины (числа, даты и др.), заведомо находящейся в определенном интервале значений
	Frame (рамка, группа)	Визуально и логически объединяет некоторые элементы управления
	Image (рисунок)	Позволяет вывести на форме графическое изображение (в любом из следующих форматов: *.bmp, *.cur, *.gif, *.ico, *.jpg, *.wmf)
	TabStrip (набор вкладок)	Используется для создания диалоговых вкладок, отображающих одни и те же данные в различных категориях
	MultiPage (набор страниц)	Используется для создания диалоговых окон с вкладками
	ToggleButton (выключатель)	Выводит установки в виде кнопки, находящейся в «нажатом» или «отжатом» состоянии

Элементы управления являются объектами, поэтому, как любые объекты, они обладают свойствами, методами и событиями.

Обращение к элементам управления осуществляется в основном через их свойства и с помощью процедур обработки событий, написанных для каждого объекта.

Одним из основных, наиболее часто употребляемых общих событий элементов управления является событие ***Click***, которое происходит, когда пользователь выбирает элемент управления с помощью одинарного щелчка кнопкой мыши.

3. РЕАЛИЗАЦИЯ В ТАБЛИЧНОМ ПРОЦЕССОРЕ MS EXCEL

3.1. Используемые инструменты

Инструмент ***«Подбор параметра»*** является частью блока задач анализа «что - если», когда желаемый результат одиночной формулы известен, но неизвестны значения, которые требуется ввести для получения этого результата.

Инструмент ***«Поиск решения»*** – надстройка, входящая в комплект поставки Excel, является частью блока задач анализа «что - если». Процедура поиска решения позволяет найти оптимальное значение формулы, содержащейся в ячейке, которая называется *целевой*. Эта процедура работает с группой ячеек, прямо или косвенно связанных с формулой в целевой ячейке. Чтобы получить по формуле, содержащейся в целевой ячейке, заданный результат, процедура изменяет значения во влияющих ячейках. Чтобы сузить множество значений, используемых в модели, применяются *ограничения*. Эти ограничения могут ссылаться на другие влияющие ячейки. Для подключения данной надстройки вызывается команда ***п. м. Сервис → Надстройки*** и в появившемся диалоговом окне устанавливается флажок ☑ Поиск решения (рис. 11).

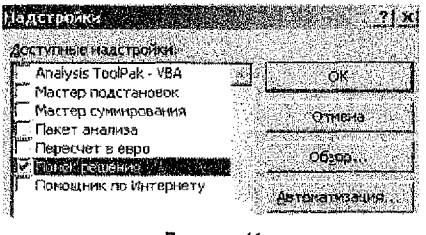


Рисунок 11

3.2. Порядок выполнения

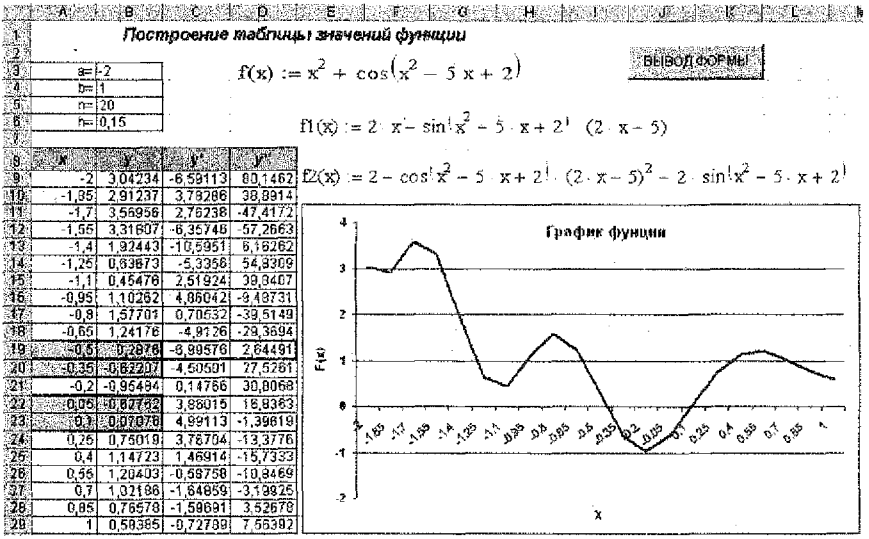
3.2.1. Рабочий лист «Функция»

На данном листе табулируем исходную функцию, первую и вторую производные (рис. 12).

Чтобы создать столбец аргумента x, в верхние две ячейки столбца заносим формулы для $x_0 = a$ и $x_1 = a + h$, используя вместо a, x_0 и h ссылки на ячейки, содержащие соответствующие значения: A9 = B3, A10 = A9 + $B$6. Затем ячейку A10 копируем с помощью маркера заполнения до конца отрезка b.

Чтобы создать столбцы значений функции $f(x)$, её производных $f'(x)$ и $f''(x)$, используем пользовательские функции, порядок создания которых описан в разделе 2.2.2, а программные коды приведены в разделе 4.2.4. Для этого заносим формулы в ячейки B9 = fun(A9), C9 = f_1(A9) и D9 = f_2(A9).

По построенной таблице значений функции $f(x)$ находим промежутки смены знака функции с «–» на «+» или наоборот с «+» на «–»: это *диапазоны B19:B20 и B22:B23, для которых выполняем цветную заливку.* Т.о. получены промежутки, на которых в последующем будут уточняться корни.



Построение таблицы значений функции

a= -2
b= 1
n= 20
h= 0,15

$f(x) := x^2 + \cos(x^2 - 5x + 2)$

ВЫВОД ФОРМЫ

$f1(x) := 2 \cdot x - \sin(x^2 - 5 \cdot x + 2) \cdot (2 \cdot x - 5)$

$f2(x) := 2 - \cos(x^2 - 5 \cdot x + 2) \cdot (2 \cdot x - 5)^2 - 2 \cdot \sin(x^2 - 5 \cdot x + 2)$

x	y	y'	y''
-2	3,04234	-6,59113	80,1462
-1,85	2,91237	3,78286	38,8914
-1,7	3,56956	2,76238	-47,4172
-1,55	3,31607	-6,35748	-57,2663
-1,4	1,92443	-10,5951	6,16262
-1,25	0,63673	-5,3358	54,8309
-1,1	0,45476	2,51924	39,3407
-0,95	1,10262	4,86042	-9,48731
-0,8	1,57701	0,70532	-39,5149
-0,65	1,24176	-4,9126	-29,3694
-0,5	0,2878	-6,89576	2,64491
-0,35	-0,62207	-4,50501	27,5261
-0,2	-0,96484	0,14766	30,8068
-0,05	-0,67762	3,86015	16,8363
0,1	0,07078	4,99113	-1,39619
0,25	0,75019	3,76704	-13,3776
0,4	1,14723	1,46914	-15,7333
0,55	1,20403	-0,59758	-10,9469
0,7	1,02186	-1,64859	-3,19925
0,85	0,76578	-1,59691	3,52678
1	0,59385	-0,72788	7,55392

Рисунок 12

Строим график функции $f(x)$, для чего вызываем команду ***п. м. Вставка → Диаграмма*** и следуем шагам Мастера диаграмм:

шаг 1 – на вкладке *Тип диаграммы* выбираем *График*;

шаг 2 – на вкладке *Диапазон данных* задаем Диапазон: =Функция!$B$9:$B$29,

на вкладке *Ряд* задаём *Подписи по оси X:* =Функция!$A$9:$A$29

шаг 3 – на вкладке *Заголовки* задаем название диаграммы и названия осей;

шаг 4 – помещаем диаграмму на имеющемся листе.

Далее на данном листе размещаем кнопку **ВЫВОД ФОРМЫ** для вызова пользовательской формы VBA-проекта.

В Excel 2003 отображаем панель инструментов «**Формы**» из ***п. м. Вид → Панели инструментов → Формы***, на которой выбираем элемент управления «Кнопка» и рисуем на рабочем листе с позиции крестика прямоугольную область, после чего в появившемся диалоговом окне «Назначить макрос объекту» указываем имя макроса, который будет запускаться нажатием на созданную кнопку (рис. 13).

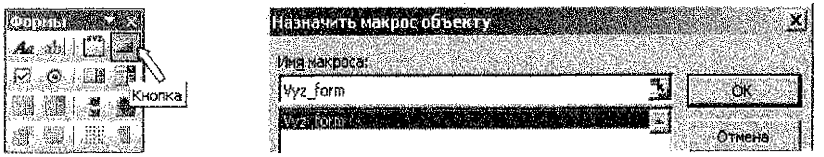


Рисунок 13

Для создания кнопки в более поздних версиях электронных таблиц сначала отображаем вкладку «**Разработчик**» на ленте:

- ✓ в Excel 2007 – с помощью команды ***Параметры Excel*** → ***Основные*** → в группе переключателей ***Основные параметры работы с Excel*** включаем флажок ☑ ***Показывать вкладку "Разработчик" на ленте.***
- ✓ в Excel 2010 – с помощью команды ***Файл*** → ***Параметры*** → ***Настройка ленты*** → в структуре ***Основные вкладки*** включаем флажок ☑ ***Разработчик.***

Далее выбираем команду ***Разработчик*** → ***Элементы управления*** → ***Вставить*** (рис. 14). В раскрывшейся палитре элементов управления щелкаем на элементе Кнопка. Перетаскиваем кнопку на рабочий лист, после чего Excel отобразит окно «Назначить макрос объекту», в котором выбираем из списка нужный макрос и OK.

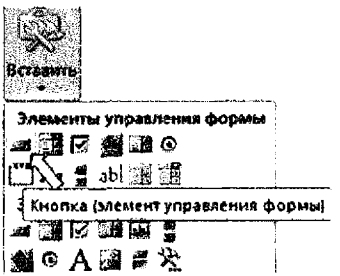


Рисунок 14

3.2.2. Рабочий лист «Корни»

На данном листе методом последовательного табулирования уточняем крайний левый корень (рис. 15) на отрезке отделения **[a*; b*] = [-0,5; -0,35]**, для чего строим таблицы по алгоритму, описанному в разделе 2.1.2. (рис. 16).

