МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «Информатика»

для студентов строительных специальностей

дневной формы обучения

второй семестр

издание 4-е дополненное и переработанное

строительный факультет

Группа _____

Фамилия _____

Имя _____

Отчество _____

Вариант _____

м/тел. +375 (___) ____-___-___

Брест 2020

УДК 004

Практикум предназначен для студентов первого курса строительного факультета, изучающих дисциплину «Информатика». В него входят задания для лабораторных работ и задания для их защиты. В теоретической части изложены методические рекомендации по работе с приложением Google Таблицы с поддержкой среды программирования Apps Script и системой компьютерной математики MathCAD.

Составители: В.А. Кофанов, к.т.н., доцент
Т.Г. Хомицкая, ст. преподаватель
И.В. Тузик, ст. преподаватель

Рецензент: доцент кафедры прикладной математики и информатики БрГУ
имени А.С. Пушкина, доцент, к.ф.-м.н. О.В. Матысик

Учреждение образования

Общие указания

Практикум предназначен для организации самостоятельной, практической и лабораторной работы студентов во втором семестре изучения дисциплины «Информатика».

Перед началом работы необходимо привести информацию на титульном листе, то есть должны быть указаны данные ее владельца (Ф.И.О., группа, номер мобильного телефона), а также вариант заданий.

На текущей странице в блок «Индивидуальное задание» необходимо вклеить листок с вариантами заданий, полученный у преподавателя. Специальный блок «Условие» на странице с лабораторной работой заполняется простым переписыванием условия задачи с бланка индивидуального задания.

При выполнении лабораторных работ результаты вычислений заносятся в отчет в том же виде, в котором они отображаются на экране монитора. Ведение записей выполняется четко и разборчиво шариковой ручкой (блок-схемы – карандашом). Неправильные (ошибочные) записи на страницах практикума необходимо исправлять с использованием корректирующих средств (корректирующие ленты, штрих-корректоры и т.п.).

Каждая лабораторная работа считается выполненной только при наличии отметки преподавателя о ее защите, подтвержденной его подписью (личной печатью) на стр. 4. Данные из этого листа служат основанием для допуска к итоговым испытаниям (зачету, экзамену).

Индивидуальное задание

Отметки о защите лабораторных работ

№	Наименование работы	Защита	Примечание
1	Лабораторная работа №1(1,2,3)		
2	Лабораторная работа №2(1,2)		
3	Лабораторная работа №3(1,2,3)		
4	Лабораторная работа №4(1,2,3)		
5	Лабораторная работа №5(1,2,3)		

Итог работы в семестре:

Все лабораторные работы выполнены в полном объеме.

_____

(дата, подпись)

Методические указания к выполнению лабораторных работ

GOOGLE ТАБЛИЦЫ

Условное форматирование

Условное форматирование позволяет задать определенный формат диапазона ячеек в зависимости от содержимого этого диапазона. Как правило, когда вы задаете условия для форматирования диапазона ячеек, инструмент проверяет каждую ячейку диапазона для определения ее соответствия заданным вами условиям. Затем он применяет выбранный вами для данного условия формат во всех ячейках, удовлетворяющих этому условию. Если содержимое ячейки не отвечает ни одному из заданных условий, форматирование ячейки не меняется.

Предположим, что необходимо отформатировать ячейки массива B7:J17 (рисунок ***1***), в которых *число больше* (*меньше*) *чисел, содержащихся в соседних восьми ячейках*. Восемь соседних ячеек есть только у ячеек диапазона C8:I16.

	A	B	C	D	E	F	G	H	I	J
6	x \ y	-3	-2.25	-1.5	-0.75	0	0.75	1.5	2.25	3
7	0	-0.275	0.362	0.581	0.265	-0.416	-1.098	-1.414	-1.194	-0.557
8	0.6	0.538	1.084	1.317	0.997	0.148	-0.920	-1.754	-1.975	-1.466
9	1.2	-0.901	0.461	1.407	1.383	0.516	-0.536	-1.086	-0.880	-0.198
10	1.8	-1.963	-1.250	0.607	1.354	0.558	-0.111	0.283	0.538	-0.493
11	2.4	-0.304	-1.965	-0.717	0.972	0.259	0.206	1.559	0.588	-1.539
12	3	0.474	-0.821	-1.779	0.397	-0.275	0.332	1.974	-0.099	0.004
13	3.6	-0.500	0.564	-1.893	-0.170	-0.859	0.275	1.302	0.195	1.537
14	4.2	0.408	0.559	-0.951	-0.563	-1.288	0.120	-0.004	1.368	0.486
15	4.8	1.985	-0.117	0.485	-0.698	-1.412	-0.013	-1.133	1.531	0.204
16	5.4	0.792	0.231	1.562	-0.594	-1.189	-0.025	-1.468	-0.021	1.472
17	6	-0.546	1.406	1.692	-0.351	-0.696	0.118	-0.973	-1.550	0.549

Рисунок 1 – Результат условного форматирования

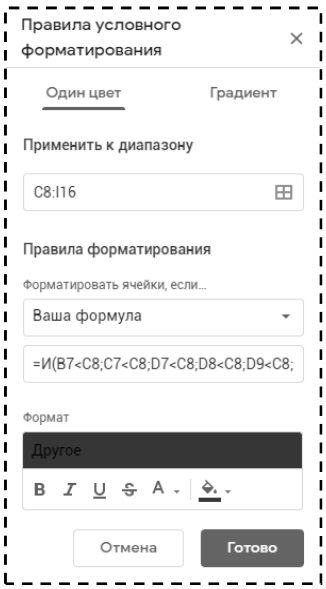


***Рисунок 2 – Боковая панель* Правила условного форматирования**

Выделяем эти ячейки, выбираем в главном меню пункт **Формат**/**Условное форматирование** и создаем критерии на боковой панели **Правила условного форматирования** (рисунок ***2***).

Для критерия **Ваша формула** составляется формула относительно левой верхней ячейки выделенного диапазона (C8) и начинается со знака «=». После знака «=» используется встроенная функция **И(условие1;условие2;...)**, которая позволяет проверить одновременное выполнение всех содержащихся в ней условий. Кроме формулы необходимо задать формат ячейки (например, цвет фона ячейки – красный), который будет применен к ней в случае, если формула выдаст значение **истина**. Созданное правило после применения автоматически тиражируется во все ячейки выделенного диапазона. Та как каждая ячейка имеет свои восемь соседних ячеек, то в формуле правила все ссылки должны быть относительными.

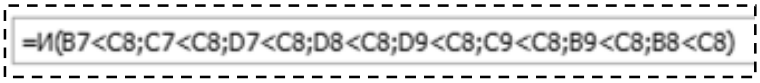


Рисунок 3 – Правило для условного форматирования

На рисунке ***3*** показано правило для случая, когда необходимо найти ячейки с числом большим *чисел, содержащихся в соседних восьми ячейках* (локальный максимум). Для случая, когда необходимо найти ячейки с числом *меньшим чисел, содержащихся в соседних восьми ячейках* (локальный минимум), в формуле знаки «<» нужно поменять на знак «>».

Построение графика

С помощью пункта меню **Вставка**/**Диаграмма** в Google Таблицах можно построить график в декартовой системе координат.

На основе предыдущего примера построим график, представляющий собой вертикальное сечение поверхности (функции двух переменных) при фиксированном значении одной переменной. Ниже приведем порядок построения такого графика (рисунок ***4***).

- Выделяем с помощью мыши два диапазона ячеек A6:J6 и A13:J13, удерживая клавишу Ctrl.
- Выбираем пункт меню **Вставка**/**Диаграмма**.

- На боковой панели **Редактор диаграмм** на вкладке **Настройки** указываем **Тип диаграммы** – **Графики** – **Плоский график**, отмечаем пункты: **Строки/Столбцы**, **Заголовки – значение столбца A** и **Ярлыки – значения строки 6**.
- На боковой панели **Редактор диаграмм** на вкладке **Дополнительные** в пункте **Название диаграмм и осей** задаем для элементов **Название диаграммы**, **Название горизонтальной оси** и **Название вертикальной оси** следующие параметры: название, шрифт, размер, выравнивание и цвет.
- На боковой панели **Редактор диаграмм** на вкладке **Дополнительные** в пункте **Серии** задаем цвет, толщину и тип линии, размер и формат точек. Для выделения особым форматом одной точки в том же пункте нажимает кнопку **Добавить**, выбирает нужную точку (3.6 : -2.25) и задаем для этой точки размер, цвет и форму.

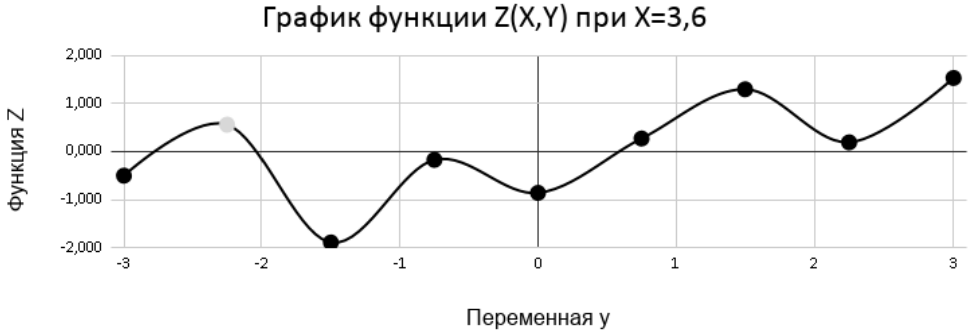


Рисунок 4 – Гладкий график с точками

Дополнение «Solver» для уточнения локальных экстремумов

С помощью дополнения **Solver** уточним, например, один локальный минимум функции Z(x,y), протабулированные значения которой показаны на рисунке ***1***. Пусть это будет локальный минимум в координатах X = 4.8, Y = 0. В этом случае точное значение локального минимума будет находиться в диапазоне 4.2..5.4 по оси *x* и в диапазоне -0.75..075 по оси *y*. Отмеченные данные занесем в отдельные ячейки рабочего листа (рисунок ***5***). В ячейку B9 вставляем функцию Z(x,y), в которой вместо переменных X и Y делаем ссылки на ячейки B7 и B8 соответственно.

На боковой пенели **Solver** (рисунок ***6***) задаем **Set Objective** B9 (в целевой ячейке обязательно должна находиться формула). Поскольку наша цель – минимизировать значение в этой ячейке, устанавливаем переключатель **To: Min**. Затем определяем изменяемые ячейки (**By Changing**), от которых зависит результат в целевой ячейке B9. Это ячейки B7 и B8. Далее задаем ограничения по нажатию кнопки **Add** (B7≥B2 и B7≤B3, т.е. найденное значение *X* должно находиться в диапазоне 4.2..5.4; B8≥B4 и B8≤B5, т.е. найденное значение *Y* должно находиться в диапазоне -0.75..075). Заданные ограничения отобразятся в списке **Subject To**.

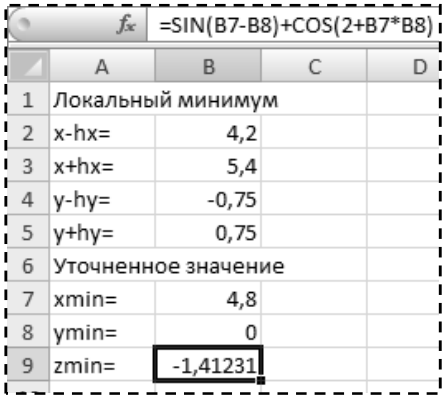


Рисунок 5 – Исходные данные для определения локального минимума

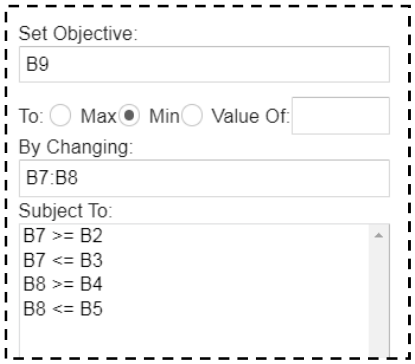


Рисунок 6 – Настройки боковой панели Solver

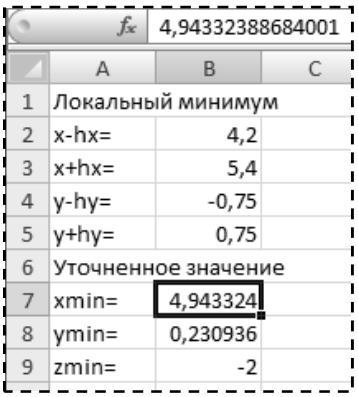


Рисунок 7 – Результат работы дополнения Solver

Чтобы начать процесс решения задачи, необходимо выбрать метод поиска **Standard LSGRG Nonlinear** и щелкнуть по кнопке **Solve**. Через некоторое время отобразится информация о том, что решение найдено. В ячейках B7 и B8 появится значение *X* и *Y*, при котором функция имеет минимальное значение на участке $x \in [4.2..5.4]$ и $y \in [-0.75..075]$. В ячейке B9 – значение локального минимума функции Z(x,y).

