

СОДЕРЖАНИЕ

| | |
|--|----|
| Раздел I. ЯЗЫК LUA | 7 |
| Занятие 1. Введение в разработку макрокоманд | 8 |
| 1.1. Что такое макрокоманды и почему на языке Lua | 8 |
| 1.2. Обзор редактора макрокоманд в МойОфис | 10 |
| 1.3. Запуск простой макрокоманды..... | 15 |
| Контрольные вопросы и задания | 16 |
| Дополнительная информация..... | 16 |
| Занятие 2. Переменные и типы данных | 17 |
| 2.1. Понятие переменной и правила использования | 17 |
| 2.2. Типы данных в языке программирования Lua | 20 |
| 2.3. Множественное присваивание | 27 |
| 2.4. Комментарии к коду макрокоманд..... | 28 |
| Контрольные вопросы и задания | 29 |
| Занятие 3. Вычисления вLua | 30 |
| 3.1. Арифметические операторы..... | 30 |
| 3.2. Операторы сравнения..... | 32 |
| 3.3. Логические операции..... | 33 |
| 3.4. Приоритет операций вLua | 33 |
| 3.5. Конкатенация строк | 34 |
| 3.6. Оператор вычисления длины..... | 34 |
| 3.7. Функции стандартной библиотеки math..... | 34 |
| Контрольные задания | 35 |
| Занятие 4. Обработка строковых значений..... | 37 |
| 4.1. Установка значения переменной | 37 |
| 4.2. Определение длины строки | 38 |
| 4.3. Конкатенация строк с помощью оператора .. | 40 |
| 4.4. Функции преобразования значений tostring и tonumber | 41 |
| 4.5. ESC-последовательности..... | 42 |
| 4.6. Функции стандартной библиотеки Lua для работы со строками | 43 |
| 4.7. Функции модуля utf8 для работы со строками на русском языке | 44 |
| Контрольные задания | 47 |
| Занятие 5. Управляющие конструкции | 48 |
| 5.1. Условный оператор if | 49 |
| 5.2. Оператор цикла while | 52 |
| 5.3. Операторы цикла repeat-until | 53 |
| 5.4. Оператор цикла for | 55 |
| 5.5. Оператор прерывания цикла break | 58 |

| | |
|---|------------|
| 5.6. Оператор безусловного перехода goto | 58 |
| 5.7. Оператор возврата значения функции return | 59 |
| Контрольные вопросы и задания | 60 |
| Занятие 6. Функции | 61 |
| 6.1. Назначение и способы объявления пользовательских функций | 61 |
| 6.2. Аргументы функции | 62 |
| 6.3. Возвращаемые значения..... | 65 |
| 6.4. Назначение и передача функции..... | 66 |
| Контрольные вопросы и задания | 66 |
| Занятие 7. Таблицы и структуры данных..... | 67 |
| 7.1. Создание и наполнение таблиц..... | 67 |
| 7.2. Доступ к элементам таблицы | 68 |
| 7.3. Функции стандартной библиотеки для управления таблицами | 69 |
| Контрольные вопросы и задания | 70 |
| Занятие 8. Обработка ошибок | 72 |
| 8.1. Виды ошибок..... | 72 |
| 8.2. Указание текста сообщения об ошибке | 73 |
| 8.3. Перехват ошибки | 74 |
| 8.4. Возвращаемые значения..... | 76 |
| Занятие 9. Метатаблицы..... | 77 |
| 9.1. Перегрузка оператора | 77 |
| 9.2. Обращение по несуществующему ключу..... | 78 |
| 9.3. Расширение таблицы | 80 |
| Занятие 10. Отладчик макрокоманд в МойОфис | 82 |
| 10.1. Типы ошибок и понятие «отладчик» | 82 |
| 10.2. Интерфейс отладчика кода макрокоманд..... | 84 |
| 10.3. Выполнение пошаговой отладки на примере | 87 |
| 10.4. Работа со стеком вызовов | 91 |
| 10.5. Использование точки останова | 94 |
| Контрольные вопросы и задания | 96 |
| Раздел II. НАДСТРОЙКИ НА LUA В РЕДАКТОРАХ МОЙОФИС | 97 |
| Занятие 1. Введение в разработку надстроек в редакторах МойОфис..... | 98 |
| 1.1. Назначение надстройки..... | 98 |
| 1.2. Возможности..... | 99 |
| 1.3. Язык программирования Lua, окружение и модули | 100 |
| 1.4. Объектная модель МойОфис | 101 |
| 1.5. Установка надстройки Runtime для «МойОфис Текст»..... | 102 |
| 1.6. Использование надстройки для «МойОфис Текст» | 103 |
| Занятие 2. Структура модуля надстройки | 105 |
| 2.1. Разработка надстройки..... | 106 |
| 2.2. Файловая структура надстройки..... | 106 |
| 2.3. Создание надстройки..... | 108 |