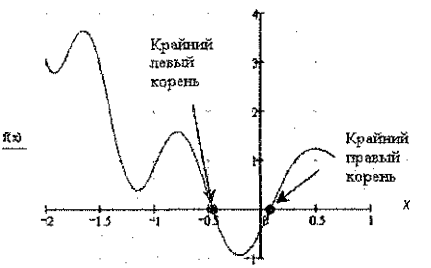


Рисунок 15

На пятой итерации достигнута указанная точность $\varepsilon = 0{,}00001$, которую проверяем формулой в ячейке

H30 = ЕСЛИ(ABS(I28 – I27)<=0,00001;"Точность достигнута";"Нет точности").

Значение точности вычисляем по формуле (2) в ячейке J30 = ABS(I28 – I27). Корень находим по формуле (3), т.е. J31 = I27 + (I28 – I27)/2.

	A	B	C	D	E	F	G	H	I	J
1	*Метод последовательного табулирования*									
2	*a=*	*-2*	*a=*	*-0,5*	*a=*	*-0,4625*	*a=*	*-0,458375*	*a=*	*-0,458266*
3	*b=*	*1*	*b=*	*-0,35*	*b=*	*-0,455*	*b=*	*-0,458000*	*b=*	*-0,458188*
4	*n=*	*20*	*n=*	*20*	*n=*	*20*	*n=*	*20*	*n=*	*20*
5	*h=*	*0,15*	*h=*	*0,0075*	*h=*	*0,000375*	*h=*	*0,00001875*	*h=*	*0,00000094*
6										
7	*X*	*Y(X)*	*X*	*Y(X)*	*X*	*Y(X)*	*X*	*Y(X)*	*X*	*Y(X)*
8	-2	3,0423405	-0,5	0,2876022	-0,4625	0,020994	-0,4583750	0,0012503	-0,4582063	0,0001194
9	-1,85	2,9123725	-0,4925	0,2362236	-0,4621250	0,026464	-0,4583563	0,0011247	-0,4582053	0,0001131
10	-1,7	3,5695631	-0,485	0,1830647	-0,4617500	0,023936	-0,4583375	0,0009990	-0,4582044	0,0001068
11	-1,55	3,3180711	-0,4775	0,131275	-0,4613750	0,021409	-0,4583188	0,0008733	-0,4582034	0,0001005
12	-1,4	1,9244332	-0,47	0,0798826	-0,4610000	0,018884	-0,4583000	0,0007477	-0,4582025	0,0000942
13	-1,25	0,6967273	-0,4625	0,0289938	-0,4606250	0,016360	-0,4582813	0,0006220	-0,4582016	0,0000880
14	-1,1	0,4547612	-0,455	-0,021307	-0,4602500	0,013838	-0,4582625	0,0004963	-0,4582006	0,0000817
15	-0,95	1,1026212	-0,4475	-0,070937	-0,4598750	0,011317	-0,4582438	0,0003707	-0,4581997	0,0000754
16	-0,8	1,5770142	-0,44	-0,119817	-0,4595000	0,008798	-0,4582250	0,0002450	-0,4581988	0,0000691
17	-0,65	1,2417552	-0,4325	-0,167868	-0,4591250	0,006281	-0,4582063	0,0001194	-0,4581978	0,0000628
18	-0,5	0,2876022	-0,425	-0,216014	-0,4587500	0,003765	-0,4581875	0,0000063	-0,4581969	0,0000565
19	-0,35	-0,622069	-0,4175	-0,261184	-0,4583750	0,001250	-0,4581688	-0,0001319	-0,4581959	0,0000503
20	-0,2	-0,954844	-0,41	-0,306306	-0,4580000	-0,001262	-0,4581500	-0,0002575	-0,4581950	0,0000440
21	-0,05	-0,627617	-0,4025	-0,350313	-0,4576250	-0,003774	-0,4581313	-0,0003832	-0,4581941	0,0000377
22	0,1	0,0707589	-0,395	-0,393139	-0,4572500	-0,006283	-0,4581125	-0,0005088	-0,4581931	0,0000314
23	0,25	0,7501656	-0,3875	-0,434724	-0,4568750	-0,008791	-0,4580938	-0,0006344	-0,4581922	0,0000251
24	0,4	1,1472273	-0,38	-0,475008	-0,4565000	-0,011298	-0,4580750	-0,0007601	-0,4581913	0,0000189
25	0,55	1,2040317	-0,3725	-0,513935	-0,4561250	-0,013803	-0,4580563	-0,0008857	-0,4581903	0,0000126
26	0,7	1,0218607	-0,365	-0,551455	-0,4557500	-0,016305	-0,4580375	-0,0010113	-0,4581894	0,0000063
27	0,85	0,7657828	-0,3575	-0,587514	-0,4553750	-0,018807	-0,4580188	-0,0011369	-0,4581884	0,0000000
28	1	0,5838532	-0,35	-0,622069	-0,4550000	-0,021307	-0,4580000	-0,0012625	-0,4581875	-0,0000063
29										
30							*Точность достигнута*		*e =*	0,0000009
31							*Корень функции*		*x1 =*	-0,4581880

Рисунок 16

Три первые итерации метода последовательного табулирования представлены в формульном виде на рисунке 17.

	A	B	C	D	E	F
1	*Метод последовательного табулирования*					
2	*a=*	*-2*	*a=*	*=A18*	*a=*	*=C13*
3	*b=*	*1*	*b=*	*=A19*	*b=*	*=C14*
4	*n=*	*20*	*n=*	*20*	*n=*	*20*
5	*h=*	*=ABS(B3-B2)/B4*	*h=*	*=ABS(D3-D2)/D4*	*h=*	*=ABS(F3-F2)/F4*
6						
7	*X*	*Y(X)*	*X*	*Y(X)*	*X*	*Y(X)*
8	=B2	=A8^2+COS(A8^2-5*A8+2)	=D2	=C8^2+COS(C8^2-5*C8+2)	=F2	=E8^2+COS(E8^2-5*E8+2)
9	=A8+$B$5	=A9^2+COS(A9^2-5*A9+2)	=C8+$D$5	=C9^2+COS(C9^2-5*C9+2)	=E8+$F$5	=E9^2+COS(E9^2-5*E9+2)
10	=A9+$B$5	=A10^2+COS(A10^2-5*A10+2)	=C9+$D$5	=C10^2+COS(C10^2-5*C10+2)	=E9+$F$5	=E10^2+COS(E10^2-5*E10+2)
11	=A10+$B$5	=A11^2+COS(A11^2-5*A11+2)	=C10+$D$5	=C11^2+COS(C11^2-5*C11+2)	=E10+$F$5	=E11^2+COS(E11^2-5*E11+2)
12	=A11+$B$5	=A12^2+COS(A12^2-5*A12+2)	=C11+$D$5	=C12^2+COS(C12^2-5*C12+2)	=E11+$F$5	=E12^2+COS(E12^2-5*E12+2)
13	=A12+$B$5	=A13^2+COS(A13^2-5*A13+2)	=C12+$D$5	=C13^2+COS(C13^2-5*C13+2)	=E12+$F$5	=E13^2+COS(E13^2-5*E13+2)
14	=A13+$B$5	=A14^2+COS(A14^2-5*A14+2)	=C13+$D$5	=C14^2+COS(C14^2-5*C14+2)	=E13+$F$5	=E14^2+COS(E14^2-5*E14+2)
15	=A14+$B$5	=A15^2+COS(A15^2-5*A15+2)	=C14+$D$5	=C15^2+COS(C15^2-5*C15+2)	=E14+$F$5	=E15^2+COS(E15^2-5*E15+2)
16	=A15+$B$5	=A16^2+COS(A16^2-5*A16+2)	=C15+$D$5	=C16^2+COS(C16^2-5*C16+2)	=E15+$F$5	=E16^2+COS(E16^2-5*E16+2)
17	=A16+$B$5	=A17^2+COS(A17^2-5*A17+2)	=C16+$D$5	=C17^2+COS(C17^2-5*C17+2)	=E16+$F$5	=E17^2+COS(E17^2-5*E17+2)
18	=A17+$B$5	=A18^2+COS(A18^2-5*A18+2)	=C17+$D$5	=C18^2+COS(C18^2-5*C18+2)	=E17+$F$5	=E18^2+COS(E18^2-5*E18+2)
19	=A18+$B$5	=A19^2+COS(A19^2-5*A19+2)	=C18+$D$5	=C19^2+COS(C19^2-5*C19+2)	=E18+$F$5	=E19^2+COS(E19^2-5*E19+2)
20	=A19+$B$5	=A20^2+COS(A20^2-5*A20+2)	=C19+$D$5	=C20^2+COS(C20^2-5*C20+2)	=E19+$F$5	=E20^2+COS(E20^2-5*E20+2)

Рисунок 17

Далее для уточнения крайнего левого корня функции на отрезке отделения **[a*; b*] = [-0,5; -0,35]** методом Ньютона создаём расчетную таблицу по алгоритму, описанному в разделе 2.1.2. (рис . 18).

В качестве начального приближения x_0 задаем границу a^* отрезка отделения, так как выполняется неравенство (6), т.е. в ячейку ***B35*** вводим значение, равное ***-0,5***.

Расчетная таблица в формульном виде представлена на рисунке 19.

Для дальнейшего уточнения корней функции с помощью встроенных средств Excel (*Подбор параметра* и *Поиск решения*) задаём необходимую точность вычислений, для чего вызываем команду ***п. м. Сервис → Параметры → вкладка Вычисления →*** изменяем параметр ***относительная погрешность*** на 0,00001.