Значение локального максимума определяется аналогично. Все отличие состоит в том, что в отдельные ячейки заносится информация о локальном максимуме и на боковой панели **Solver** переключатель устанавливается в положение **To: Max**.

СРЕДА ПРОГРАММИРОВАНИЯ APPS SCRIPT

Apps Script – это скриптовая платформа, разработанная компанией Google для создания не громоздких приложений на платформе G Suite. Благодаря его сравнительной лёгкости освоения, приложения могут создавать даже пользователи, не программирующие профессионально.

Программный код в Apps Script основан на языке программирования JavaScript, который оформляется в виде совокупности функций, каждая из которых предназначена для решения определенной задачи.

Редактор скриптов

Для открытия редактора скриптов необходимо выбрать в Google Таблицах пункт меню **Инструменты**/**Редактор скриптов**. В открывшейся среде Apps Script отобразится пустой **Проект** скриптов, который автоматически прикрепляется к файлу Google Таблиц, из которого он был вызван. В этом случае файл Google Таблиц становится контейнером будущего **Проекта** скриптов и неразрывно связан с ним. В левом верхнем углу необходимо задать имя **Проекта**.

Любое изменение кода в редакторе скриптов не сохраняется автоматически, поэтому необходимо периодически сохранять код путем выбора пункта меню **Файл**/**Сохранить**, либо воспользоваться одноименной кнопкой на панели инструментов.

Каждый новый **Проект** содержит по умолчанию один скриптовый файл (например, Код.gs). Для изменения имени файла необходимо справа от имени файла раскрыть дополнительное меню (пиктограмма в виде треугольника) и выбрать пункт **Rename**. Это же меню (пункт **Удалить**) можно использовать, чтобы удалить файл. Для добавления скриптового файла необходимо воспользоваться пунктом меню **Файл**/**Создать**/**Скрипт**.

Создание функций

Функции – ключевая концепция в JavaScript. Любая функция это объект, и, следовательно, ею можно манипулировать как объектом.

Объявление функции состоит из ключевого слова **function** и следующих за ним частей (таблица 1):

- Имя функции.
- Список аргументов (принимаемых функцией) заключенных в круглые скобки и разделенных запятыми.
- Инструкции, которые будут выполнены после вызова функции, заключают в фигурные скобки. Сами же инструкции разделяются между собой знаком «;».

Любую функцию можно использовать в качестве макроса в Google Таблицах (**Инструменты**/**Макросы**/**Импортировать**). Однако для правильного использования такая функция не должна содержать список аргументов и, как правило, должна содержать набор инструкций, которые выполняют определенные действия без возврата вместо себя результата.

Если функция после выполнения своих инструкций должна вернуть результат, то такая функция обязательно содержит инструкцию **return**. Такого рода функции могут использоваться в Google Таблицах при создании формул.

Таблица 1 – Пример создания функций в Apps Script

Способ использования	Синтаксис	Пример

Функция, без возврата результата	**function** *имя* ([*список_аргументов*]) {
	[*инструкции*];
	}
		function Tab_F() {
		//Объявление переменной
		//и присвоение ей значения
		var a = 5;
		}

Функция, с возвратом результата	**function** *имя* ([*список_аргументов*]) {
	[*инструкции*];
	return *выражение*;
	}
		function Fun_F(x) {
		//возврат результата
		return 2 * x + 3;
		}

Примеры функций, показанных в таблице 1, содержат комментарии (например, «//возврат результата») и инструкцию **var**.

Комментарии – это заметки для того, кто составляет код процедуры, Apps Script их игнорирует. Строка комментариев начинается со знаков «//». Комментарий можно поместить после любого оператора. Другими словами, если Apps Script встречает знаки «//», он игнорирует остальной текст этой строки.

Инструкция **var** позволяет объявлять переменные, а оператор «=» – присваивать начальные значения.

Вызов функций

Перед вызовом функций, возвращающих результат, обязательно нужно задать в скобках значения всех аргументов, если таковые имеются. Так, функцию Fun_F(x) из таблицы 1 можно вызвать следующим образом:

- из любой функции Apps Script (пример кода: var y = Fun_F(4));
- из формулы в ячейке рабочего листа. Точно так же, как и встроенные функции (рисунок ***8***).

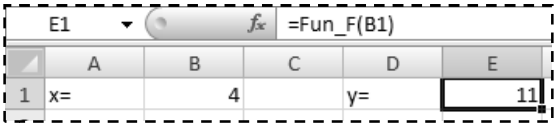


Рисунок 8 – Использование функции Fun_F(x) на рабочем листе

Вызов функций, не содержащих списка аргументов и не возвращающих результат, осуществляется следующими способами:

- из любой функции Apps Script (пример кода: Tab_F());
- из Apps Script через пункт меню **Выполнить**/**Запустить функцию**;
- из Google Таблицы через пункт меню **Инструменты**/**Макросы** (прежде импортировав ее туда).

Пошаговое выполнение инструкций

При отладке программы часто возникает необходимость проследить за тем, как выполняются отдельные инструкции или же какие значения принимают те или иные переменные на различных этапах выполнения. Для достижения этих целей в Apps Script реализован режим отладки.

Перевести программу в режим отладки в некоторый, заранее известный момент ее выполнения можно с помощью установки **точки останова**. Точка останова отмечается щелчком мыши по номеру строки и отображается красной точкой слева от номера строки. Чаще всего **точку останова** помещают в строке программного кода непосредственно перед оператором, который, по вашему предположению, является причиной возникновения ошибки. Как только по ходу выполнения программы достигается оператор, которому назначена **точка останова**, работа программы приостановится. В этот момент можно проверить текущие значения любых переменных (рисунки 9 и *10*).

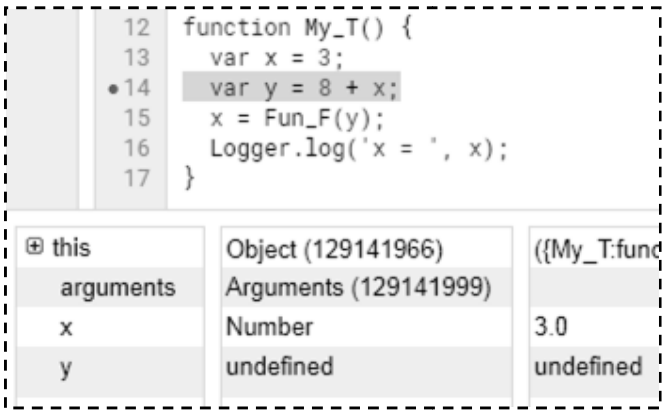


Рисунок 9 – Режим отладки с точкой останова и текущей остановкой в строке 14

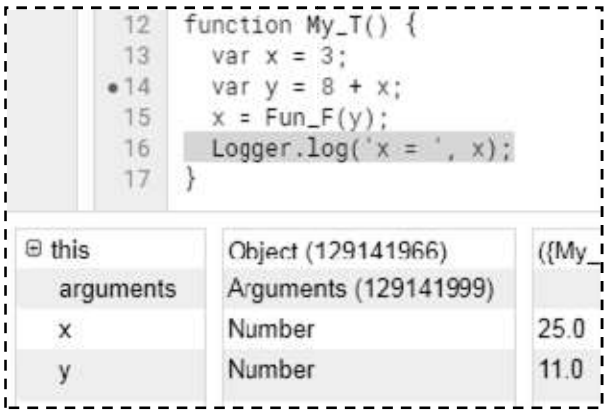


Рисунок 10 – Режим отладки с точкой останова в строке 14 и текущей остановкой в строке 16

Дальнейшее управление отладкой осуществляется с помощью элементов управления, расположенных на панели инструментов (рисунок ***11***):

Рисунок 11 – Элементы управления отладкой на панели инструментов

- кнопка 1 – перейти к следующей **точке останова**;
- кнопка 2 – выйти из режима отладки;
- кнопка 3 – перейти к следующей инструкции;
- кнопка 4 – обойти следующую инструкцию.

Нажатие на кнопку 3 позволит перейти из строки 14 на строку 15. Повторное нажатие на кнопку 3 приведет к тому, что программа отладки перейдет в функцию Fun_F. Если в переходе к функции Fun_F нет необходимости, то можно в этом случае нажать на кнопку 4 и перейти к строке 16 (рисунок *10*).

Количество размещаемых **точек останова** неограниченно, поэтому в разных строках программы можно разместить столько точек останова, сколько сочтете необходимым. Единственное условие – нельзя размещать точки останова в строках комментариев и строках с инструкциями, которые Apps Script на самом деле не выполняет, например, в строках с объявлениями переменных без присваивания им начальных значений (пример кода: var a;).

При выполнении инструкций могут возникнуть ошибки, отличные от синтаксических, – ошибки выполнения. Такие ошибки могут проявиться только в процессе выполнения программы, приводящие к ее остановке. При этом отображается сообщение об ошибке. Подробные сведения об ошибке можно посмотреть в отчете, который доступен через пункт главного меню **Вид**/**Отчет о выполнении**.

При необходимости можно намеренно отправить сообщение в **Отчет о выполнении** программы с помощью класса **Logger.log()**. Пример его использования можно увидеть на рисунках *9* и *10*.

Ввод-вывод данных на рабочий лист

JavaScript – это объектно-ориентированный язык программирования. Это означает, что основными его элементами являются объекты, которые расположены, как правило, в иерархическом порядке (рисунок *12*).

Например, чтобы обратиться к ячейке Google Таблицы необходимо соблюдать следующую структуру:

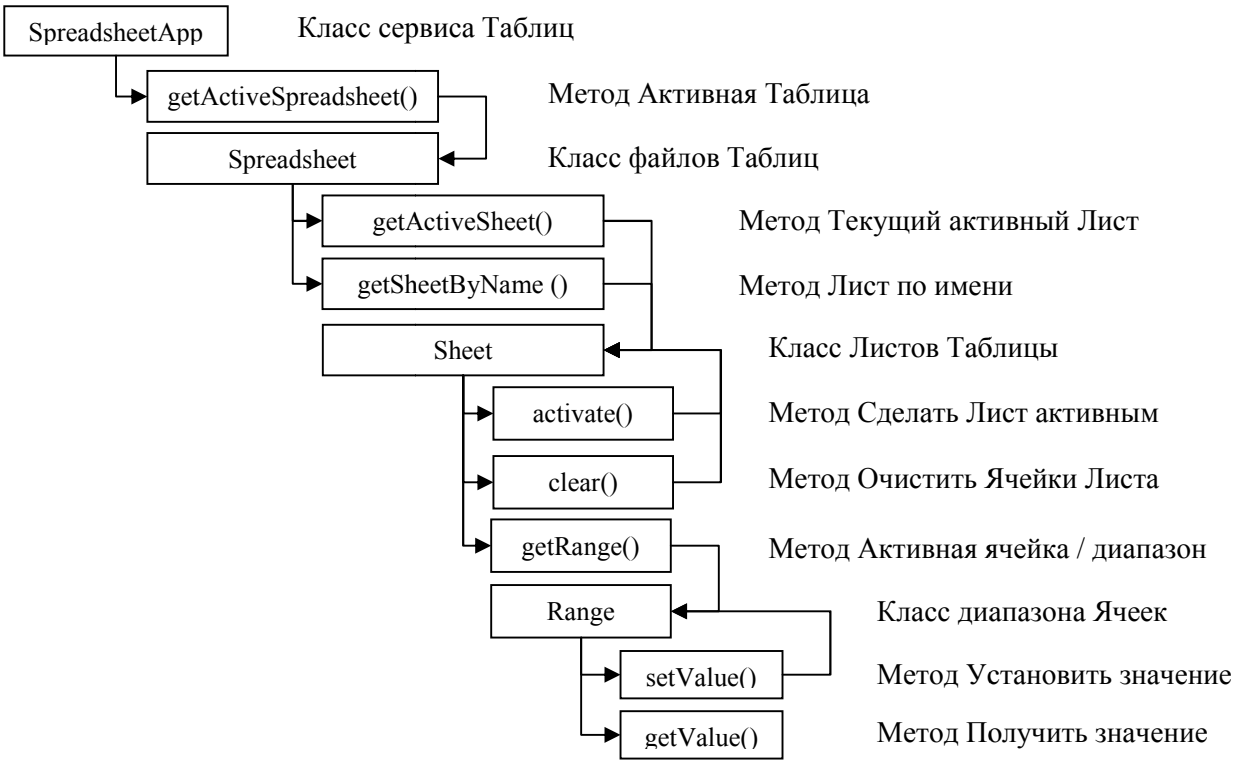


Рисунок 12 – Элементы структуры сервиса Google Таблицы

Опираясь на схему, приведенную на рисунке *12*, можно составить конструкцию обращения к объекту **Лист**, который в данный момент является активным:

```
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet().
```