| | |
|--|------------|
| 2.4. Подготовка файла регистрации | 109 |
| 2.5. Использование необязательных ключей конфигурации | 110 |
| 2.6. Подготовка файла сценария..... | 111 |
| 2.7. Подготовка файла лицензионного соглашения..... | 114 |
| 2.8. Сборка надстройки | 114 |
| 2.9. Подписание надстройки | 115 |
| 2.10. Запуск..... | 116 |
| Занятие 3. Структура модуля надстройки. Контекст | 117 |
| 3.1. Введение | 117 |
| 3.2. Контекст приложения | 118 |
| 3.3. Контекст документа | 119 |
| 3.4. Контекст фрагмента | 120 |
| Занятие 4. Абзацы и списки | 121 |
| 4.1. Введение | 122 |
| 4.2. Контекст взаимодействия и доступ к рабочему документу | 122 |
| 4.3. Определение и выделение диапазона..... | 124 |
| 4.4. Удаление текста диапазона | 125 |
| 4.5. Вставка текста в документ | 126 |
| 4.6. Форматирование текста в документе | 126 |
| 4.7. Списки..... | 128 |
| 4.8. Управление защищенным контентом..... | 129 |
| Занятие 5. Таблицы | 131 |
| 5.1. Введение | 132 |
| 5.2. Создание таблицы в текстовом документе | 132 |
| 5.3. Заполнение таблицы | 133 |
| 5.4. Добавление строк и столбцов..... | 135 |
| 5.5. Форматирование ячейки таблицы | 137 |
| 5.6. Диапазоны ячеек..... | 138 |
| 5.7. Удаление строк, столбцов и таблицы..... | 140 |
| Занятие 6. Работа с изображениями | 141 |
| 6.1. Возможности работы с изображением..... | 141 |
| 6.2. Вставка изображения | 142 |
| 6.3. Список изображений..... | 142 |
| 6.4. Список графических объектов | 143 |
| 6.5. Работа с рамкой для встроенного объекта (Frame) | 144 |
| Контрольные вопросы и задания | 145 |
| Занятие 7. Закладки в текстовом документе..... | 146 |
| 7.1. Введение..... | 146 |
| 7.2. Поиск закладки в текстовом документе | 148 |
| 7.3. Установка закладки в текстовом документе..... | 149 |
| 7.4. Удаление закладки | 151 |
| Занятие 8. Рецензирование документа | 152 |
| 8.1. Инструменты для рецензирования документа | 153 |

| | |
|---|------------|
| 8.2. Работа со списком изменений в текстовом документе..... | 154 |
| 8.3. Работа со списком комментариев в текстовом документе | 156 |
| Занятие 9. Листы | 158 |
| 9.1. Введение | 159 |
| 9.2. Переименование листа | 159 |
| 9.3. Добавление новых листов в книгу | 160 |
| 9.4. Копирование и перемещение листов..... | 161 |
| 9.5. Удаление листов | 162 |
| 9.6. Перечисление листов | 162 |
| 9.7. Выбор листа | 162 |
| 9.8. Скрытие листов..... | 162 |
| Занятие 10. Ячейки и группы ячеек | 164 |
| 10.1. Доступ к ячейке | 165 |
| 10.2. Установка значений | 165 |
| 10.3. Считывание значений..... | 167 |
| 10.4. Форматирование ячейки..... | 167 |
| 10.5. Диапазон ячеек | 170 |
| 10.6. Настройка границ..... | 170 |
| 10.7. Объединение ячеек..... | 171 |
| 10.8. Добавление строк и столбцов..... | 172 |
| 10.9. Удаление строк и столбцов..... | 173 |
| Занятие 11. Печать | 175 |
| 11.1. Печать документа..... | 175 |
| 11.2. Печать выделенного фрагмента..... | 177 |
| 11.3. Печать области | 177 |
| Занятие 12. Работа с файлами | 179 |
| Занятие 13. Разработка форм ввода данных..... | 182 |
| 13.1. Введение | 183 |
| 13.2. Создание надстройки..... | 184 |
| 13.3. Создание диалогового окна..... | 184 |
| 13.4. Список каталогов | 188 |
| 13.5. Список файлов | 191 |
| 13.6. Имя файла | 194 |
| 13.7. Выбор отображаемых файлов..... | 195 |
| 13.8. Компоновки и виджеты..... | 197 |
| Занятие 14. Работа с файлами | 200 |
| Занятие 15. Локализация надстройки | 202 |
| 15.1. Введение | 202 |
| 15.2. Словарь локализации | 204 |
| 15.3. Локализация полей таблицы регистрации..... | 204 |
| 15.4. Локализация формы ввода..... | 208 |