	A	B	C	D	E	F	G
33				Метод Ньютона			
34	№ п/п	Xn	F(Xn)	F'(Xn)	F(Xn)/F'(Xn)	Xn+1	ε
35	0	-0,5	0,2976022	-6,996757	-0,041111	-0,458889	0,041111
36	1	-0,458889	0,0046976	-6,708779	-0,000700	-0,458189	0,000700
37	2	-0,458189	0,0000027	-6,700912	0,000000	-0,458188	0,000000
38							
39	Подбор параметра			Поиск Решения			
40	x1	y1		x1	y1		
41	-0,458186	0,000000		-0,458186	-0,000003		
42							
43	x2	y2		x2	y2		
44	0,085841	0,000000		0,085841	0,000000		

Рисунок 18

	A	B	C	D	E	F	G
33				Метод Ньютона			
34	№ п/п	Xn	F(Xn)	F'(Xn)	F(Xn)/F'(Xn)	Xn+1	ε
35	0	-0,5	=B35^2+COS(B35^2-5*B35+2)	=2*B35-SIN(B35^2-5*B35+2)*(2*B35-5)	=C35/D35	=B35-E35	=ABS(F35-B35)
36	1	=F35	=B36^2+COS(B36^2-5*B36+2)	=2*B36-SIN(B36^2-5*B36+2)*(2*B36-5)	=C36/D36	=B36-E36	=ABS(F36-B36)
37	2	=F36	=B37^2+COS(B37^2-5*B37+2)	=2*B37-SIN(B37^2-5*B37+2)*(2*B37-5)	=C37/D37	=B37-E37	=ABS(F37-B37)

Рисунок 19

Для уточнения крайнего левого корня средством ***Подбор параметра***:

1) на рабочем листе отводим ячейку **A41** для приближенного значения искомого корня и указываем начальное приближение для поиска, т.е. вводим с клавиатуры значение, равное ***-0,5***;

2) в ячейку **B41** записываем формулу исследуемой функции, ссылаясь в ней на ячейку **A41** в качестве её аргумента: B41 = A41^2 + COS(A41^2 – 5*A41 + 2);

3) выполняем команду ***п. м. Сервис → Подбор Параметра*** и в открывшемся диалоговом окне (рис. 20) заполняем следующее:

- ✓ в поле *Установить в ячейке* мышью ссылаемся на ячейку **B41**;
- ✓ в поле *Значение* вводим с клавиатуры значение ***0***;
- ✓ в поле *Изменяя значение ячейки* мышью ссылаемся на ячейку **A41**;
- ✓ нажимаем кнопку **ОК**.

Решение считается успешным, если в результате подбора параметра в одноименном диалоговом окне будет выдано сообщение «Решение найдено» (рис. 20).

В ячейке **A41** получаем значение корня функции.

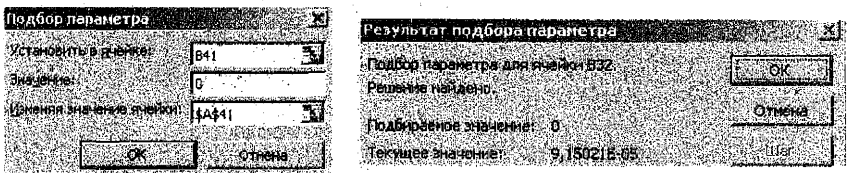


Рисунок 20

Аналогично уточняем значение второго (крайнего правого) корня функции на отрезке отделения **[a*; b*] = [-0,05; 0,1]**, указывая перед выполнением команды *Подбор параметра* начальное приближение для поиска, равное, например, ***0,1***.

Для уточнения крайнего левого корня средством ***Поиск Решения***:

1) на рабочем листе отводим ячейку **D41** для искомого корня функции;

2) в ячейку **E41** записываем формулу исследуемой функции, ссылаясь в ней на ячейку **D41** в качестве её аргумента: E41 = D41^2 + COS(D41^2 – 5*D41 + 2);

3) вызываем команду из ***п. м. Сервис → Поиск решения***, после чего на экране появится одноименное диалоговое окно (рис. 21), в котором:

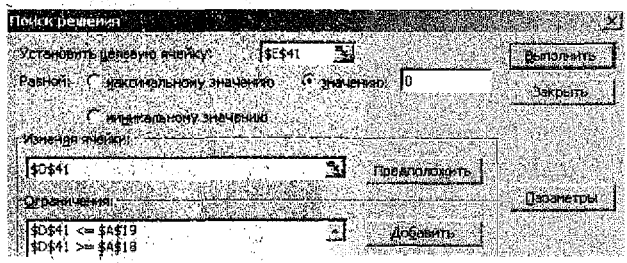


Рисунок 21

a) назначаем ячейку с целевой функцией, для чего курсор помещаем в поле ***Установить целевую ячейку*** и мышью ссылаемся на ячейку ***$E$41;***

b) выбираем направление целевой функции к значению, равному нулю;

c) назначаем ячейку для искомого результата (значения корня), для чего курсор помещаем в поле *Изменяя ячейки* и мышью выделяем ячейку ***$D$41***;

d) вводим ограничения, для чего нажимаем экранную кнопку **Добавить**. В появившемся диалоговом окне «Добавление ограничения» (рис. 22) задаем ограничение на поиск корня (отрезок отделения, выделенный цветом в таблице значений аргумента), для чего:

- ✓ в поле *Ссылка на ячейку:* мышью указываем на ячейку ***$D$41***;
- ✓ из раскрывающегося списка знаков выбираем **>=**;
- ✓ в поле *Ограничение:* мышью указываем ячейку ***$A$18*** со значением левой границы отрезка из первой таблицы метода последовательного табулирования;

e) нажимаем экранную кнопку **Добавить** и аналогично указываем ограничение по правой границе отрезка;

f) нажимаем экранную кнопку **Выполнить**.

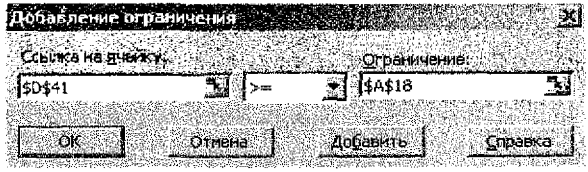


Рисунок 22

Аналогично уточняем значение второго (крайнего правого) корня функции на отрезке отделения **[a*; b*] = [-0,05; 0,1]**.

3.2.3. Рабочий лист «Экстремумы»

На данный рабочий лист из таблицы, построенной на листе «Табулирование», копируем блоки ячеек со значениями функций и производных, в которых выполняются условия существования экстремума:

- ✓ если $f(x) > f(x-h)$ и $f(x) > f(x+h)$, то на отрезке $[x-h; x+h]$ существует локальный максимум;
- ✓ если $f(x) < f(x-h)$ и $f(x) < f(x+h)$, то на отрезке $[x-h; x+h]$ существует локальный минимум.

Поскольку ячейки помещаются на другой лист вместе с формулами, значения самих аргументов переносим на лист «Экстремумы» с помощью копирования только значений ячеек (***п. м. Правка → Специальная вставка →⊙ Значения***).

Рассмотрим порядок нахождения крайних левых минимума и максимума функции (рис. 23).

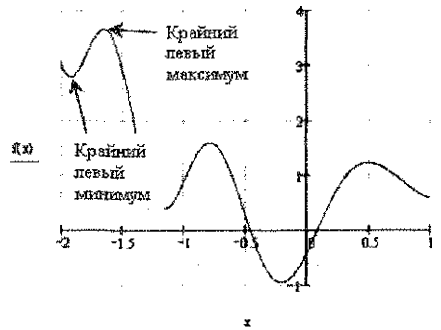


Рисунок 23

Для уточнения, например, крайнего левого минимума копируем с листа «табулирование» диапазон **значений** аргумента функции **A9:A11** с помощью специальной вставки

и диапазон **B9:D11** значений функции и производных обычным копированием на рабочий лист «Экстремумы» (рис. 24).

Создаем дополнительную строку, в которой:

- ✓ в ячейку **A8** вводим начальное значение, принадлежащее отрезку **[-2; -1,7]**;
- ✓ в ячейке **B8** задаём формулу для расчёта функции;
- ✓ в ячейках **C8** и **D8** задаём формулы для расчёта первой и второй производных функции соответственно.

Используем инструмент **Поиск решения** (из *п. м. Сервис*) и в диалоговом окне указываем следующее:

- ✓ в поле ***Установить целевую ячейку*** мышью ссылаемся на ячейку ***$B$8;***
- ✓ выбираем направление целевой функции к минимальному значению;
- ✓ в поле *Изменяя ячейки* мышью выделяем ячейку ***$A$8***;
- ✓ с помощью кнопки **Добавить** вводим условия принадлежности подбираемой экстремальной точки заданному отрезку **[-2; -1,7]**;
- ✓ нажимаем экранную кнопку **Выполнить**.

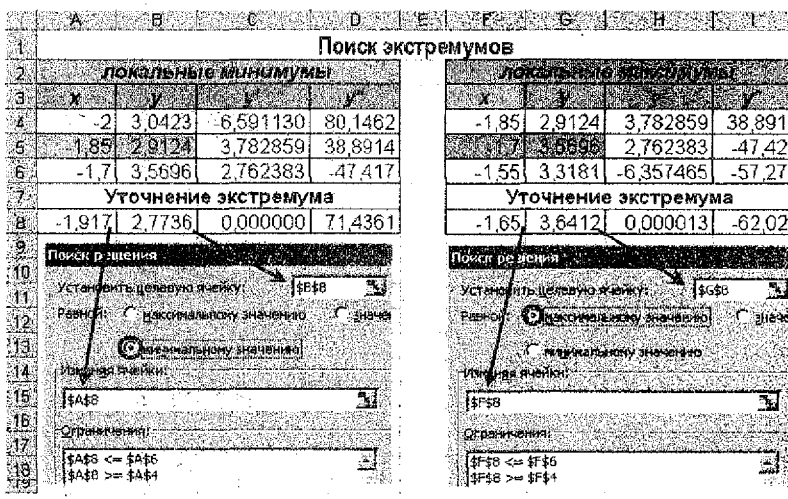


Рисунок 24

Для уточнения, например, крайнего левого максимума копируем с листа «табулирование» диапазон **A10:A12 значений** аргумента функции с помощью специальной вставки и диапазон **B10:D12** значений функции и производных обычным копированием на рабочий лист «Экстремумы» (рис. 24).

Создаем дополнительную строку, в которой:

- ✓ в ячейку **F8** вводим начальное значение, принадлежащее отрезку **[-1,85; -1,55]**;
- ✓ в ячейке **G8** задаём формулу для расчёта функции;
- ✓ в ячейках **H8** и **I8** задаём формулы для расчёта первой и второй производных функции соответственно.