Чтобы сделать существующий Лист «Лист2» активным, необходима следующая конструкция:

```
SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Лист2').activate().
```

Для удобства частого обращения к объекту необходимо передать его некоторой переменной, например:

```
var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Лист2').
```

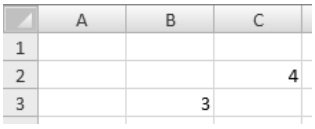
В этом случае, чтобы очисть ячейки Листа «Лист2», достаточно использовать следующий код:

```
sheet.clear().
```

Для обращения к определенной ячейке рабочего листа необходимо использовать метод **getRange(**"*адрес ячейки*"**)** или **getRange(***номер строки ячейки*, *номер столбца ячейки***)**. В зависимости от совершаемых над ячейкой действий будут зависеть применяемые к ней методы, примеры которых приведены в таблице 2.

Таблица 2 – Примеры использования методов **getValue**() и **setValue**().

Условие	Присвоить переменной f значение из ячейки **C2** рабочего листа	Вывести на рабочий лист в ячейку **B3** результат выражения 2^2-1

	// sheet – текущий лист
	var f =sheet.getRange('C2').getValue();
	// или
	var f = sheet.getRange(2,3).getValue();
		// sheet – текущий лист
		sheet.getRange('B3').setValue(3);
		// или
		sheet.getRange(3,2).setValue(3);

Результат	f = 4	В ячейке B3 число 3

Диалоговые окна

В Apps Script есть два метода – **alert** и **prompt**, которые позволяют отображать простые диалоговые окна для ввода и вывода данных. Эти методы принадлежат классу экземпляра пользовательского интерфейса Google Таблиц (рисунок *13*).

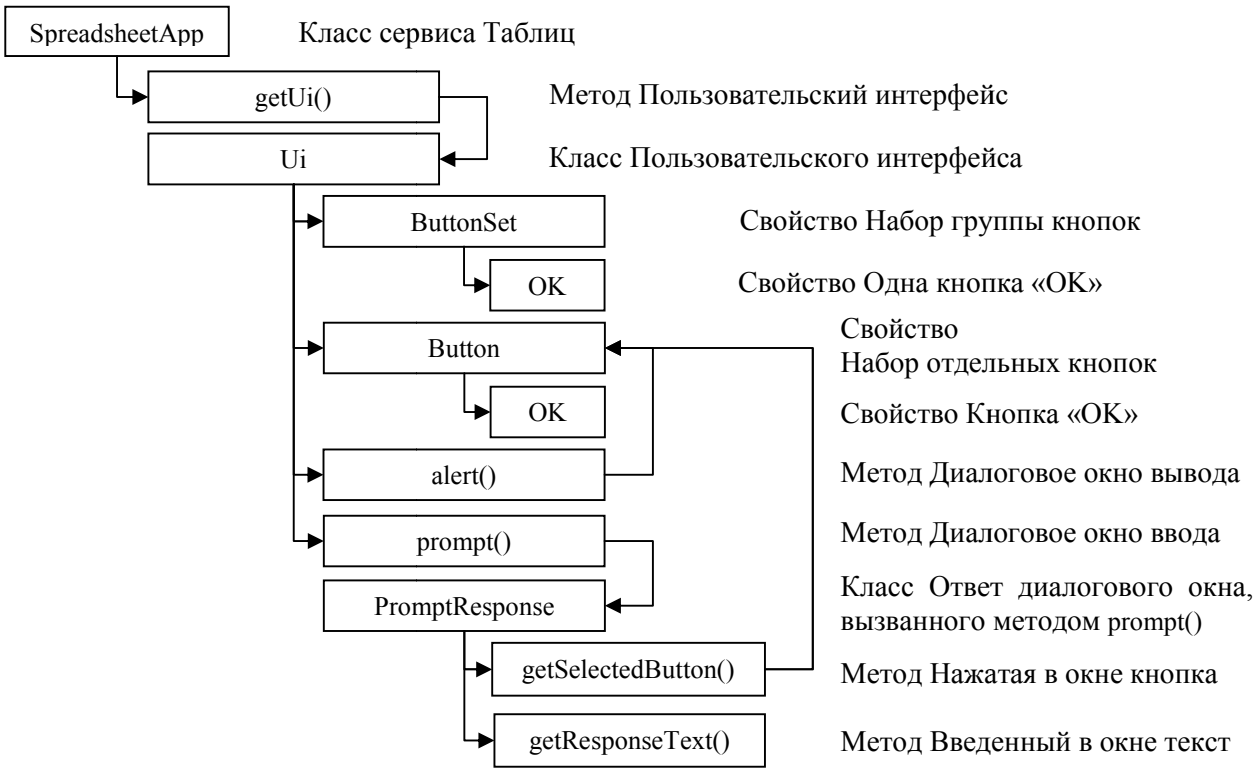


Рисунок 13 – Элементы структуры пользовательского интерфейса сервиса Google Таблицы

Для того чтобы обратиться к классу Пользовательского интерфейса необходимо:

```
var ui = SpreadsheetApp.getUi().
```

Далее согласно схеме на рисунке *13* можно вызвать диалоговое окно для ввода сообщения, синтаксис которого выглядит следующим образом:

```
prompt(title, prompt, buttons),
```

где title – заголовок диалогового окна (текст);
prompt – сообщение в диалоговом окне (текст);
buttons – кнопки в диалоговом окне (свойство ButtonSet).

Любое значение, введенное в диалоговое окно (prompt), воспринимается как текст. Поэтому для ввода численного значения (рисунок *14*), текст необходимо преобразовать в число при помощи функции Number:

```
var a = Number(ui.prompt('Ввод данных', 'Введите значение a', ui.ButtonSet.OK).getResponseText()).
```

Для вывода значения переменной в диалоговое окно можно использовать метод alert:

```
alert(title, prompt, buttons).
```

Численное значение в свою очередь необходимо преобразовать в текст для отображения в диалоговом окне (рисунок *15*) с помощью функции String():

```
ui.alert('Вывод данных', 'Значение a = ' + String(a), ui.ButtonSet.OK).
```

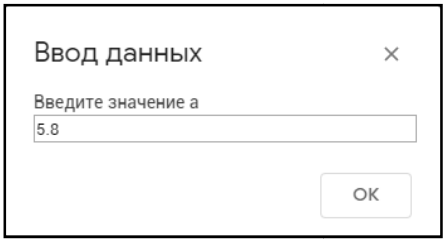


Рисунок 14 – Диалоговое окно метода prompt()

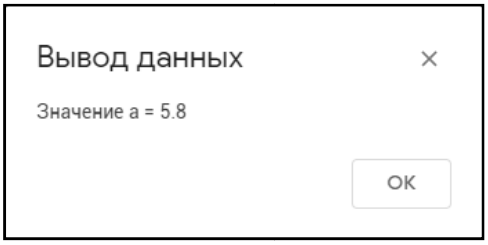


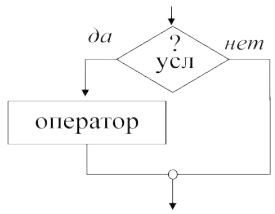
Рисунок 15 – Диалоговое окно метода alert()

Оператор if

Оператор **if** – один из важнейших управляющих операторов. Он организует выполнение заданного блока операторов, если при вычислении указанного условного выражения будет получено значение ИСТИНА, и не делает ничего, если будет получено значение ЛОЖЬ. Указанная инструкция **if** может записываться так, как показано в таблице 3.

Таблица 3 – Примеры краткой формы разветвляющейся структуры

Блок-схема	Оператор	один	несколько

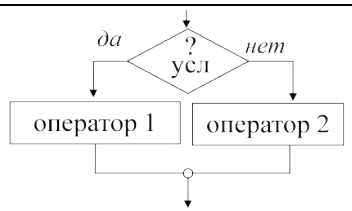
	Синтаксис	if (*условие*) *оператор* ;	if (*условие*) {
			операторы;
			};

	Пример	if (x > 5) ui.alert('x больше 5');	if (x > 5) {
			ui.alert('x не меньше 5');
			ui.alert('x больше 5');
			};

Для более сложных ситуаций, когда на основании некоторого условия необходимо выбрать одну из двух различных последовательностей операторов, используется оператор **if...else** (таблица 4).

Таблица 4 – Пример полной формы разветвляющая структуры

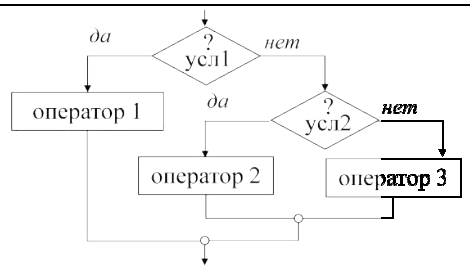
Блок-схема	Синтаксис	Пример

	if (*условие*) {
	оператор 1 ;
	} else {
	оператор 2 ;
	};
		If (x > 5) {
		ui.alert('x больше 5');
		} else {
		ui.alert('x не больше 5');
		};

Часто при написании программ возникают такие ситуации, когда приходится проверять не одно, а два или больше условий. В этом случае можно поместить один оператор **if** внутри другого оператора **if** (или оператор **if...else** внутри оператора **if...else**), что называется вложением операторов (таблица 5).

Таблица 5 – Пример оператора **if** с вложенным оператором **if**

Блок-схема	Синтаксис	Пример

	if (*условие 1*) {
	оператор 1 ;
	} else {
	if (*условие 2*) {
	оператор 2 ;
	} else {
	оператор 3 ;
	};
	};
		if (x > 5) {
		ui.alert('x больше 5');
		} else {
		if (a < 5) {
		ui.alert('x меньше 5');
		} else {
		ui.alert('x равно 5');
		};
		};

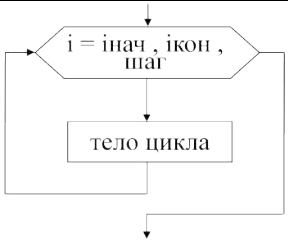
Операторы цикла for, while и do...while

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

Оператор **for** относится к операторам создания циклов простейшего типа. С его помощью блок операторов выполняется заданное количество раз, известное до начала выполнения цикла. Пример такого оператора приведен в таблице 6.

Таблица 6 – Пример оператора цикла **for**

Блок-схема	Синтаксис	Пример

	for (*i* = *інач* ; *i* < *ікон* ; i = i + *шаг*) {
	тело цикла ;
	};
		for (var i = 0; i <= 10; i = i + 2) {
		ui.alert(String(Math.pow(i, 2)));
		}

В таблице 6 *i* – любая числовая переменная, значение которой меняется в начале каждого цикла (интерации). Параметр *інач* и *ікон* – это числовые выражения, задающие начальное и конечное значения переменной цикла *i*. Числовая переменная *шаг* задает приращение, на которое увеличивается переменная цикла *i* при каждой итерации.