Раздел



ЯЗЫК LUA

Занятие 1

Введение в разработку макрокоманд

В ХОДЕ ЗАНЯТИЯ ВЫ:

- узнаете, зачем нужны макрокоманды и особенности их использования в редакторах МойОфис;
- познакомитесь с возможностями встроенного редактора макрокоманд: создавать, удалять и переименовывать макрокоманды;
- запустите первую макрокоманду на Lua в приложении МойОфис.

СОДЕРЖАНИЕ ЗАНЯТИЯ

- 1.1. Что такое макрокоманды и почему на языке Lua
 - 1.2. Обзор редактора макрокоманд в МойОфис
 - 1.3. Запуск простой макрокоманды
- Контрольные вопросы и задания
Дополнительная информация

1.1. Что такое макрокоманды и почему на языке Lua

При работе с текстом, таблицами, фигурами или иными объектами в документах пользователь офисного ПО выполняет различные операции, используя кнопки на панели инструментов, пункты основного и контекстного меню, горячие клавиши. При этом каждое действие пользователя запускает ровно одну операцию над выбранным объектом.

Чтобы ускорить работу пользователей, разработчики редакторов создали множество инструментов, позволяющих выполнить сразу несколько операций с выбранными объектами, например в текстовых редакторах с помощью стилей можно сразу изменить настройки шрифта и абзаца, во всех редакторах с помощью «кисточки» быстро перенести форматирование с одного

объекта на другой, в табличном редакторе – автоматически применять форматирование к новым строкам.

Но что делать, если одну и ту же последовательность действий необходимо выполнить на нескольких листах в табличном документе? Или даже выполнить действия с несколькими разными объектами? Для этого придуманы макрокоманды (дословно – команды, которые позволяют выполнить несколько простых команд).

Макрокоманды (макросы) представляют собой программы небольшого размера, с помощью которых пользователь автоматизирует выполнение продолжительных или часто повторяющихся операций внутри документов.

С помощью макрокоманд можно решать следующие задачи работы с документами и данными в них:

- автоматизировать форматирование текста;
- проверить корректность данных;
- отредактировать документ, заменить одни данные на другие.

Макросы фиксируются в виде программ на том языке программирования, который поддерживается в редакторах. Начинающие пользователи предполагают работу с функцией «Запись макрокоманды», которая формирует код макроса, фиксируя действия пользователя. Опытные пользователи создают макросы «с нуля» во встроенной среде разработки, что позволяет использовать все возможности языка программирования, в том числе реализовать сценарии обработки данных, недоступные в интерфейсе программы.

Для того чтобы пользователи могли самостоятельно разобраться в работе макросов, их создают и редактируют с использованием скриптовых языков программирования, для которых характерны следующие черты:

- простой синтаксис, что обеспечивает быстрый старт;
- хранение программ в виде текста, что упрощает изучение, отладку и контроль работы макросов;
- выполнение программ с использованием интерпретатора (специальной программы, переводящей код в действия только при непосредственном вызове), что позволяет выполнять макросы по отдельности.

В редакторах МойОфис разработка макрокоманд ведется на языке программирования Lua, который не только является скриптовым языком программирования, но и обеспечивает дополнительные возможности:

- макросы одинаково работают в настольных редакторах МойОфис для ОС Linux, для ОС Windows и в веб-редакторах;
- не требует покупки дополнительного ПО или специальных лицензий для работы;
- допускает подключение внешних модулей для расширения функциональных возможностей – так реализован доступ к объектам в документе.

Для создания макрокоманд в редакторах МойОфис можно использовать встроенные средства программирования или специальную функцию «Запись макрокоманды», которая фиксирует действия пользователя и сохраняет в виде программы.