Используем инструмент **Поиск решения**, в диалоговом окне которого указываем следующее:

- ✓ в поле ***Установить целевую ячейку*** мышью ссылаемся на ячейку ***$G$8;***
- ✓ выбираем направление целевой функции к минимальному значению;
- ✓ в поле *Изменяя ячейки* мышью выделяем ячейку ***$F$8***;
- ✓ с помощью кнопки **Добавить** вводим условия принадлежности подбираемой экстремальной точки заданному отрезку **[-1,85; -1,55];**
- ✓ нажимаем экранную кнопку **Выполнить**.

По аналогии находим остальные локальные экстремумы функции.

Далее, среди локальных экстремумов находим глобальный минимум и максимум функции по следующим формулам:

B23 = МИН(B8;B14;B20) и G23 = МАКС(G8;G14;G20) (рис. 25).

	A	B	C	D	E	F	G	H	I
1	Поиск экстремумов								
2	*локальные минимумы*					*локальные максимумы*			
3	*x*	*y*	*y'*	*y''*		*x*	*y*	*y'*	*y''*
4	-2	3,0423	-6,59112985	80,146		-1,85	2,9124	3,782859338	38,891
5	-1,85	2,9124	3,782859338	38,891		-1,7	3,5696	2,762383242	-47,417
6	-1,7	3,5696	2,762383242	-47,417		-1,55	3,3181	-6,357464849	-57,266
7	Уточнение экстремума					Уточнение экстремума			
8	-1,9169	2,7736	-0,000000070	71,436		-1,6503	3,6412	0,000012563	-62,015
9									
10	-1,25	0,6367	-5,33560393	54,831		-0,95	1,1026	4,860420914	-9,4873
11	-1,1	0,4548	2,51923813	39,841		-0,8	1,577	0,705322793	-39,515
12	-0,95	1,1026	4,860420914	-9,4873		-0,65	1,2418	-4,912602928	-29,369
13	Уточнение экстремума					Уточнение экстремума			
14	-1,1542	0,3834	-0,000000052	52,047		-0,7824	1,5833	0,000000043	-40,33
15									
16	-0,35	-0,6221	-4,505009303	27,526		0,4	1,1472	1,469136468	-15,733
17	-0,2	-0,9548	0,147657126	30,807		0,55	1,204	-0,587580932	-10,847
18	-0,05	-0,6276	3,860151698	16,836		0,7	1,0219	-1,648594641	-3,1993
19	Уточнение экстремума					Уточнение экстремума			
20	-0,2048	-0,9552	0,000000001	31,028		0,5008	1,2189	0,000000024	-12,976
21									
22		*глобальный минимум*					*глобальный максимум*		
23		-0,955196902					3,641160376		

Рисунок 25

4. РЕАЛИЗАЦИЯ В СРЕДЕ ПРОГРАММИРОВАНИЯ VBA

4.1. Используемые инструменты

Компонента ***многостраничной формы (MultiPage)*** предоставляет набор из нескольких страниц, каждая из которых содержит собственный заголовок и собственный набор элементов управления. Для перехода между страницами достаточно кликнуть по заданному корешку.

Добавление данной компоненты осуществляется командой из *п. м.* ***Tools → Additional Control → ☒ Microsoft Forms 2.0 MultiPage***.

После того, как выбран компонент *MultiPage*, из панели *ToolBox* (рис. 26) можно помещать на поверхность формы **Набор страниц** (по умолчанию отображается только две вкладки: *Page1* и *Page2*).

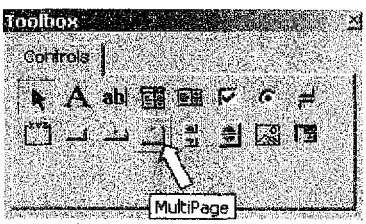


Рисунок 26

Для управления вкладками используются следующие команды из контекстного меню, вызываемого нажатием правой кнопки мыши по одной из имеющихся вкладок:

NewPage – добавление новой страницы к уже имеющимся (имена корешков страниц назначаются автоматически).

DeletePage – удаление выделенной страницы.

Rename – переименование выбранной страницы, (аналогичное действие выполняется в окне *Properties*: изменяется значение свойства *Caption*).

Move – перемещает выбранную страницу: после щёлчка мыши в открывшемся диалоговом окне указывается направление, в котором происходит перемещение.

Элемент управления ***«гибкая таблица» или «сетка» (MS FlexGrid)*** используется для вывода табличных данных на форму и является дополнительным, самостоятельным элементом управления, обладающим как общими для всех ЭУ свойствами и методами, так и присущими только ему. Добавление данной компоненты осуществляется из *п. м.* ***Tools → Additional Control → ☒ Microsoft Flex Grid Control 6.0*** (рис. 27).

Свойства элемента управления ***MS FlexGrid:***

TextMatrix – позволяет считывать или вносить текст в заданную ячейку.

GridsLines – контролирует отображение разделительных линий.

ScrollBars – контролирует отображение линеек прокрутки.

FixedCol – возвращает или задает общее количество стационарных (фиксированных) колонок в пределах сетки.

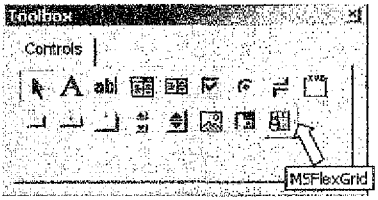


Рисунок 27

FixedRows – возвращает или задает общее количество фиксированных рядов в пределах сетки.

Фиксированные столбцы и строки являются постоянными при прокрутке других столбцов или строк в MSFlexGrid. Можно указать ноль или несколько фиксированных столбцов или строк. Кроме того, можно выбрать цвета, шрифт, линии сетки, и стиль текста фиксированных столбцов и строк.

Heigt – высота; **Width** – ширина; **Enabled** – доступность.

ScrolBar – линейка прокрутки, которая имеет 4 значения: 0 – выводится автоматически, 1 – горизонтальная, 2 – вертикальная, 3 – обе.

Cols, Rows – устанавливает число колонок и столбцов.

Col, Row – возвращает / устанавливает номер колонки и строки.

ColWidth, RowHeight – ширина и высота столбца.

Text, TextMatrix – возвращает или устанавливает текст, хранящийся в текущей ячейке.

ColAlignment – выравнивание текста в ячейках:

0 – выравнивание по левому краю;

1 – выравнивание по правому краю;

2 – центрирование текста.

4.2. Порядок выполнения

Приступая к работе в среде программирования VBA, понижаем уровень безопасности:

- ✓ в Excel 2003 – ***п. м. Сервис → Макрос → Безопасность → ⊙ Средняя***;
- ✓ в Excel 2007 – ***Меню → Параметры Excel → Центр управления безопасностью → Параметры центра управления безопасностью → Параметры макросов → ⊙ Включить все макросы (не рекомендуется, возможен запуск опасной программы).***
- ✓ в Excel 2010 – вкладка ***Разработчик*** → группа ***Код → Безопасность макросов → Параметры макросов → ⊙ Включить все макросы (не рекомендуется, возможен запуск опасной программы).***

Открываем *редактор VB* (***п. м. Сервис → Макрос → Редактор Visual Basic***) и добавляем стандартный модуль, в котором прописываем процедуры для определения заданной функции $f(x)$, а также её первой $f'(x)$ и второй $f''(x)$ производных:

```
' Описание исходной функции
Public Function fun(X)
fun = X ^ 2 + Cos(X ^ 2 – 5 * X + 2)
End Function

' Описание первой производной функции
Public Function f_1(X)
f_1 = 2 * X – Sin( X ^ 2 – 5 * X + 2) * (2 * X – 5)
End Function

' Описание второй производной функции
Public Function f_2(X)
f_2 = 2 – Cos(X ^ 2 – 5 * X + 2) * (2 * X – 5) ^ 2 – 2 * Sin( X ^ 2 – 5 * X + 2)
End Function
```

Далее добавляем новую форму командой ***п. м. Insert → UserForm*** и делаем её многостраничной с помощью элемента управления набор страниц ***(MultiPage)***, работа с которым описана выше в разделе «Используемые инструменты».

4.2.1. Вкладка «Паспорт программы»

На данной вкладке размещаем текст титульного листа курсовой работы (рис. 28).

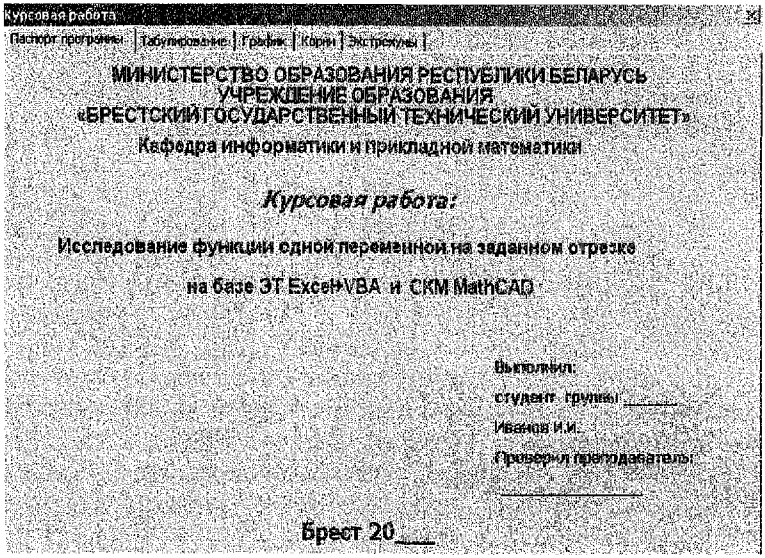


Рисунок 28

Для добавления и оформления текста используем элемент управления ***надпись – Label*** (рис. 29) с изменением следующих его свойств:

Caption – возвращает текст, отображаемый в надписи.

Font – настройка элементов форматирования текста: гарнитура, высота, начертание шрифта и пр.

Alignment – выравнивание текста (***0*** – выравнивание по левому краю, ***1*** – по правому краю, ***2*** – по центру).