Операторы циклов **while** и **do...while** (в отличие от оператора **for**) выполняются не заданное количество раз, а сколь угодно долго – до тех пор, пока выполняется некоторое логическое условие. Отличие их друг от друга состоит в том, что **while** это цикл с предусловием, а **do...while** – цикл с постусловием. В цикле с предусловием логическое условие проверяется перед началом цикла (таблица 7), а цикле с постусловием логическое условие проверяется после выполнения цикла (таблица 8).

Таблица 7 – Пример оператора цикла с предусловием **while**

Блок-схема	Синтаксис	Пример

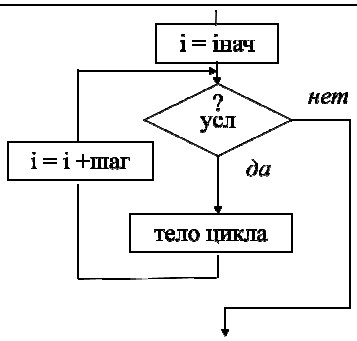
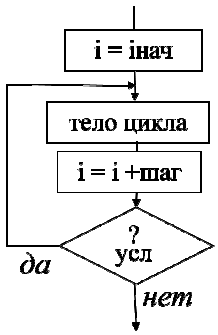
	*i = інач* ;
	while (*условие*) {
	тело цикла ;
	i = i + шаг ;
	};
		var i = 0;
		while (i <= 10) {
		ui.alert(String(Math.pow(i, 2)));
		i = i + 2;
		};

Таблица 8 – Пример оператора цикла с постусловием **do...while**

Блок-схема	Синтаксис	Пример

	*i = інач* ;
	do {
	тело цикла ;
	i = i + шаг ;
	} while (*условие*) ;
		var i = 0;
		do {
		ui.alert(String(Math.pow(i, 2)));
		i = i + 2;
		} while (i <= 10);

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

Скрытая область

Скрытая область – это часть MathCAD-документа, которая присутствует в документе, участвует в расчетах, но не видна на экране. Необходимость в создании такой области возникает в двух случаях: для уменьшения размера документа и для сокрытия информации от посторонних глаз.

Чтобы создать скрытую область, сделайте следующее:

- В главном меню MathCAD выберите пункт **Insert**/**Area** (**Вставка**/**Область**). На экране появятся две горизонтальные линии с указателями слева. Перетащите их мышью в начало и в конец области, которую надо скрыть.
- Щелкните правой кнопкой мыши в выделенной линиями области. В открывшемся контекстном меню выберите **Collapse** (**Свернуть**). Выделенная область исчезнет. Останется лишь одна горизонтальная линия с указателем.

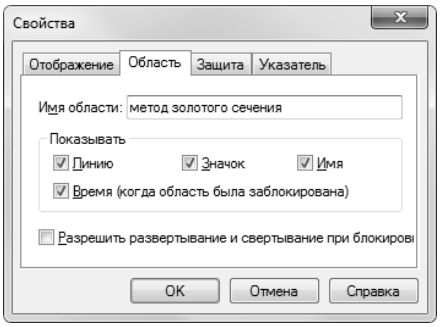


Рисунок 16 – Диалоговое окно Properties (Свойства) для скрытой области

Выражения, спрятанные в скрытой области, продолжают работать в документе, хотя и не видны на экране. Щелкнув правой кнопкой мыши на скрытой области, выберите в контекстном меню пункт **Properties** (**Свойства)**. В диалоговом окне **Properties** (**Свойства)** перейдите на вкладку **Area** (**Область)**. На этой вкладке (рисунок *16*) можно вписать заголовок скрытой области, отметить выделение ее линиями и значком, сделать эту область совсем невидимой.

Чтобы открыть скрытую область, нужно сделать двойной щелчок левой кнопкой мыши на линии с указателем. Скрытая область становится открытой и появляется на экране в обрамлении двух линий с указателями. Чтобы удалить скрытую область, щелкните мышью на одной из линий с указателем, выделив ее, и нажмите клавишу **Del**.

Взаимодействие документов

MathCAD предоставляет возможность объединения нескольких документов в один с помощью ссылки на них. Для этого в основном документе делается ссылка на другой документ. В результате оба документа объединяются в один и работают совместно, хотя второго документа и не видно на экране.

Для создания ссылки надо в главном меню MathCAD выбрать пункт **Insert**/**Reference...** (**Вставка**/**Ссылка...**). В открывшемся окне, выбрав **Browse...** (**Обзор...**), укажите путь к документу, который хотите включить в расчет. Нажмите **ОК**. В основном документе появится строка с именем подключенного документа.

Рекомендуется при создании ссылки использовать относительный путь. Указав путь к документу, пометьте флажком пункт **Use relative pass for reference** (**Использовать для ссылки относительный путь**). При этом в строке документа, указывающей на ссылку, появится буква (R). В этом случае ссылка останется действующей, даже если вы переместите основной документ в другую папку. Важно лишь, чтобы оба документа находились на одном диске.

Элементы программирования

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

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

Для создания программного модуля надо:

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

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

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

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

Условный оператор if

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

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

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

- в создаваемой программе установить курсор на свободное место ввода, где должен появиться условный оператор;
- на панели **Программирование** щелкнуть на кнопке **if.** В программе появится шаблон оператора с двумя маркерами;
- в правый маркер ввести условие. Пользуйтесь при этом логическими операторами, вводя их с панели **Булевы операторы**;
- слева от оператора **if** ввести выражение, которое должно выполняться, если условие верно. Если при выполнении условия должно выполняться сразу несколько выражений, надо иметь несколько маркеров. Установите курсор ввода на маркер слева от **if** и нажмите **Add Line** столько раз, сколько строк надо ввести. Обратите внимание на то, что при этом изменяется вид условного оператора. Столбик маркеров появится не слева, а под оператором **if**;
- на следующем свободном маркере (на следующей строке) пишут выражение, выполняемое «иначе», если условие не выполняется;
- выделяют курсором выражение так, чтобы уголок курсора ввода был в конце выражения, и на панели **Программирование** нажимают **otherwise**.

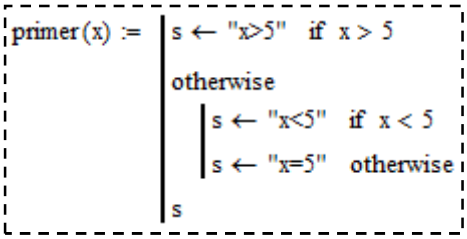


Рисунок 17 – Реализация в MathCAD примера из таблицы 5

Операторы цикла for и while

Панель **Программирование** содержит два оператора цикла, **for** и **while**, работа которых аналогична одноименным операторам Apps Script.

Чтобы записать цикл **while**, выполните следующие действия:

- Установите курсор на свободное место ввода в программе (справа от вертикальной черты).
- На панели **Программирование** нажмите кнопку **while.** Появится шаблон с двумя местами ввода.
- Справа от слова **while** введите условие выполнения цикла. Обычно это логическое выражение.
- В оставшееся поле ввода (внизу под словом **while)** введите выражение, которое вычисляется в цикле.
- Если в цикле надо вычислять несколько выражений, то вначале установите курсор на место ввода и нажмите кнопку **Add Line** столько раз, сколько строк будет содержать цикл. Затем заполните все места ввода, введя нужные выражения. Удалите лишние места ввода.

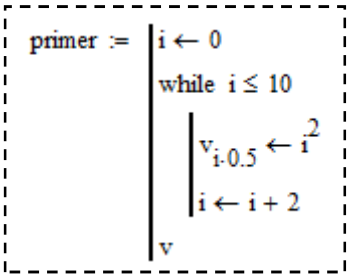


Рисунок 18 – Реализация в MathCAD примера из таблицы 7

В цикле **for** число повторений цикла определяется переменной, задаваемой в начале цикла. Рассмотрим создание такого цикла:

- Установите курсор на свободное место ввода в программе (справа от вертикальной черты).
- На панели **Программирование** нажмите кнопку **for**. Появится шаблон с тремя местами ввода.
- Справа от слова **for** введите имя переменной цикла. После знака ∈ введите диапазон изменения переменной цикла так же, как это делается с помощью дискретной переменной. Переменной цикла может быть ряд чисел, или вектор, или список скаляров, диапазонов, векторов, разделенных запятой.
- В оставшееся поле ввода (внизу, под словом **for**) введите выражение, которое вычисляется в цикле.
- Если в цикле надо вычислять несколько выражений, то вначале установите курсор на место ввода и нажмите кнопку **Add Line** столько раз, сколько строк будет содержать цикл. Затем заполните все места ввода, введя нужные выражения. Удалите лишние места ввода.

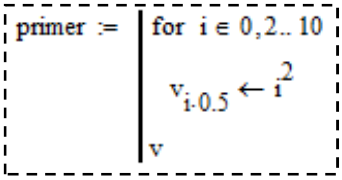


Рисунок 19 – Реализация в MathCAD примера из таблицы 6

ЗАПИСЬ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ В GOOGLE ТАБЛИЦАХ, MATHCAD И APPS SCRIPT

Математические константы

мат. запись	Google Таблицы	MathCAD	Apps Script
число π	ПИ()	π	Math.PI
число e^1	EXP(1)	e^1	Math.exp(1)

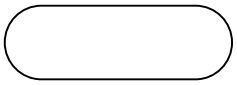
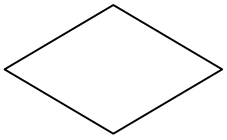
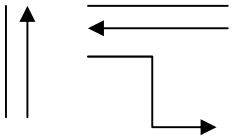
Математические функции

мат. запись	Google Таблицы	MathCAD	Apps Script
|x|	ABS(x)	|x|	Math.abs(x)
$\sqrt{x}$	КОРЕНЬ(x)	$\sqrt{x}$	Math.sqrt(x)
e^x	EXP(x)	e^x	Math.exp(x)
ln x	LN(x)	ln(x)	Math.log(x)
lg x	LOG10(x)	log(x)	Math.log(x)/Math.log(10)
$\log_n x$	LOG(x;n)	log(x,n)	Math.log(x)/Math.log(n)
cos x	COS(x)	cos(x)	Math.cos(x)
sin x	SIN(x)	sin(x)	Math.sin(x)
tg x	TAN(x)	tan(x)	Math.tan(x)
ctg x	COT(x)	cot(x)	1/ Math.tan(x)
arccos x	ACOS(x)	acos(x)	Math.acos(x)
arcsin x	ASIN(x)	asin(x)	Math.asin(x)
arctg x	ATAN(x)	atan(x)	Math.atan(x)
arcctg x	ACOT(x)	acot(x)	Math.PI/2 - Math.atan(x)

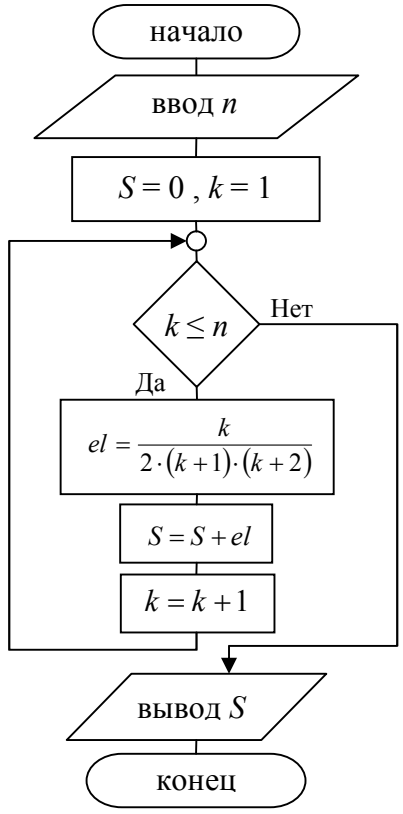
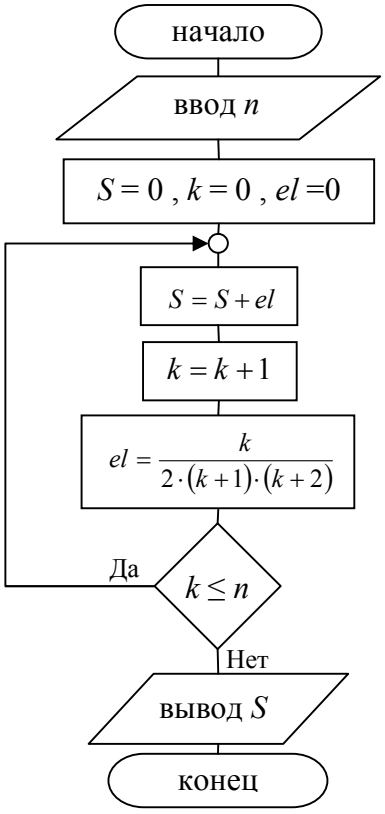
Арифметические операции