СПРАВОЧНО Язык Lua разработан в 1993 году как средство программирования для пользователей, не являющихся профессиональными программистами. Малый размер интерпретатора, относительно высокая скорость исполнения и легкая расширяемость позволили этому языку найти применение в сфере разработки компьютерных игр (в 2003 году Lua был признан самым популярным скриптовым языком для разработки игр, по версии сообщества GameDev.net). Также Lua активно используется для написания расширений и плагинов к программным продуктам, при создании пользовательских интерфейсов и в конфигурационных файлах.

Таким образом, язык Lua – интерпретируемый язык, созданный специально для непрограммистов, прост в освоении и платформонезависимый, а также давно зарекомендовал себя на мировом рынке как эффективный скриптовый язык.

1.2. Обзор редактора макрокоманд в МойОфис

Диалоговое окно **Редактирование макроса** открывает доступ к редактору макрокоманд, с помощью которого пользователь создает и запускает макрокоманду.

Запустите приложение «МойОфис Текст», затем в командном меню выберите **Макрокоманды – Редактор макрокоманд** (рис. 1.1).

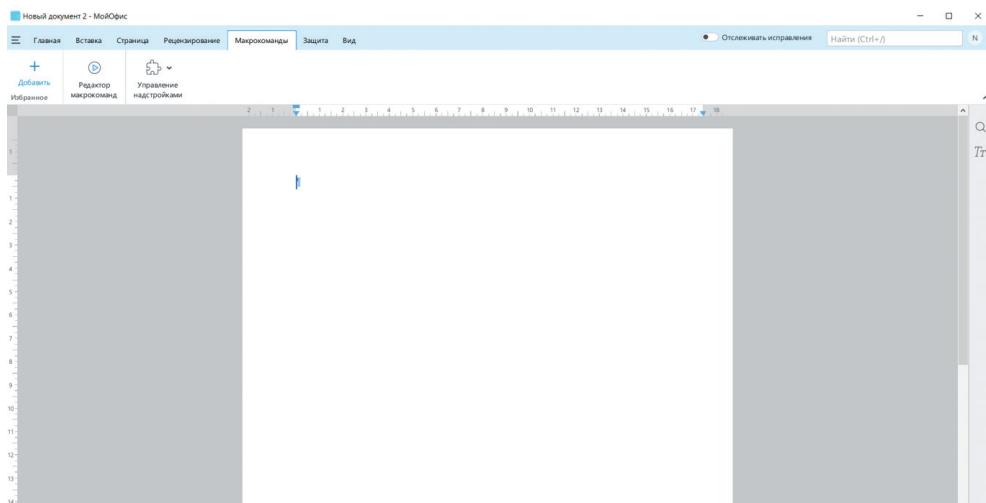


Рис. 1.1 ❖ Окно приложения «МойОфис Текст»

Окно редактора макрокоманд является модальным по отношению к главному окну приложения (рис. 1.2). Это означает, что пользователь не может просматривать или редактировать текст документа, пока открыто окно редактора макрокоманд.

Чтобы закрыть окно редактора макрокоманд, необходимо нажать на крестик в строке заголовка окна. Специальной кнопки для сохранения введенной макрокоманды нет, при закрытии окна сохранение происходит автоматически.