AutoSize – автоматическое приведение ширины объекта в соответствие с длиной текста (если данное свойство = ***False*** и длина вводимого текста больше ширины надписи, то текст усекается, если свойство = ***True***, то размер объекта приводится в соответствие с длиной текста).

WordWrap – автоматический перенос длинного текста на другую строку.

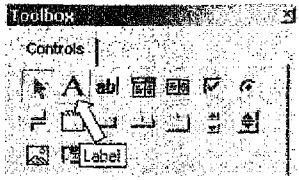


Рисунок 29

4.2.2. Вкладка «Табулирование»

На данной вкладке (рис. 30) размещаем следующие элементы управления:

- ✓ ***Сетка – MS FlexGrid*** – для вывода таблицы значений функции и её производных, установив следующие свойства: ***Cols*** = 5, ***FixedCols*** = 1, ***FixedRows*** = 1 (добавление и использование данного ЭУ описано в разделе 4.1).

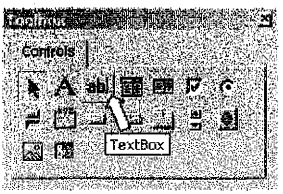
i	X	fun	f_1	f_2
1	-2	3,04234	-6,59113	80,14622
2	-1,85	2,91237	3,78286	38,89136
3	-1,7	3,56956	2,76238	-47,41721
4	-1,55	3,31807	-6,35746	-57,26631
5	-1,4	1,92443	-10,59506	6,16262
6	-1,25	0,63673	-5,3356	54,83088
7	-1,1	0,45476	2,51924	39,84068
8	-0,95	1,10262	4,86042	-9,48731
9	-0,8	1,57701	0,70532	-39,51492
10	-0,65	1,24176	-4,9126	-29,36938
11	-0,5	0,2876	-6,99576	2,64491
12	-0,35	-0,62207	-4,50501	27,52614
13	-0,2	-0,95484	0,14766	30,80681
14	-0,05	-0,62762	3,86015	16,83634
15	0,1	0,07076	4,99113	-1,39619
16	0,25	0,75019	3,76704	-13,37765
17	0,4	1,14723	1,46914	-15,73333
18	0,55	1,20403	-0,58758	-10,84667
19	0,7	1,02166	-1,64859	-3,19925
20	0,85	0,76578	-1,59691	3,52678
21	1	0,58385	-0,72789	

Рисунок 30

- ✓ ***Текстовые поля – TextBox*** – для ввода границ и точек разбиения исследуемого отрезка, а также для ввода количества знаков после запятой, необходимого при

отображении значений функций в «гибкой таблице» (рис. 31). Для данных ЭУ изменяем свойство **Font** (гарнитура, размер и начертание шрифта).

✓ ***Надпись*** – ***Label*** – для подписи текстовых полей, не имеющих собственных заголовков (рис. 29).



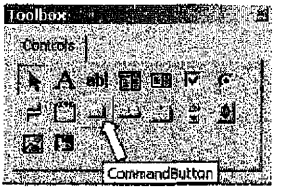


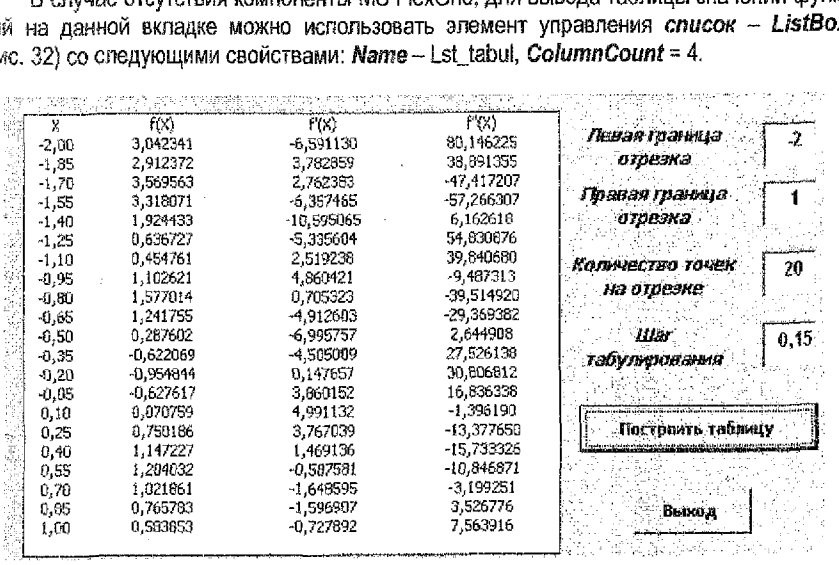
Рисунок 31

✓ ***Кнопки*** – ***CommandButton*** (рис. 31) – для вызова процедуры обработки события **cmdTabul_Click** *(табулирование функций с выводом результатов в сетку MS FlexGrid)* и процедуры обработки события **cmdexit1_Click** (*закрытие формы*), расположенных в ***модуле формы***. Первая кнопка имеет соответственно следующие свойства: ***Name*** – cmdTabul, ***Caption*** – «Построить таблицу», а вторая: ***Name*** – cmdexit1, ***Caption*** – «Выход».

```
Private Sub cmdTabul_Click()
a = Val(TextBox1.Text)          ' начало отрезка
b = Val(TextBox2.Text)          ' конец отрезка
n = Val(TextBox3.Text)          ' количество точек
E = Val(TextBox4.Text)          ' количество знаков после запятой
h = (b – a) / n : TextBox5.Text = h  ' шаг табулирования
' заполнение "шапки" (заголовков столбцов) сетки
FlexGrid1.TextMatrix(0, 0) = "i" :    FlexGrid1.TextMatrix(0, 1) = "X"
FlexGrid1.TextMatrix(0, 2) = "fun" :  FlexGrid1.TextMatrix(0, 3) = "f_1"
FlexGrid1.TextMatrix(0, 4) = "f_2"
i = 1
' цикл по заполнению ячеек сетки значениями функций
For x = a To b + h / 2 Step h
   f = fun(x) :   f1 = f_1(x) :   f2 = f_2(x)
FlexGrid1.TextMatrix(i, 0) = i
FlexGrid1.TextMatrix(i, 1) = Round(x, 2)
FlexGrid1.TextMatrix(i, 2) = Round(f, E)
FlexGrid1.TextMatrix(i, 3) = Round(f1, E)
FlexGrid1.TextMatrix(i, 4) = Round(f2, E)
i = i + 1
Next x
End Sub
```

```
Private Sub cmdexit1_Click()
End
End Sub
```

В случае отсутствия компоненты MS FlexGrid, для вывода таблицы значений функций на данной вкладке можно использовать элемент управления ***список*** – ***ListBox*** (рис. 32) со следующими свойствами: ***Name*** – Lst_tabul, ***ColumnCount*** = 4.



X	f(X)	f'(X)	f''(X)
-2,00	3,042341	-6,591130	80,146225
-1,85	2,912372	3,782859	38,891355
-1,70	3,569563	2,762383	-47,417207
-1,55	3,318071	-6,367465	-57,266307
-1,40	1,924433	-10,595065	6,162618
-1,25	0,636727	-5,335604	54,830676
-1,10	0,454761	2,519238	39,840680
-0,95	1,102621	4,860421	-9,487313
-0,80	1,577014	0,705323	-39,514920
-0,65	1,241755	-4,912603	-29,369382
-0,50	0,287602	-6,995757	2,644908
-0,35	-0,622069	-4,505009	27,526138
-0,20	-0,954844	0,147657	30,606812
-0,05	-0,627617	3,860152	16,836338
0,10	0,070759	4,991132	-1,396190
0,25	0,758186	3,767039	-13,377650
0,40	1,147227	1,469136	-15,733326
0,55	1,204032	-0,587581	-10,846871
0,70	1,021861	-1,648595	-3,199251
0,85	0,765783	-1,596807	3,526776
1,00	0,593653	-0,727892	7,563916

Рисунок 32

В данном случае, командной кнопке **Построить таблицу** будет соответствовать процедура обработки события **cmdTabul_Click** со следующим программным кодом VBA:

```
Private Sub cmdTabul_Click()
Dim val_x(), val_fun(), val_fun1(), val_fun2()
a = Val(txtA.Text)              ' начало отрезка
b = Val(txtB.Text)              ' конец отрезка
n = Val(txtN.Text)              ' количество точек
h = (b – a) / n
txtH.Text = h                   ' шаг табулирования
ReDim val_x(), val_fun(n), val_fun1(n), val_fun2(n)
' заполнение "шапки" (заголовков столбцов) списка
With Me.Lst_tabul
    .AddItem
    .List(0, 0) = "X"
    .List(0, 1) = "f(X)"
    .List(0, 2) = "f'(X)"
    .List(0, 3) = "f''(X)"
End With
i = 0
```

```
'цикл по заполнению столбцов списка значениями функций
For x = a To b + h / 2 Step h
   val_x(i) = Format(x, "0.00") :  val_fun(i) = Format(fun(x), "0.000000")
   val_fun1(i) = Format(f_1(x), "0.000000") :  val_fun2(i) = Format(f_2(x), "0.000000")
With Me.Lst_tabul
      .AddItem
      .List(i + 1, 0) = val_x(i)
      .List(i + 1, 1) = val_fun(i)
      .List(i + 1, 2) = val_fun1(i)
      .List(i + 1, 3) = val_fun2(i)
End With
i = i + 1
Next x
End Sub
```

4.2.3. Вкладка «График»

На данную вкладку помещаем график, построенный либо в ЭТ Excel, либо в СКМ MathCad (рис. 34), для чего сначала копируем изображение графика в любой графический редактор и сохраняем его в отдельном файле, а затем на вкладке формы размещаем элемент управления ***рисунок – Image*** (рис. 33), для которого изменяем свойство ***Picture*** в окне *Properties* с помощью кнопки … – выбираем имя сохранённого файла.