мат. запись	Google Таблицы	MathCAD	Apps Script
$-a^n$	– (a^n)	$-a^n$	– (Math.pow(a,n)
$\sin^2 x$	SIN(x)^2	$\sin(x)^2$	Math.pow(Math.sin(x),2)
$\cos^2 x^3$	COS(x^3)^2	$\cos(x^3)^2$	Math.pow(Math.cos(Math.pow(x,3)),2)
$\sqrt[n]{a}$	a^(1/n)	$\sqrt[n]{a}$	Math.pow(a,1/n)
$\frac{a+b}{c \cdot d}$	(a+b)/(c*d)	$\frac{a+b}{c \cdot d}$	(a+b)/(c*d)

ОСНОВНЫЕ ЭЛЕМЕНТЫ БЛОК-СХЕМ

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

Примеры блок-схем для нахождения суммы $S = \frac{1}{3\cdot 4} + \frac{2}{4\cdot 6} + \frac{3}{5\cdot 8} + ...n - \text{раз} = \sum_{k=1}^{n} \frac{k}{2\cdot(k+1)\cdot(k+2)}$

оператор с предусловием **while**	оператор с постусловием **do...while**
	
код функции Summa_S в Apps Script	

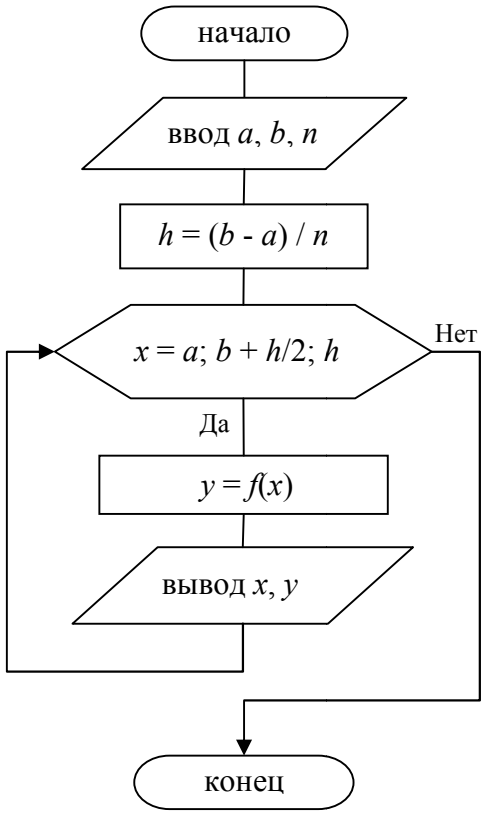
function Summa_S(n) {
  var S = 0, k = 1;
  while (k <= n) {
  var el = k / (2 * (k + 1) * (k + 2));
  S = S + el;
  k = k + 1;
  };
  return S;
}
	function Summa_S(n) {
	  var S = 0, k = 0, el = 0;
	  do {
	  S = S + el;
	  k = k + 1;
	  el = k / (2 * (k + 1) * (k + 2));
	  } while (k <= n);
	  return S;
	}

ЧИСЛЕННЫЕ МЕТОДЫ

Табулирование функции

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

Блок-схема алгоритма табулирования:



Реализация табулирования $y(x) = \cos^2(x) - \sin(x^2 + 2x - 3)$ в MathCAD с помощью программного модуля

$$\text{Tab\_F}(a,b,n) := \left| \begin{array}{l} \left(T_{0,0} \leftarrow \text{"x"} \quad T_{0,1} \leftarrow \text{"y(x)"}\right) \\ h \leftarrow \dfrac{b-a}{n} \\ i \leftarrow 1 \\ \text{for } x \in a, a+h .. b + \dfrac{h}{2} \\ \quad \left| \begin{array}{l} \left(T_{i,0} \leftarrow x \quad T_{i,1} \leftarrow y(x)\right) \\ i \leftarrow i + 1 \end{array} \right. \\ T \end{array} \right.$$

Tab_F(0,3,10) =

	0	1
0	"x"	"y(x)"
1	0	1.14112
2	0.3	1.65167
3	0.6	1.67264
4	0.9	0.76659
5	1.2	-0.61334
6	1.5	-0.77307
7	1.8	0.69462
8	2.1	0.87835
9	2.4	-0.41335
10	2.7	1.07947
11	3	1.51666

Программная реализация в Apps Script:

```
function Tab_f(){
  var  a = 0, b = 3, n = 10;
  var  h = (b - a) / n;
  var  sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange('A1').setValue('a='); sheet.getRange('B1').setValue(a);
  sheet.getRange('A2').setValue('b='); sheet.getRange('B2').setValue(b);
  sheet.getRange('A3').setValue('n='); sheet.getRange('B3').setValue(n);
  sheet.getRange('A4').setValue('h='); sheet.getRange('B4').setValue(h);
  sheet.getRange('A6').setValue('x'); sheet.getRange('B6').setValue('y');
  var rw = 7;
  for (var x = a; x < b + h/2; x = x + h) {
    var  y = Fun_y(x);
    sheet.getRange(rw, 1).setValue(x); sheet.getRange(rw, 2).setValue(y);
    rw = rw + 1;
  }}
```

Результат выполнения в GT:

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

Метод дихотомии (уточнение нуля функции)

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

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

где *c* – середина отрезка [*a*, *b*], *c* = (*b* - *a*) / 2;

ε – точность вычисления (задается пользователем).

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

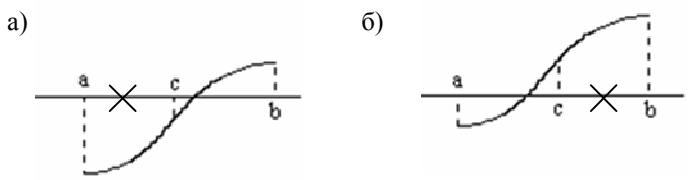
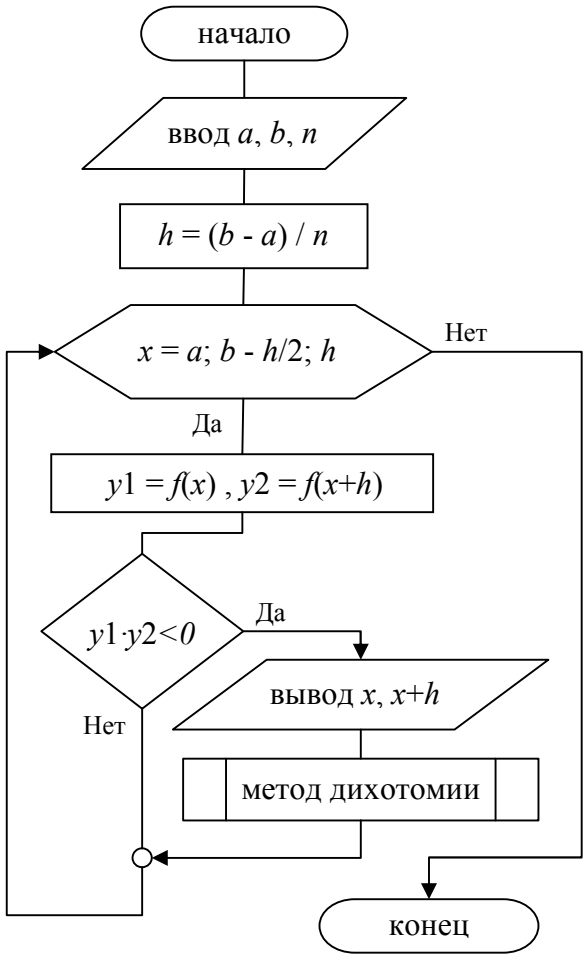
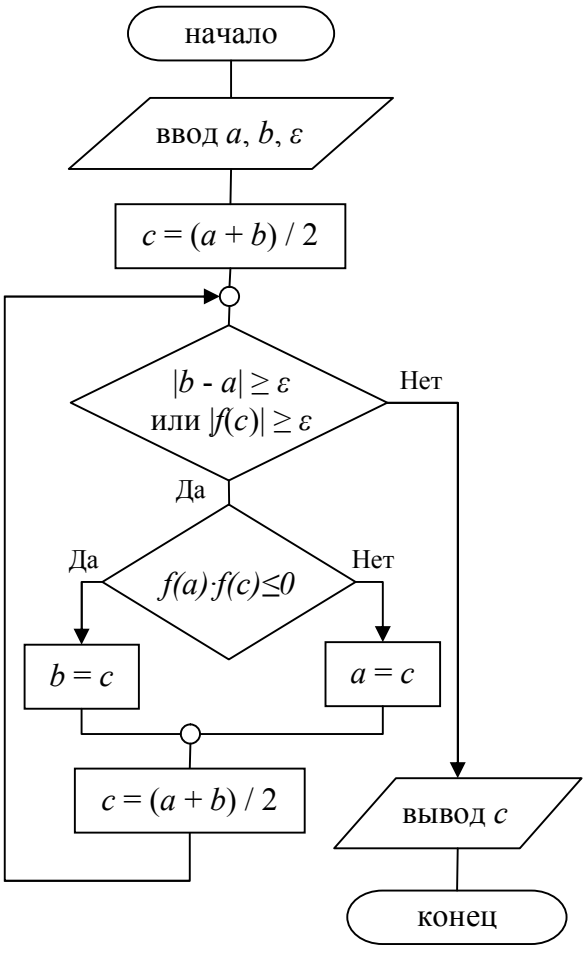


Рисунок 20 – Графическая интерпретация метода дихотомии

Аналитически, процедура «отбрасывания» происходит следующим образом:

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

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

Метод золотого сечения

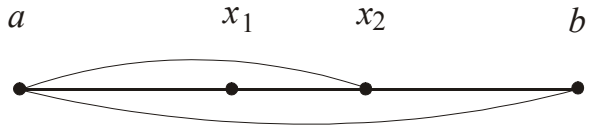
Метод золотого сечения – метод поиска экстремума функции одной переменной на локализованном отрезке. В основе метода лежит принцип деления отрезка в пропорциях **золотого сечения**: деление отрезка на две части так, чтобы отношение длины всего отрезка к длине большей части равнялось отношению длины большей части к длине меньшей части.

$$\frac{b-a}{b-x}=\frac{b-x}{x-a}=t$$

или

$$\frac{b-a}{x-a}=\frac{x-a}{b-x}=t$$

Для того чтобы найти определённое значение функции на заданном отрезке, отвечающее критерию поиска (пусть это будет минимум), рассматриваемый отрезок делится в пропорции золотого сечения в обоих направлениях, то есть выбираются две симметрично расположенные точки x_1 и x_2 такие, что:



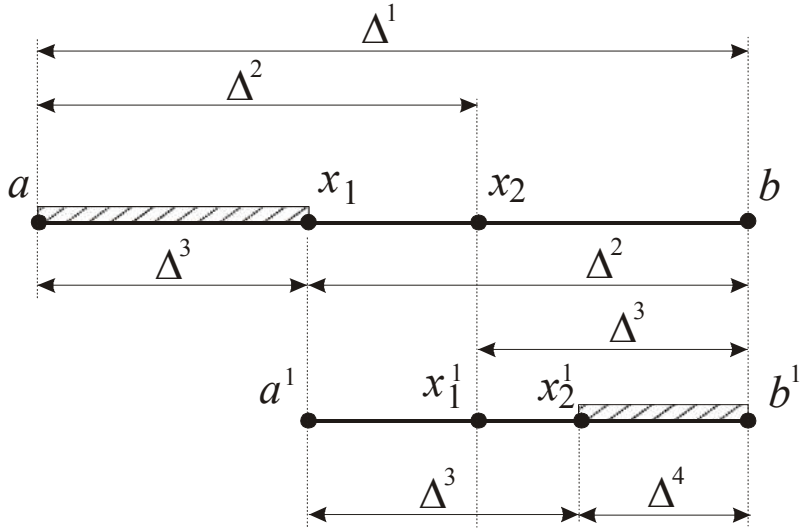
т.е. $\frac{b-a}{b-x_1}=\frac{b-x_1}{x_1-a}=t$ и $\frac{b-a}{x_2-a}=\frac{x_2-a}{b-x_2}=t$

Примечательно, что точка x_1 в свою очередь производит золотое сечение отрезка $[a, x_2]$, т.е.

$$\frac{x_2-a}{x_1-a}=\frac{x_1-a}{x_2-x_1}=t\,.$$

Аналогично, точка x_2 производит золотое сечение отрезка $[x_1, b]$.