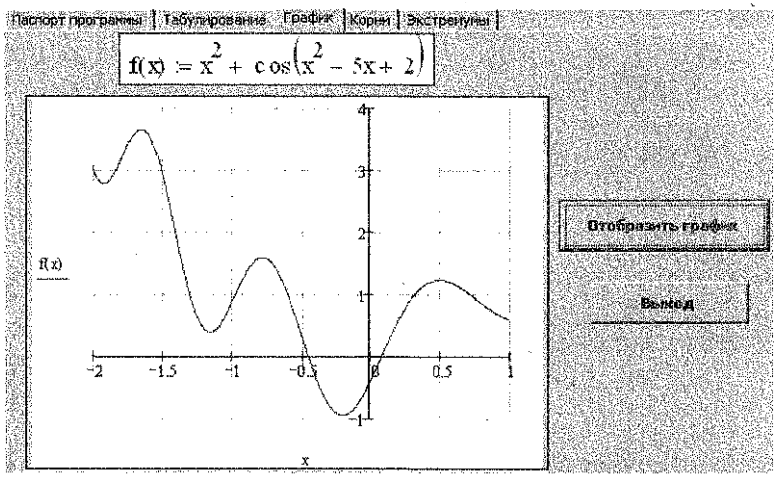


Рисунок 33

Для того, чтобы рисунок полностью вписался в рамку ЭУ ***Image,*** устанавливаем свойство ***Picture Size Mode → fmPictureSizeModeZoom.***

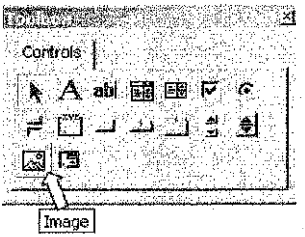


Рисунок 34

Изменяем свойство ***Visible*** = False (при открытии формы рисунок невидим).

Для того, чтобы отобразить рисунок на форме, добавим командную кнопку со следующими свойствами: ***Name*** – cmdGrafic, ***Caption*** – «Отобразить график». Данной кнопке будет соответствовать процедура обработки события **cmdGrafic_Click()**.

```
Private Sub cmdGrafic_Click()
imgGrafik.Visible = True
End Sub
```

Для закрытия формы, по аналогии с предыдущей вкладкой, добавим вторую кнопку со свойствами: ***Name*** – cmdexit2, ***Caption*** – «Выход» и закрепленной процедурой обработки события **cmdexit2_Click**.

4.2.4. Вкладка «Корни»

На данной вкладке (рис. 33) для сравнительного анализа найденных корней в Excel и в среде VBA, размещаем следующие элементы управления:

- ✓ ***Текстовые поля*** – ***TextBox*** – для вывода значений корней, рассчитанных на рабочем листе Excel «Корни».
- ✓ ***Сетка*** – ***MS FlexGrid*** – для вывода отрезков отделения и уточнённых значений корней функции, установив следующие свойства: ***Cols*** = 7, ***Rows*** = 4, ***FixedCols*** = 0, ***FixedRows*** = 1.

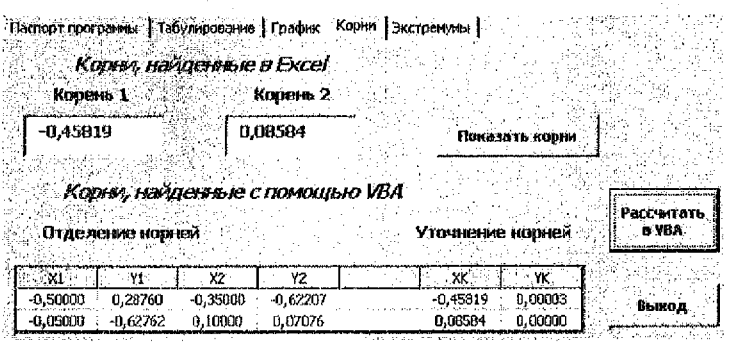


Рисунок 35

✓ ***Кнопки – CommandButton*** – для вызова процедуры обработки события **cmdKorniExc_Click** *(вывод значений корней с рабочего листа)* и процедуры обработки события **cmdKorniVBA_Click** (*вывод рассчитанных в VBA отрезков отделения и корней*). Первая кнопка имеет соответственно следующие свойства: ***Name*** – cmdKorniExc, ***Caption*** – «Показать корни», а вторая: ***Name*** – cmdKorniVBA, ***Caption*** – «Расчитать в VBA».

```
Private Sub cmdKorniExc_Click()
'считывание информации с листа Excel
Worksheets("Корни").Activate
txtx1.Text = Round(Range("A41").Value, 5)
txtx2.Text = Round(Range("A44").Value, 5)
End Sub
```

```
Private Sub cmdKorniVBA_Click()
a = -2
b = 1
n = 20
h = (b – a) / n
Kol_Korney = 0
'цикл для подсчёта количества корней
Do
 Kol_Korney_o = Kol_Korney
 Kol_Korney = 0
 For x = a To (b – h / 2) Step h
  x1 = x :  y1 = fun(x1) :      x2 = x + h :  y2 = fun(x2)
  If y1 * y2 <= 0 Then Kol_Korney = Kol_Korney + 1
 Next x
 If Kol_Korney_o < Kol_Korney Then h = h / 10
Loop While Kol_Korney_o < Kol_Korney
'заполнение «шапки» (заголовков столбцов) сетки MSFlexGrid
GrKorni.Rows = Kol_Korney + 1
GrKorni.Row = 0
GrKorni.Col = 0 :  GrKorni.Text = "X1"
GrKorni.Col = 1 :  GrKorni.Text = "Y1"
GrKorni.Col = 2 :  GrKorni.Text = "X2"
GrKorni.Col = 3 :  GrKorni.Text = "Y2"
GrKorni.Col = 5 :  GrKorni.Text = "XK"
GrKorni.Col = 6 :  GrKorni.Text = "YK"
For i = 0 To 6
With GrKorni
.ColAlignment(i) = 3
End With
```

```
    Next i
    i = 1
    h1 = (b – a) / n
    'вывод результатов табулирования в сетку MSFlexGrid
    For x = a To (b – h1 / 2) Step h1
     x1 = x
     y1 = fun(x1)
     x2 = x + h1
     y2 = fun(x2)
     If y1 * y2 <= 0 Then
      GrKorni.Row = i
      GrKorni.Col = 0 :  GrKorni.Text = Format(x1, "0.00000")
      GrKorni.Col = 1 :  GrKorni.Text = Format(y1, "0.00000")
      GrKorni.Col = 2 :  GrKorni.Text = Format(x2, "0.00000")
      GrKorni.Col = 3 :  GrKorni.Text = Format(y2, "0.00000")
      xk = met_dich(x1, x2, 0.00001)
      yk = fun(xk)
      GrKorni.Col = 5: GrKorni.Text = Format(xk, "0.00000")
      GrKorni.Col = 6: GrKorni.Text = Format(yk, "0.00000")
      i = i + 1
     End If
    Next x
    End Sub
```

В указанной выше процедуре обработки события **cmdKorniVBA_Click** происходит вызов процедуры-функции **met_dich** с передачей в неё значений трёх параметров: **x1** – значение левой границы отделённого отрезка, **x2** – значение правой границы, **0.00001** – требуемая точность расчётов.

Данная пользовательская функция расположена в ***стандартном модуле VBA*** текущего проекта и реализует алгоритм метода дихотомии для уточнения корней функции на отделённом отрезке, описанный в разделе 2.1.2.:

```
Function met_dich(x1, x2, E)
y = fun(x1)
Do While Abs(x2 – x1) > E
xk = (x1 + x2) / 2
y1 = fun(xk)
If y * y1 < 0 Then x2 = xk Else x1 = xk :  y = y1
Loop
met_dich = (x1 + x2) / 2
End Function
```

Для закрытия формы добавим кнопку со свойствами: ***Name*** – cmdexit3, ***Caption*** – «Выход» и закрепленной процедурой обработки события **cmdexit3_Click**.

4.2.5. Вкладка «Экстремумы»

На данной вкладке (рис. 36) для сравнительного анализа найденных экстремумов в Excel и в среде VBA размещаем следующие элементы управления:

- ✓ ***Текстовые поля – TextBox*** – для вывода значений локальных и глобальных экстремумов функции, рассчитанных на рабочем листе Excel «Экстремумы».
- ✓ ***Сетки – MS FlexGrid*** – для вывода значений локальных и глобальных экстремумов функции, рассчитанных в среде VBA.

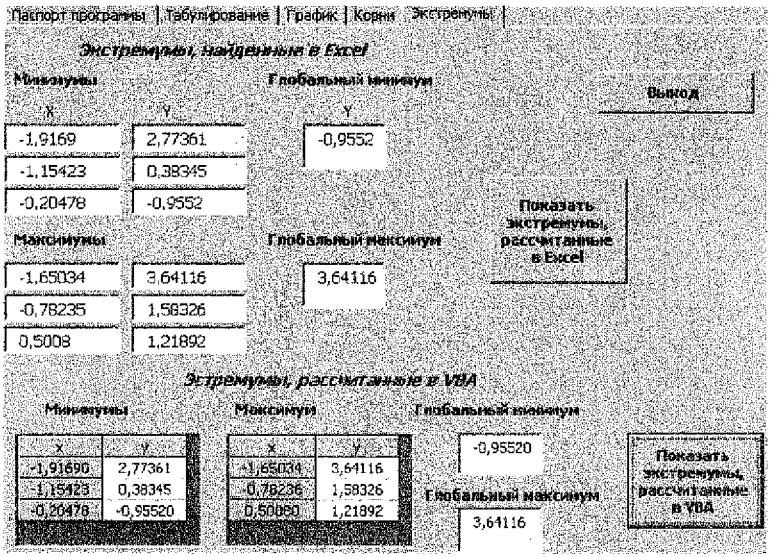


Рисунок 36

- ✓ ***Кнопки – CommandButton*** – для вызова процедуры обработки события **cmdExtrem_Click** *(вывод значений экстремумов с рабочего листа)* и процедуры обработки события **extremymVBA_Click** (*вывод рассчитанных экстремумов в VBA*). Первая кнопка имеет соответственно следующие свойства: ***Name*** – cmdExtrem, ***Caption*** – «Показать экстремумы, рассчитанные в Excel», а вторая: ***Name*** – extremymVBA, ***Caption*** – «Показать экстремумы, рассчитанные в VBA».

Для закрытия формы добавим кнопку со свойствами: ***Name*** – cmdexit4, ***Caption*** – «Выход» и закрепленной процедурой обработки события **cmdexit4_Click**.

```
Private Sub extremymVBA_Click()
Dim ArMin() As Single
Dim ArMax() As Single
a = -2 :  b = 1
ep = 0.00001
n = 20
```

```
h = (b – a) / n
Kol_max = 0 :  Kol_min = 0
i = 1
k = 1
' определение количества экстремумов
For x = (a + h) To (b – h / 2) Step h
y = fun(x)
x1 = x – h :  y1 = fun(x1)
x2 = x + h :  y2 = fun(x2)
If y > y1 And y > y2 Then Kol_max = Kol_max + 1
If y < y1 And y < y2 Then Kol_min = Kol_min + 1
Next x
' переопределение массивов для хранения экстремумов
ReDim ArMin(Kol_min, 2)
ReDim ArMax(Kol_max, 2)
' нахождение экстремумов методом золотого сечения
For x = (a + h) To (b – h / 2) Step h
y = f_1(x)
x1 = x – h :  y1 = fun(x1)
x2 = x + h :  y2 = fun(x2)
If y > y1 And y > y2 Then
' вызов функции пользователя с параметрами
 XMax = Meth_gs(x1, x2, ep, "max")
 ArMax(i, 1) = XMax :  ArMax(i, 2) = fun(XMax)
 i = i + 1
 End If

If y < y1 And y < y2 Then
' вызов функции пользователя с параметрами
 XMin = Meth_gs(x1, x2, ep, "min")
 ArMin(k, 1) = XMin :  ArMin(k, 2) = fun(XMin)
 k = k + 1
 End If
Next x
FGMin.Rows = Kol_min + 1
FGMax.Rows = Kol_max + 1
' заполнение «шапки» сеток MS FlexGrid
FGMin.Row = 0 :  FGMin.Col = 0 :  FGMin.Text = "x"
FGMin.Row = 0 :  FGMin.Col = 1 :  FGMin.Text = "y"
FGMax.Row = 0 :  FGMax.Col = 0 :  FGMax.Text = "x"
FGMax.Row = 0 :  FGMax.Col = 1 :  FGMax.Text = "y"
```

```
'вывод результатов по уточнению экстремумов в сетку MS FlexGrid
For kk = 1 To k – 1
 FGMin.ColAlignment(0) = 3 :  FGMin.ColAlignment(1) = 3
 FGMin.Row = kk
 FGMin.Col = 0
 FGMin.Text = Format(ArMin(kk, 1), "0.00000")
 FGMin.Col = 1
 FGMin.Text = Format(ArMin(kk, 2), "0.00000")
Next kk
For ii = 1 To i – 1
 FGMax.ColAlignment(0) = 3 :  FGMax.ColAlignment(1) = 3
 FGMax.Row = ii :  FGMax.Col = 0
 FGMax.Text = Format(ArMax(ii, 1), "0.00000")
 FGMax.Col = 1
 FGMax.Text = Format(ArMax(ii, 2), "0.00000")
Next ii
'нахождение глобальных экстремумов и вывод их на форму
'вынесено для самостоятельной разработки
End Sub
```

В указанной выше процедуре обработки события **cmdExtrem_Click** происходит вызов процедуры-функции **Meth_gs** с передачей в неё значений четырёх параметров:

a – значение левой границы отделённого отрезка, **b** – значение правой границы, **ep** – требуемая точность расчётов, **vid** – тип экстремума: максимум или минимум.

Данная пользовательская функция расположена в **стандартном модуле VBA** текущего проекта и реализует алгоритм метода золотого сечения для нахождения локальных экстремумов функции на отделённом отрезке, описанный в разделе 2.1.3.:

```
Function Meth_gs(a, b, ep, vid)
t = (1 + Sqr(5)) / 2
x1 = b – (b – a) / t :  x2 = a + (b – a) / t
y1 = fun (x1) :  y2 = fun (x2)
While Abs(b – a) > ep
  If (y1 <= y2 And vid = "min") Or (y1 >= y2 And vid = "max") Then
    b = x2: x2 = x1 :  x1 = b – (b – a) / t
    y2 = y1 :  y1 = fun(x1)
  Else
    a = x1: x1 = x2 :  x2 = a + (b – a) / t
    y1 = y2 :  y2 = fun(x2)
  End If
Wend
Meth_gs = (a + b) / 2
End Function
```

5. РЕАЛИЗАЦИЯ В СИСТЕМЕ MATHCAD

5.1. Используемые инструменты

Для решения уравнения с одним неизвестным в СКМ Mathcad используется встроенная функция ***root***, которая, в зависимости от типа задачи, может включать либо два, либо четыре аргумента и, соответственно, использует разные алгоритмы поиска корней:

a) root(f(x), x);

b) root(f(x), x, a, b);

где f(x) – скалярная функция, определяющая уравнение f(x) = 0;

x – имя скалярной переменной, относительно которой решается уравнение;

a, b – границы интервала, внутри которого происходит поиск корня.

При использовании синтаксиса **a)** необходимо задавать ***начальное значение*** переменной. Данная функция, находит и выводит только один корень, ближайший к начальному приближению.

При использовании синтаксиса **b)** необходимо учитывать, что внутри интервала не должно находиться более одного корня и значения f(a) и f(b) должны иметь разный знак, иначе будет выдано сообщение об ошибке.

Поиск корня уравнения осуществляется итерационным методом с заданной точностью, за которую отвечает системная переменная TOL = 0,001.

5.2. Порядок выполнения

Определяем функцию и строим ее график, по которому будем определять приближения для поиска корней.

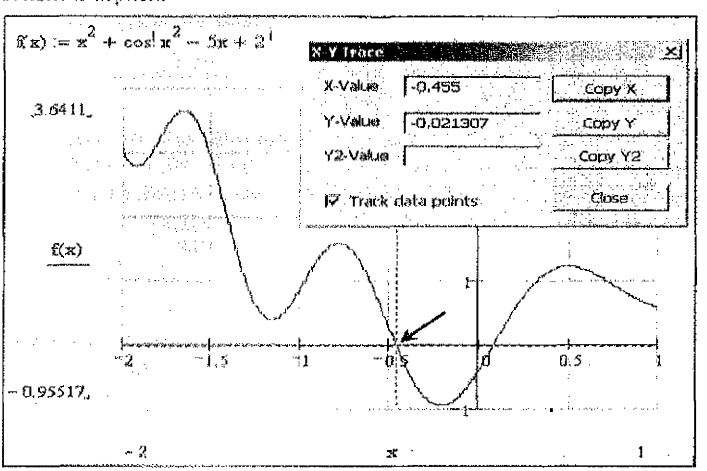


Рисунок 37

Построение таблицы значений функции показано на рисунке 38.

С помощью функции ***root*** уточняем нули функции (рис. 39), используя возможность по графику определять начальные приближения с помощью команды ***п. м. Формат → Графики → Трассировка*** (рис. 37).

$a := -2 \qquad b := 1 \qquad n := 20$

$$h := \frac{(b - a)}{n}$$

$i := 0..20$

$x_i := a + i \cdot h \qquad Y_i := f(x_i)$

$x =$

	0
0	-2
1	-1.85
2	-1.7
3	-1.55
4	-1.4
5	-1.25
6	-1.1
7	-0.95
8	-0.8
9	-0.65
10	-0.5
11	-0.35
12	-0.2
13	-0.05
14	0.1
15	0.25
16	0.4
17	0.55
18	0.7
19	0.85
20	1

$Y =$

	0
0	3.04234
1	2.91237
2	3.56956
3	3.31807
4	1.82443
5	0.63673
6	0.45478
7	1.10282
8	1.57701
9	1.24178
10	0.2876
11	-0.62207
12	-0.95484
13	-0.62762
14	0.07076
15	0.75019
16	1.14729
17	1.20403
18	1.02186
19	0.76578
20	0.58385

Рисунок 38

Изменяем формат полученных корней до шести знаков после запятой командой из ***п.м. Формат → Результат*** → вкладка ***Формат номера*** → Format: ***десятичное число → Число десятичных мест – 6*** (***Format → Result*** → вкладка ***Number Format*** → Format: ***Decimal → Number of decimal places – 6***).

$f(x) := x^2 + \cos(x^2 - 5x + 2)$

Задаём начальное приближение 1-го корня

$x1 := 0.05$

Уточняем 1-ый корень функции

$x2 := root(f(x1), x1)$

$x2 = 0.085841$

Задаем начальное приближение 2-го корня

$x3 := -0.4$

Уточняем 2-ой корень функции

$x4 := root(f(x3), x3)$

$x4 = -0.458188$

Рисунок 39

Рассмотрим порядок нахождения крайних левых минимума и максимума функции.

Поиск экстремумов функции осуществляем разными способами:

1) на основании второго достаточного условия существования экстремума функции, описанного в разделе 2.1.3.;
2) с использованием функций ***Maximize*** и ***Minimize.***

Способ 1: Решаем уравнение вида $f'(x) = 0$, т.е. находим корни первой производной с помощью функции ***root*** (рис. 40).

Вычисляем вторую производную функции, в которую подставляем нули первой производной (рис. 41). По знаку второй производной в этих точках определяем характер экстремума (***максимум*** или ***минимум).***

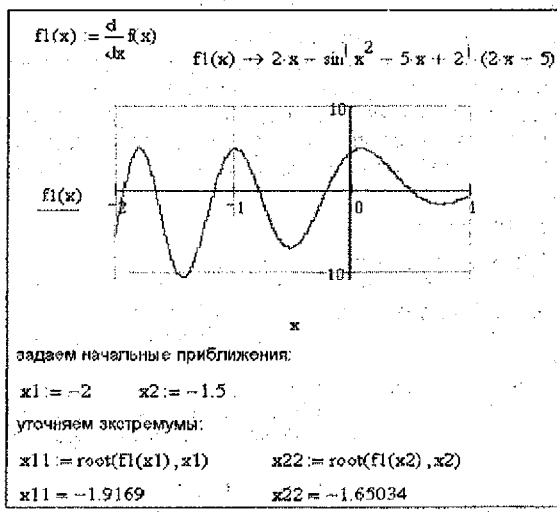


Рисунок 40

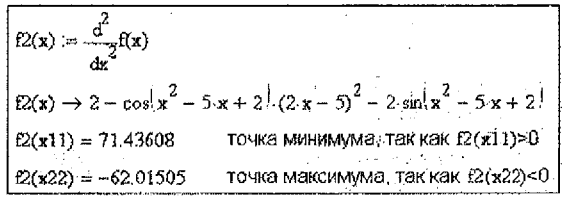


Рисунок 41

Способ 2: Находим локальный минимум и локальный максимум с помощью функций ***Minimize*** и ***Maximize*** (рис. 42).