Проиллюстрируем на рисунке (отбрасываемый интервал на рисунке будет заштрихован):



где $\Delta_1=b-a$ – исходный интервал;

Δ_2 – интервал, полученный после уменьшения интервала Δ_1 отбрасыванием его левого или правого подынтервала.

Итак, метод золотого сечения состоит в том, что длины последовательных интервалов берутся в фиксированном отношении:

$$\frac{\Delta^1}{\Delta^2}=\frac{\Delta^2}{\Delta^3}=t\,.$$

Поскольку $\Delta^1=\Delta^2+\Delta^3$, то получаем

$$t=\frac{\Delta^1}{\Delta^2}=\frac{\Delta^2+\Delta^3}{\Delta^2}=1+\frac{\Delta^3}{\Delta^2}=1+\frac{1}{t}\Rightarrow t^2-t-1=0\,.$$

Корнем этого уравнения является золотое сечение:

$$t = \frac{\sqrt{5}+1}{2} \approx 1.618033988\ ,\ \frac{1}{t} \approx 0.618$$

Можно записать формулы для точек x_1 и x_2, производящих золотое сечение на интервале $[a, b]$:

$$x_1 = b - \frac{b-a}{t} = b - (b-a)\cdot 0.618\ ;\ x_2 = a + \frac{b-a}{t} = a + (b-a)\cdot 0.618\ .$$

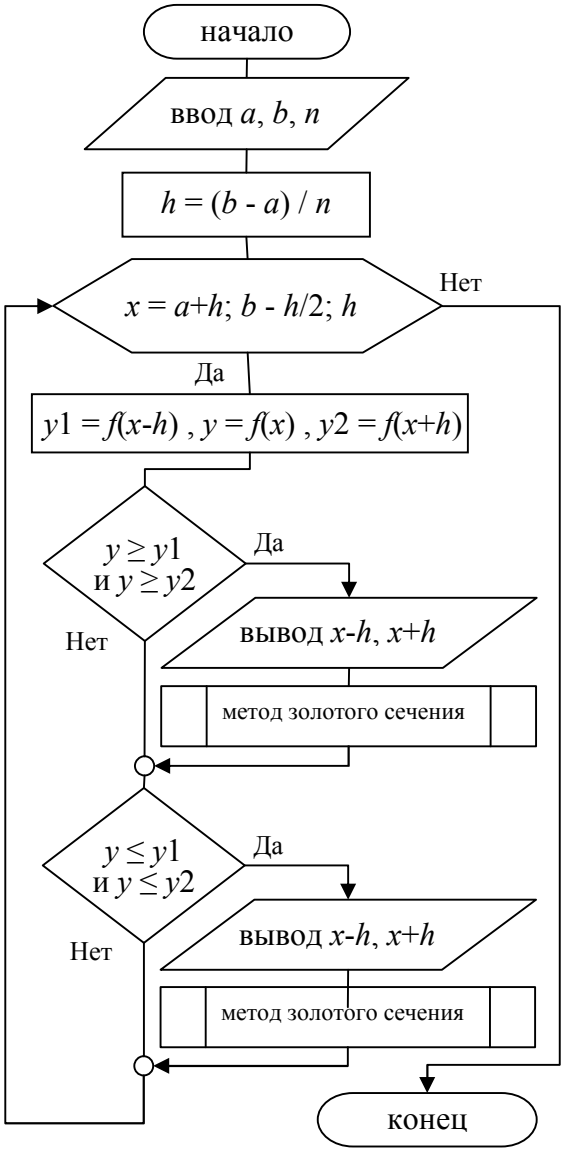
Алгоритм метода золотого сечения следующий.

На первой итерации заданный отрезок делится двумя симметричными относительно его центра точками x_1 и x_2; рассчитываются значения в этих точках. После чего тот из концов отрезка, к которому среди двух точек ближе оказалась та, значение в которой больше (для случая поиска минимума), отбрасывают.

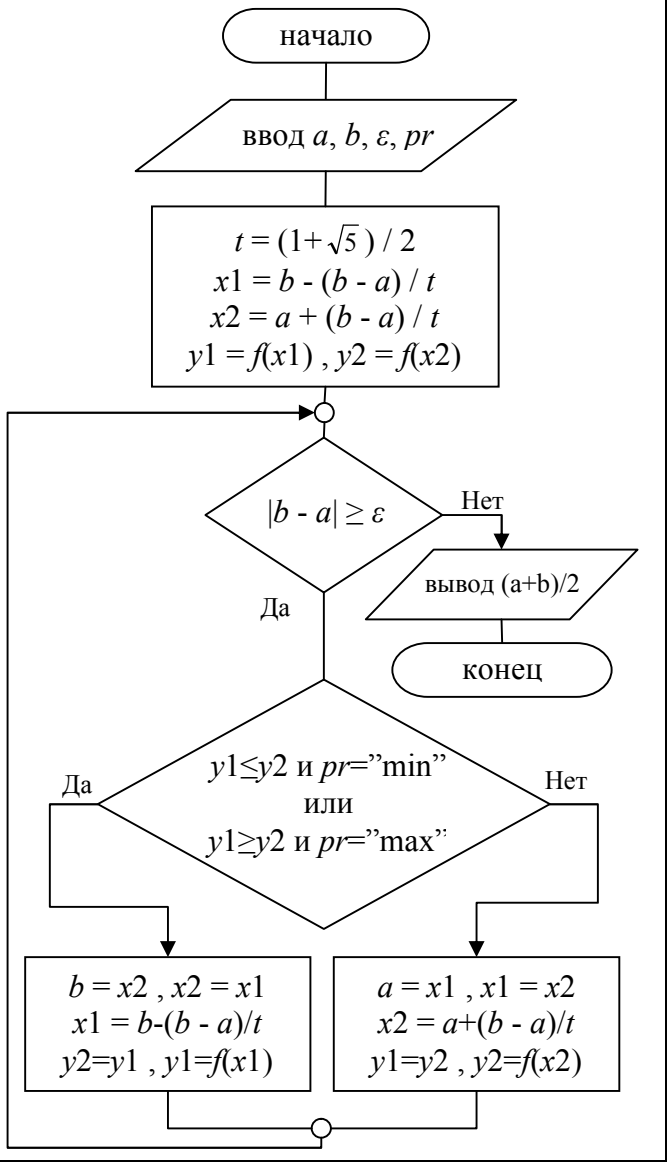
На следующей итерации в силу показанного выше свойства золотого сечения уже надо искать всего одну новую точку.

Процедура продолжается до тех пор, пока не будет достигнута заданная точность.

Блок-схема алгоритма отделения отрезков, содержащих экстремумы



Блок-схема алгоритма метода золотого сечения



Лабораторная работа № 1(1,2,3)

ТЕМА. Основы работы в Apps Script. Создание пользовательских функций (с линейной структурой) в Google Таблицах + Apps Script и в MathCAD.

Условие: Функция: $f(x)$ =

Выражение F1: Выражение F2:

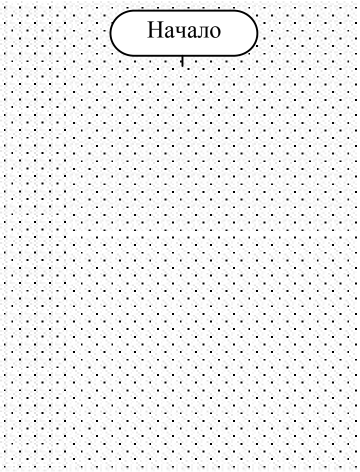
Задача по геометрии:____

Задание 1. Разработать линейную программу для вычисления значений функции $f(x)$.

```
function Fun_f(     ) {
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
}
```

Результаты пошагового выполнения программы

	x =	x =	x =
переменная			
переменная			
переменная			
функция Fun_f			



Задание 2. Вычислить значения выражений F1 и F2.

	A	B	C	D	E
1	Параметры:			Выражение:	
2	x=			F1=	
3	y=			F2=	
4	z=				

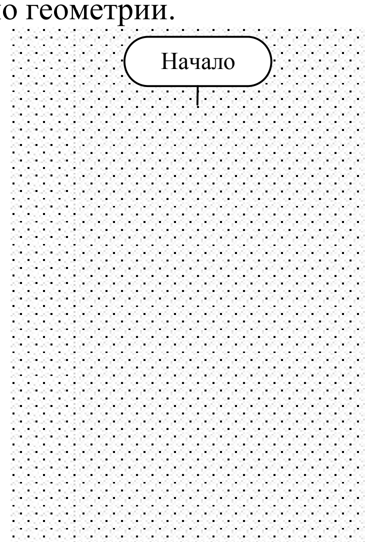
Ячейка E2:______________________________

Ячейка E3:______________________________

Задание 3. Разработать линейную программу для решения задачи по геометрии.

Рисунок

№ п/п	Дано	Результат	
		GT	MathCAD
1			
2			



Строка кода в Apps Script с методами prompt и alert:

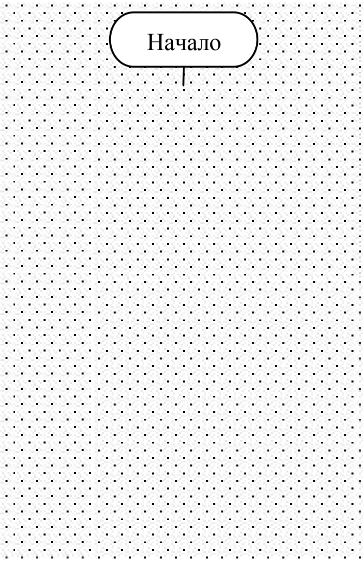
– Ввод первого параметра______________________________

– Вывод результата______________________________

Работа выполнена верно: ____________ ________________
(дата) (подпись)

Задания на защиту лабораторной работы № 1(1,2,3)

Задание 1. Разработать линейную программу для вычисления значений функции *f*(*x*):



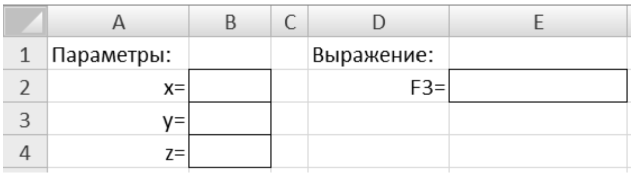
function Fun_f2(   ) {
_ _
_ _
_ _
_ _
_ _
}

Результаты пошагового выполнения программы

	x =	*x* =
переменная		
переменная		
переменная		
переменная		
функция Fun_f2		

выдано: ___________ _________________
(дата) (подпись)

Задание 2. Вычислить значение выражения F3 =

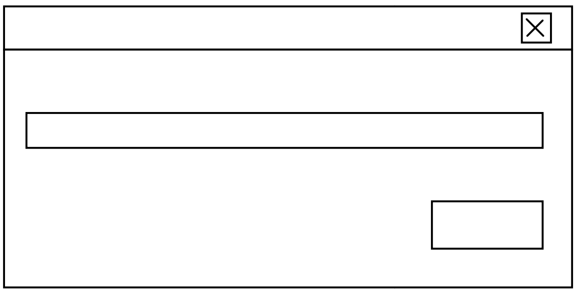


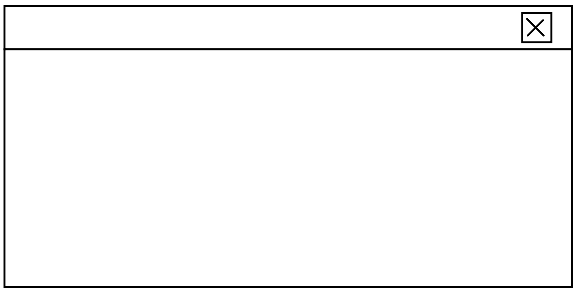
	A	B	C	D	E
1	Параметры:			Выражение:	
2	x=			F3=	
3	y=				
4	z=				

function Fun_f3(   ) {
_ _
_ _
_ _
_ _
_ _
}

выдано: ___________ _________________
(дата) (подпись)

Задание 3. Сформировать диалоговое окно на основе рисунка.





Строка кода в Apps Script:

– Рисунок слева____

____

– Рисунок справа____

____

выдано: ___________ _________________
(дата) (подпись)

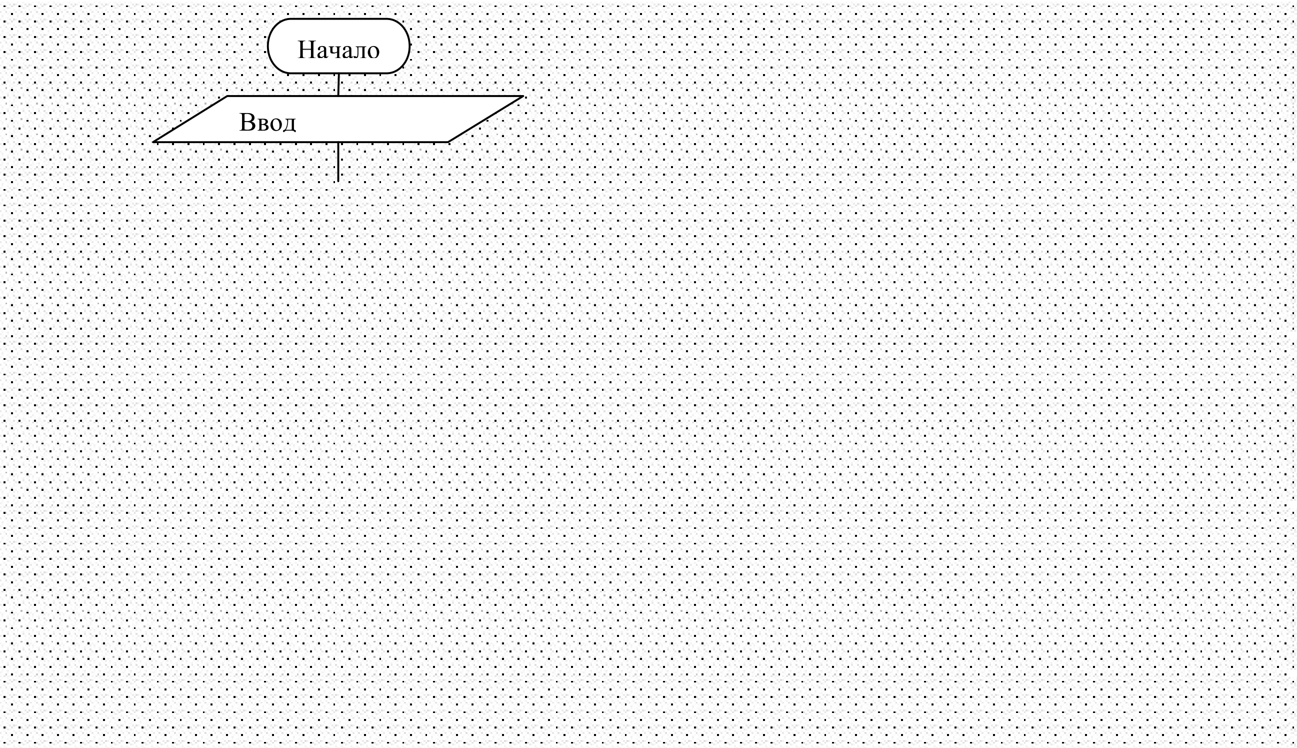
Лабораторная работа № 2(1,2)

ТЕМА. Основы работы в Apps Script. Создание пользовательских функций (с разветвляющейся структурой) в Google Таблицах + Apps Script.

Условие:

функция q(x,a,b) =

Задание 1. Составить блок-схему алгоритма, реализующего вычисление заданной функции.



Задание 2. Разработать программу в Apps Script.

```
function Fun_q(                ) {

}
```

Задание 3. Вычислить значение функции.

а)

x=	a=	b=

q = __________________(ветвь __)

б)

x=	a=	b=

q = __________________(ветвь __)

в)

x=	a=	b=

q = __________________(ветвь __)

г)

x=	a=	b=

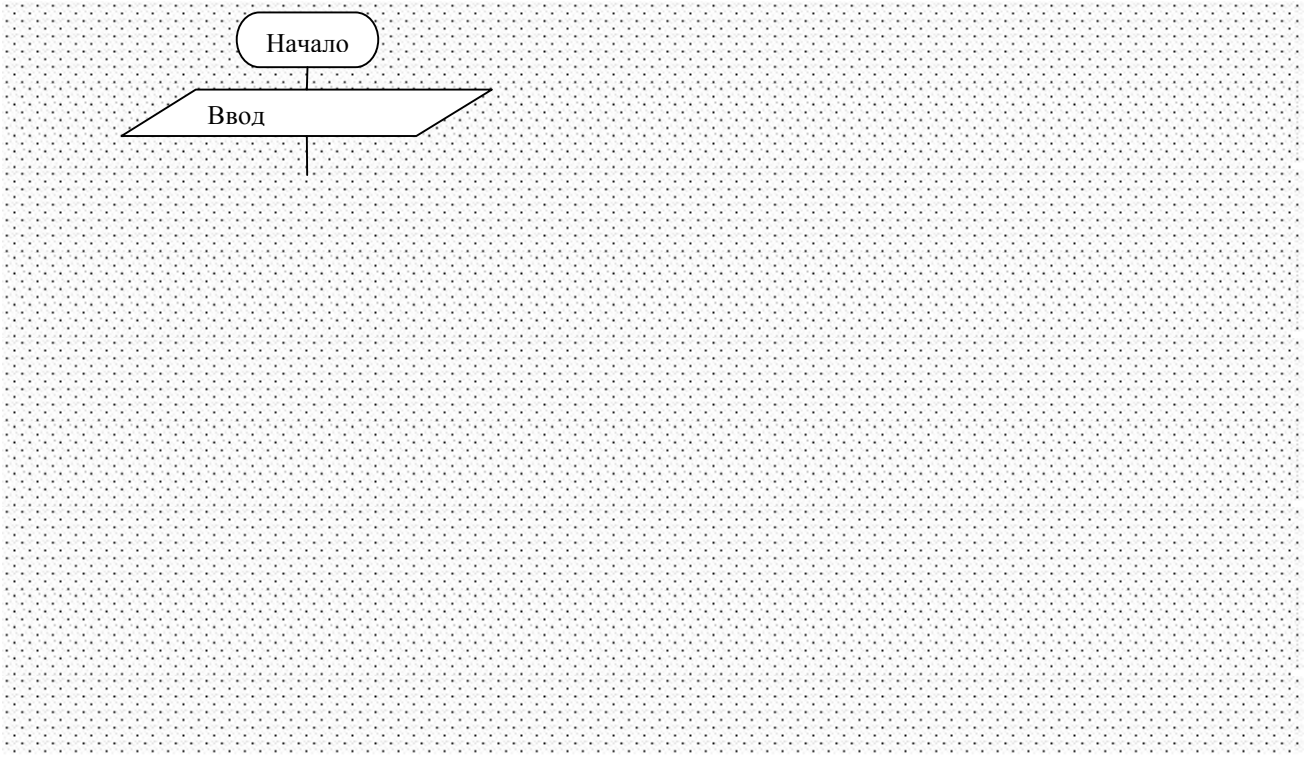
q = __________________(ветвь __)

Работа выполнена верно: ____________ ________________
(дата) (подпись)

Задания на защиту лабораторной работы № 2(1,2)

Задание 1. Составить блок-схему алгоритма, реализующего вычисление функции:

q( ) =



выдано: _____ _____
(дата) (подпись)

Задание 2. Разработать программу в Apps Script для функции:

q(x,a,b) =

а)	x =	a =	b =	q =	ветвь –
б)	x =	a =	b =	q =	ветвь –

выдано: _____ _____
(дата) (подпись)

Задание 3. Разработать программу в Apps Script для функции:

q(x,a,b) =

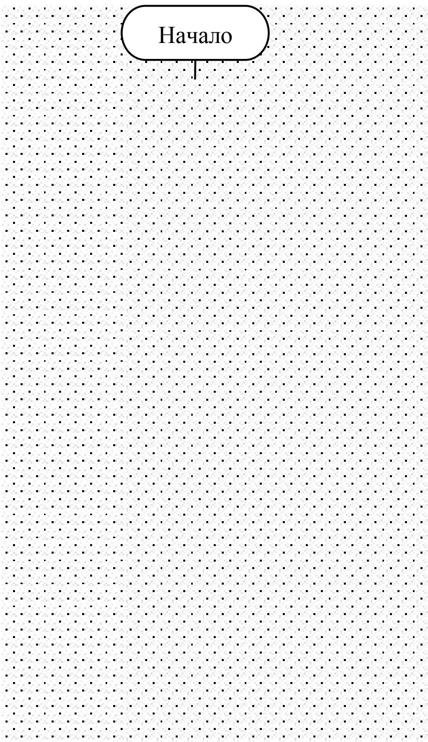
а)	x =	a =	b =	q =	ветвь –
б)	x =	a =	b =	q =	ветвь –

выдано: _____ _____
(дата) (подпись)

Лабораторная работа № 3(1,2,3)

ТЕМА. Основы работы в Apps Script. Создание пользовательских функций (с циклической структурой) в Google Таблицах + Apps Script и MathCAD.

Задание 1. Разработать циклическую программу для вычисления суммы S =

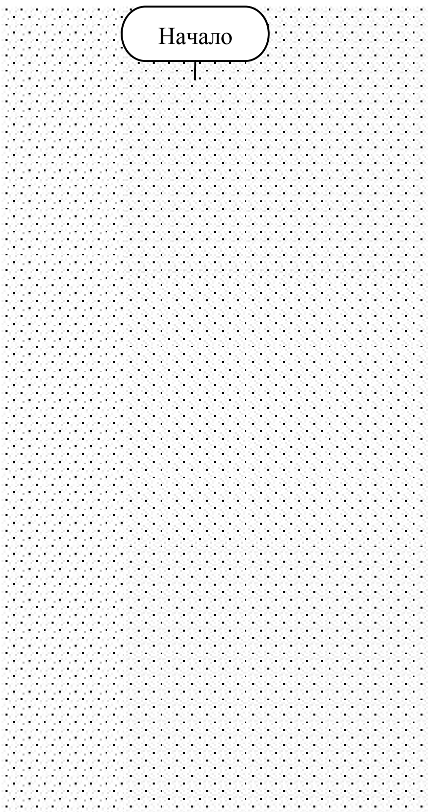



```
function Summa_S(n) {
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
}
```

Результаты выполнения программы

	A	B	C	D
1	n=	0	s=	
2	n=	1	s=	
3	n=	2	s=	
4	n=		s=	

Задание 2. Разработать программу для вычисления суммы S = с заданной точностью ε.




```
function Summa(eps) {
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
    _________________________________________
}
```

а) Результаты выполнения программы

	A	B	C	D
	eps =		S =	
	eps =		S =	

б) Результат последнего шага при ε = ________________

	A	B	C	D
3	n	элемент	условие	сумма

Формула в ячейке B__ = ____

Отрезок, содержащий значение суммы с заданной точностью, [____________;____________].