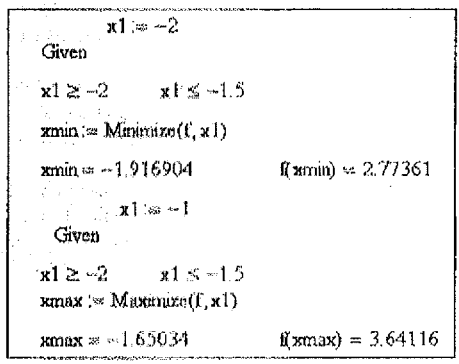


Рисунок 42

По аналогии находим остальные локальные экстремумы функции.

6. ЛИТЕРАТУРА

Литература по курсу «Информатика» ввиду быстрой смены компьютерного программного обеспечения быстро устаревает и, как правило, отсутствует в достаточных количествах в библиотеке. Поэтому ниже приводится максимальный, на момент составления методического пособия, перечень литературы, которую студенты могут использовать при изучении нового материала и выполнении курсовой работы. Этот постоянно пополняемый перечень находится в локальной вычислительной сети и доступен каждому пользователю.

6.1. Основная литература

1. Быков, В.Л. Основы информатики. – Брест: БГТУ, 2003. – 253 с.
2. Быков, В.Л. Основы информатики: пособие для студентов технических специальностей / В.Л. Быков, Ю.П. Ашаев – Брест: БрГТУ, 2006. – 430 с.
3. Информатика. Базовый курс. 2-е издание / Под ред. С.В. Симоновича. – СПб.: Питер, 2004. – 640 с.
4. Плис, А.Н. MathCAD: Математический практикум для инженеров и экономистов. учеб. пособ.. 2-е изд. / А.Н. Плис, Н.А. Сливина. – М.: Финансы и статистика, 2003. – 656 с.
5. Попов, А.А. Excel: Практическое руководство: учебное пособие для вузов. – М.: ДессКом, 2000. – 301 с.
6. Рабин, Ч. Эффективная работа с Microsoft Word 2000. – СПб: Питер, 2000. – 944 с.
7. Численные методы / И.В. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – М.: МАИК "Наука". Физматлит и др., 2000. – 622 с.

6.2. Дополнительная литература

Литература по работе с табличным процессором Excel

8. Бондаренко, С. Excel 2007. Популярный самоучитель / С. Бондаренко, М. Бондаренко – СПб.: Питер, 2007. – 224 с.
9. Вадзинский, Р.Н. Статистические вычисления в среде Excel. – СПб.: Питер, 2008. – 608 с.
10. Васильев, А. Excel 2007 на примерах. – СПб.: БХВ-Петербург, 2007. – 656 с.
11. Гельман, В.Я. Решение математических задач средствами Excel. Практикум. – СПб: Питер, 2002. – 240 с.
12. Кардберг Конрад. Бизнес-анализ с помощью Excel, пер. с англ. – К.: Диалектика, 1997. – 448 с.
13. Курбатова, Е. Самоучитель Excel 2003. – М.: Вильямс, 2005. – 352 с.
14. Курицкий, Б.Я. Поиск оптимальных решений средствами Excel 7.0. – СПб: БХВ-Петербург, 1997. – 384 с.
15. Минько, А.А. Принятие решений с помощью Excel. – М.: Эксмо, 2007. – 240 с.
16. Минько, А.А. Сводные таблицы и диаграммы в Excel. – М.: Эксмо, 2008. – 208 с.
17. Орвис, В. Excel для ученых, инженеров и студентов, пер. с англ. – К.: Юниор, 1999. – 528 с.
18. Рудикова, Л. Microsoft Excel для студента. – СПб.: БХВ-Петербург, 2005. – 368 с.
19. Рычков, В. Microsoft Excel 2000: Краткий курс. – СПб: Питер, 2000. – 320 с.
20. Рычков, В. Самоучитель Excel 2000. – СПб: Питер, 2000. – 336 с.
21. Соколенко, А. Microsoft Office Excel 2007. Просто как дважды два. – М.: ЭКСМО, 2007. – 256 с.
22. Соломенчук, В. Excel 2007. Начали! – СПб.: Питер, 2007. – 128 с.

Литература по работе с СКМ MathCAD

23. Алексеев, Е.Р Решение задач вычислительной математики в пакетах MathCAD 12, MATLAB 7, Maple 9 / Е.Р. Алексеев, О.В. Чеснокова – М.: НТ Пресс, 2006. – 496 с.

24. Бертяев, В.Д. Техническая механика на базе MathCAD. Практикум. – СПб.: БХВ-Петербург, 2005. – 752 с.
25. Бидасюк, Ю. Mathsoft MathCAD 12: самоучитель. – М.: Вильямс, 2005. – 224 с.
26. Васильев, А.Н. Mathcad 13 на примерах. – СПб.: БХВ-Петербург, 2006. – 528 с.
27. Гурский, Д.А. MathCAD для студентов и школьников. Популярный самоучитель. – СПб.: Питер, 2005. – 400 с.
28. Гурский, Д.А. Вычисления в MathCAD 12 / Д.А Гурский, Е.С. Турбина – СПб.: Питер, 2006. – 544 с.
29. Каганов, В.И. Компьютерные вычисления в средах Excel и MathCAD. – М.: Горячая линия-Телеком, 2003. – 328 с.
30. Кирьянов, Д. MathCAD 13 в подлиннике. – СПб.: БХВ-Петербург, 2007. – 608 с.
31. Кирьянов, Д. Самоучитель MathCAD 13. – СПб.: БХВ-Петербург, 2007. – 528 с.
32. Кудрявцев, Е.М. MathCAD 8.0. Символьное и численное решение разнообразных задач. – М.: ДМК, 2000. – 320 с.
33. Макаров, Е.Г. MathCAD. Учебный курс. – СПб.: Питер, 2009. – 384 с.
34. Макаров, Е.Г. Инженерные расчеты в MathCAD. Учебный курс. – СПб: Питер, 2004. – 448 с.
35. Макаров, Е.Г. Сопротивление материалов на базе Mathcad. – СПб.: БХВ-Петербург, 2004. – 512 с.
36. Очков, В. MathCAD 12 для студентов и инженеров. – СПб.: БХВ-Петерб., 2005. – 464 с.
37. Очков, В. MathCAD 14 для студентов, инженеров и конструкторов. – СПб.: БХВ-Петербург, 2007. – 368 с.
38. Половко, А.М. MathCAD для студента / А.М. Половко, И.В. Ганичев – СПб.: БХВ-Петербург, 2006. – 336 с.
39. Салманов, О.Н. Математическая экономика с применением MathCAD и Excel. – СПб.: БХВ-Петербург, 2003. – 464 с.
40. Черняк, А.А. MathCAD: программная среда, примеры и обучение. / А.А. Черняк, М.А. Хотомцева – СПб.: Питер, 2003. – 256 с.
41. Черняк, А.А. Высшая математика на базе MathCAD. Общий курс / А.А. Черняк, Ж.А. Черняк, Ю.А. Доманова – СПб.: БХВ-Петербург, 2005. –608 с.

Литература по программированию в среде Visual Basic for Application

42. Билл, Дж. Применение VBA и макросов в Microsoft Excel. / Дж. Билл, Т. Сирстад – М.: Вильямс, 2005. – 624 с.
43. Васильев, А.А. VBA в Office 2000. Учебный курс. / А.А. Васильев, А.Ф. Андреев – СПб.:Питер, 2001. – 432 с.
44. Гайдышев, И. Решение научных и инженерных задач средствами Excel, VBA и C++. – СПб.: БХВ-Петербург, 2004. – 512 с.
45. Гарнаев, А.Ю. VBA в подлиннике. – СПб.: БХВ-Петербург, 2005. – 848 с.
46. Гарнаев, А.Ю. VBA: самоучитель. 2-е изд. – СПб.: БХВ-Петербург, 2004. – 560 с.
47. Кузьменко, В. VBA 2003: самоучитель. – М.: Бином, 2004. – 432 с.
48. Кузьменко, В. Программирование на VBA 2003. – М.: Бином, 2005. – 880 с.
49. Саймон, Дж. Программирование в Excel. – М.: Вильямс, Диалектика, 2002. – 336 с.
50. Слепцова, Л.Д. Программирование на VBA в Microsoft Office 2007. – М.: Вильямс, 2007. – 432 с.
51. Слепцова, Л.Д. Программирование на VBA: самоучитель. – М.: Вильямс. Диалектика, 2004. – 384 с.
52. Уокенбах, Дж. профессиональное программирование на VBA в Excel 2003. – М.: Вильямс. Диалектика, 2005. – 800 с.

Учебное издание

Составители:
Гучко Ирина Михайловна,
Рамская Людмила Константиновна

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению курсовой работы
по дисциплине «Информатика»
для студентов технических специальностей
дневной формы обучения

Ответственный за выпуск: ***Гучко И.М.***
Редактор: ***Боровикова Е.А.***
Компьютерная вёрстка: ***Кармаш Е.Л.***
Корректор: ***Никитчик Е.В.***

Подписано в печать 15.12.2014 г. Формат 60x84 1/16. Бумага писчая.
Усл.-п.л. 3,02. Усл.-изд.л. 3,25. Тираж 50 экз. Заказ № 1070.
Отпечатано на ризографе учреждение образования "Брестский государственный технический университет". 224017, Брест, ул. Московская, 267.