MathCAD □ – n=_____ Работа выполнена верно: ___________ ______________

(дата) (подпись)

Задания на защиту лабораторной работы № 3(1,2,3)

Задание 1. Разработать циклическую программу для вычисления суммы:

с помощью оператора цикла: □ for, □ while, □ do...while

function Summa_S(n) {

____

____

____

____

____

____

____

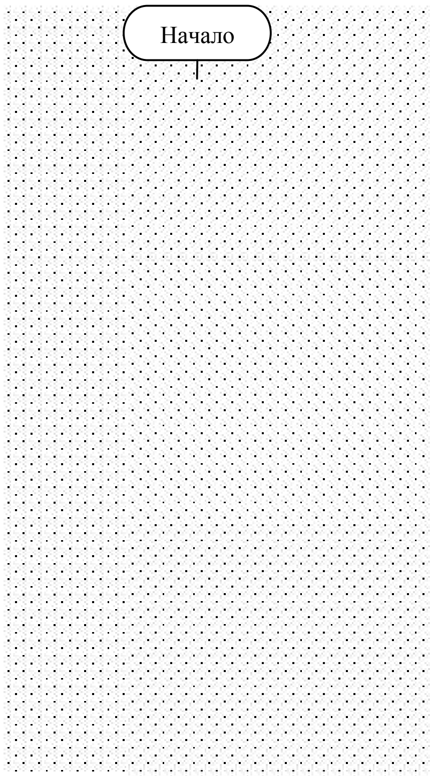
____

____

____

____

}



Результаты выполнения программы

	A	B	C	D
1	n=		s=	

выдано: __________ ________________
(дата) (подпись)

Задание 2. Определить, сколько чисел последовательности из второго задания лабораторной работы нужно взять, в случае когда точность вычисления ее суммы □ увеличится / □ уменьшится в ___ раз(а). Ответ: ____ значений.

выдано: __________ ________________
(дата) (подпись)

Задание 3. Определить, сколько чисел последовательности из второго задания лабораторной работы нужно взять, в случае когда точность вычисления ее суммы □ увеличится / □ уменьшится в ___ раз(а). Ответ: ____ значений.

выдано: __________ ________________
(дата) (подпись)

Задание 4. Определить отрезок, содержащий значение суммы последовательности из второго задания лабораторной работы, вычисленной с точностью ε = ______________.
Ответ: n = ______, S_n = _____________________, отрезок [_____________;____________].

выдано: __________ ________________
(дата) (подпись)

Задание 5. Определить отрезок, содержащий значение суммы последовательности из второго задания лабораторной работы, вычисленной с точностью ε = ______________.
Ответ: n = ______, S_n = _____________________, отрезок [_____________;____________].

выдано: __________ ________________
(дата) (подпись)

Лабораторная работа № 4(1,2,3)

ТЕМА. Исследование функции одной переменной на отрезке в Google Таблицах + Apps Script и MathCAD с использованием подпрограмм (блоков).

Условие:

$y(x)$ =____ a =____ b =____ n =____

Задание 1. В Apps Script и MathCAD разработать функцию (программный модуль) для построения таблицы значений на участке [a,b] функции $y(x)$:

```
function Fun_y(x) {
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
}
```

Задание 2. В Apps Script и MathCAD разработать функцию (программный модуль) для отделения и уточнения нулей функции $y(x)$ на участке [a,b].

Результаты первых трех шагов метода дихотомии для первого нуля функции

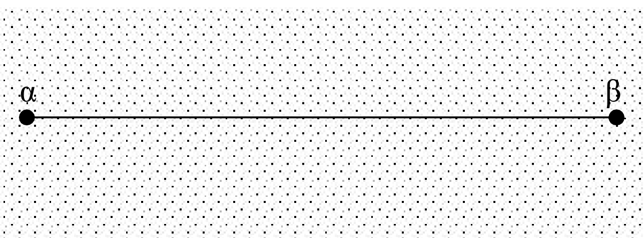
шаг 1	$\alpha =$	$c =$	$\beta =$
	$y(\alpha) =$	$y(c) =$	$y(\beta) =$
шаг 2	$\alpha =$	$c =$	$\beta =$
	$y(\alpha) =$	$y(c) =$	$y(\beta) =$
шаг 3	$\alpha =$	$c =$	$\beta =$
	$y(\alpha) =$	$y(c) =$	$y(\beta) =$

Задание 3. В Apps Script и MathCAD разработать функцию (программный модуль) для отделения и уточнения локальных экстремумов функции $y(x)$ на участке [a,b].

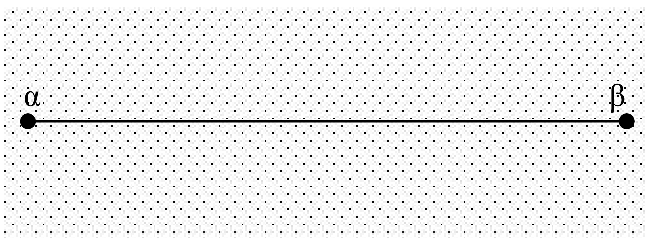
Результаты первых трех шагов метода золотого сечения для первого локального экстремума

шаг 1	$\alpha =$	$x1 =$	$x2 =$	$\beta =$
		$y(x1) =$	$y(x2) =$	
шаг 2	$\alpha =$	$x1 =$	$x2 =$	$\beta =$
		$y(x1) =$	$y(x2) =$	
шаг 3	$\alpha =$	$x1 =$	$x2 =$	$\beta =$
		$y(x1) =$	$y(x2) =$	

Задание 4. Проиллюстрировать результаты, полученные в задании 2 и 3.



метод дихотомии



метод золотого сечения

Работа выполнена верно: ___________ ______________

(дата) (подпись)

Задания на защиту лабораторной работы № 4(1,2,3)

Задание 1. Получить результаты первых трех шагов выполнения программы, реализующей уточнение ___ нуля или ___ локального экстремума функции при n =_____.

шаг 1	α =			β =
шаг 2	α =			β =
шаг 3	α =			β =

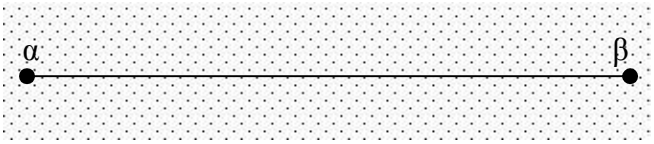
выдано: _____ _____
(дата) (подпись)

Задание 2. Получить результаты первых трех шагов выполнения программы, реализующей уточнение ___ нуля или ___ локального экстремума функции при n =_____.

шаг 1	α =			β =
шаг 2	α =			β =
шаг 3	α =			β =

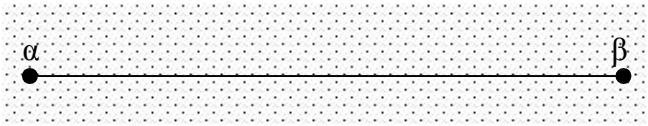
выдано: _____ _____
(дата) (подпись)

Задание 3. Проиллюстрировать результаты первых трех шагов выполнения программы, реализующей уточнение ___ нуля или ___ локального экстремума функции при n =_____, либо результат четвертого шага для уточнения □ нуля или □ локального экстремума функции из лабораторной работы.



выдано: _____ _____
(дата) (подпись)

выдано: _____ _____
(дата) (подпись)



выдано: _____ _____
(дата) (подпись)

выдано: _____ _____
(дата) (подпись)

Лабораторная работа № 5(1,2,3)

ТЕМА. Исследование функции двух переменных на прямоугольнике.

Условие:

$Z(x, y) =$

ax =____ bx =____ nx =____

ay =____ by =____ ny =____

Задание 1. В Google Таблицах + Apps Script и MathCAD разработать функции для построения таблицы значений функции $z = Z(x, y)$.

```
function Fun_Z(x, y) {
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
}
```

Z(ax,ay) = _____________; Z(bx,by) = _____________.

Задание 2. Найти глобальные экстремумы для заданной области (MathCAD).

Нач. прибл.	x	y	Z
макс			
мин			

Уточненное	x	y	Z
макс			
мин			

Задание 3. Определить области, содержащие локальные экстремумы (Google Таблицы).

макс – {____$\le x \le$____, ____$\le y \le$____}, мин – {____$\le x \le$____, ____$\le y \le$____}

Ут. значения: x_{max} = ____________; y_{max} = ____________; z_{max} = ____________;

x_{min} = ____________; y_{min} = ____________; z_{min} = ____________ .

Задание 4. Построить график линий уровня и показать все локальные экстремумы (MathCAD).

График линий уровня (MathCAD).
Размер: 9 X 9 см.
Вручную отметить:
точки локальных максимумов
символом – ●;
точки локальных минимумов
символом – ■.

	x	y	Z(x,y)
Максимум ●			
Минимум ■			

Работа выполнена верно: ___________ ______________

(дата) (подпись)

Задания на защиту лабораторной работы № 5(1,2,3)

Приклеить распечатку с графиками (MathCAD): графики сечений при фиксированном значении *x* и фиксированном значении *y*

Задание 1. Уточнить значение локального экстремума (указано на графике).

Система	Минимум			Максимум		
	x	y	Z(x,y)	x	y	Z(x,y)
Google Таблицы						
MathCAD						

выдано: _____ _____
(дата) (подпись)

Задание 2. С помощью инструмента «условное форматирование» отметить цветом _____ и _____ места локальных экстремумов функции на участке [ax, bx] = = [____,____] и [ay, by] = [____,____] при количестве разбиений по оси *x* nx = ____ и по оси *y* ny = ____. Выписать по одному результату на каждый вид экстремума.
Ответ: область _____.

Минимум			

Максимум			

выдано: _____ _____
(дата) (подпись)

Задание 3. С помощью инструмента «условное форматирование» отметить цветом _____ места локальных □ максимумов (□ минимумов) функции на участке [ax, bx] = = [____,____] и [ay, by] = [____,____] при количестве разбиений по оси *x* nx = ____ и по оси *y* ny = ____. Уточнить один из экстремумов.

Ответ: область локализации экстремума _____ ≤ x ≤ _____ ; _____ ≤ y ≤ _____;

уточненное значение x_{extr} = _____; y_{extr} = _____; $Z(x_{extr}; y_{extr})$ = _____.

выдано: _____ _____
(дата) (подпись)

Задание 4. С помощью инструмента «условное форматирование» отметить цветом _____ места локальных □ максимумов (□ минимумов) функции на участке [ax, bx] = = [____,____] и [ay, by] = [____,____] при количестве разбиений по оси *x* nx = ____ и по оси *y* ny = ____. Уточнить один из экстремумов.

Ответ: область локализации экстремума _____ ≤ x ≤ _____ ; _____ ≤ y ≤ _____;

уточненное значение x_{extr} = _____; y_{extr} = _____; $Z(x_{extr}; y_{extr})$ = _____.

выдано: _____ _____
(дата) (подпись)

Список дополнительной литературы

1. Очков, В. Ф. Mathcad 14 для студентов, инженеров и конструкторов / В. Ф. Очков. – СПб.: БХВ-Петербург, 2007. – 368 с.
2. Справка - Google Диск [Электронный ресурс]. – Режим доступа: https://support.google.com/drive.
3. Справка - Редакторы документов [Электронный ресурс]. – Режим доступа: https://support.google.com/docs/topic/9054603

Оглавление

УЧЕБНОЕ ИЗДАНИЕ

Составители:
Кофанов Валерий Анатольевич
Хомицкая Татьяна Георгиевна
Тузик Ирина Владимировна

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «Информатика»

для студентов строительных специальностей

дневной формы обучения

второй семестр

издание 4-е дополненное и переработанное

Текст печатается в авторской редакции,
орфографии и пунктуации

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

Подписано в печать 22.01.2020 г. Формат 60x84 $^{1}/_{8}$. Бумага «Performer».
Гарнитура «Times New Roman». Усл. печ. л. 4,19. Уч. изд. л. 4,50. Заказ № 50. Тираж 12 экз.
Отпечатано на ризографе учреждения образования «Брестский государственный
технический университет». 224017, г. Брест, ул. Московская, 267.