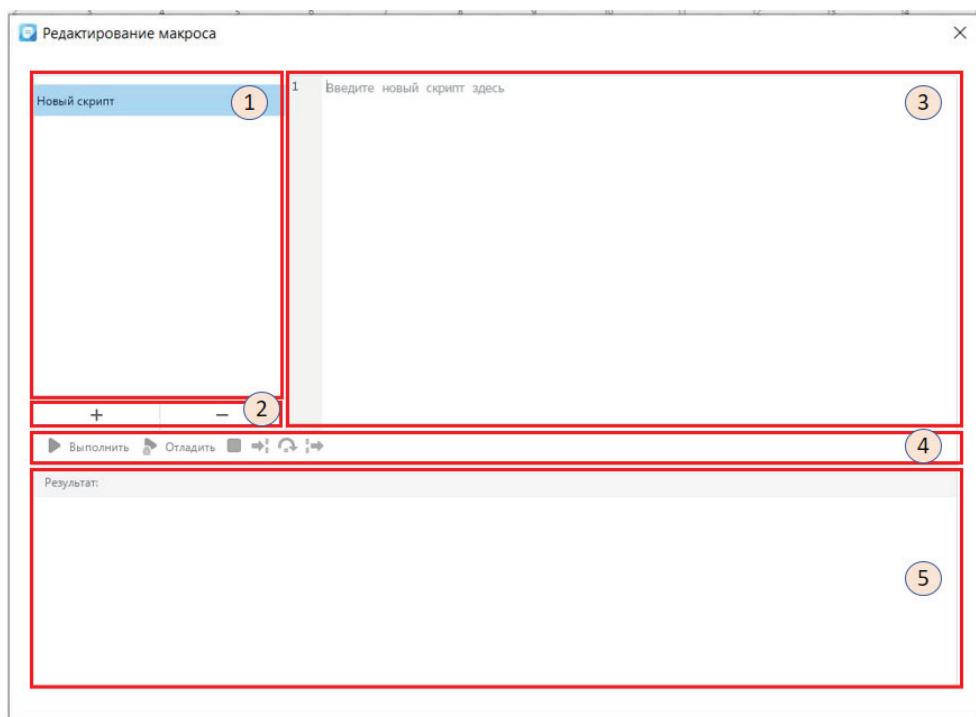


Рис. 1.2 ♦ Окно редактора макрокоманд

Окно редактора макрокоманд состоит из областей:

- 1) перечень созданных в документе макрокоманд;
- 2) кнопки для создания и удаления макрокоманд;
- 3) область ввода текста макрокоманд;
- 4) кнопки выполнения и отладки макрокоманд;
- 5) область вывода результата выполнения макрокоманд, а также отображения информации в процессе отладки макрокоманд.

Рассмотрим эти области подробнее.

1.2.1. Перечень макрокоманд в документе

Перечень представляет собой список макрокоманд в открытом документе. Вновь созданный документ не содержит макрокоманд. Если в перечне содержится несколько макрокоманд, переход между ними выполняется с помощью щелчка мыши по названию макроса, активная макрокоманда в перечне подсвечивается.

1.2.2. Кнопки для создания и удаления макрокоманд

Для добавления новой макрокоманды необходимо нажать кнопку с изображением знака «+». После ее нажатия в перечне появится новый пункт под названием **Без имени**. В области ввода текста макрокоманды (область 3) появилась надпись: «*Введите новый скрипт здесь*».

Чтобы задать название новой макрокоманде, достаточно начать печатать его с клавиатуры и нажать клавишу **Enter**. Для изменения названия макрокоманды в перечне макрокоманд необходимо дважды щелкнуть левой кнопкой мыши по текущему названию и ввести новое имя. Возможно задавать имена макрокоманд как на русском, так и на английском языке (латиница).

Чтобы удалить макрокоманду, необходимо выделить ее и нажать кнопку с изображением знака «–» внизу списка макрокоманд.

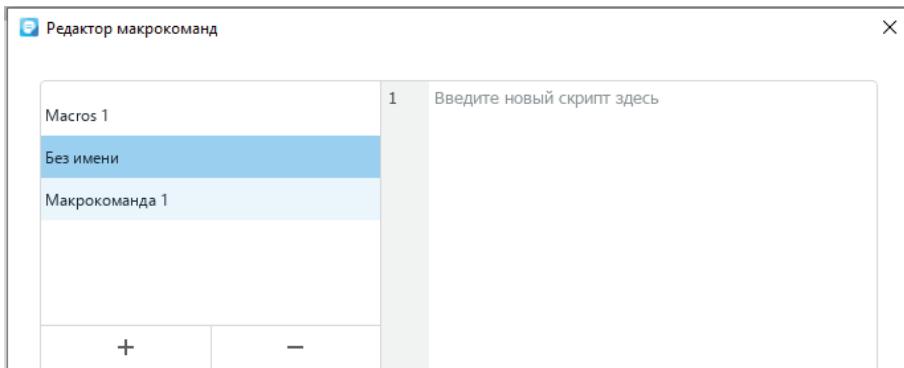


Рис. 1.3 ♦ Панель со списком макрокоманд в окне редактора макрокоманд

1.2.3. Область ввода текста макрокоманды

Область ввода представляет собой небольшой редактор текста и предназначена для написания исходного кода макрокоманды на языке программирования Lua.

Редактор поддерживает синтаксис языка Lua. Обратите внимание, что ключевые слова языка отображаются синим цветом, комментарии – зеленым и т. д.

```

1 Months = { "Январь", "Февраль",
2     "Март", "Апрель", "Май",
3     "Июнь", "Июль", "Август",
4     "Сентябрь", "Октябрь", "Ноябрь",
5     "Декабрь" }
6
7 myComputer = {cpu = "Intel Corei7", ram = 16, hdd = 2}
8
9 -- объявляем пустую таблицу
10 myNumbers = {}
11 -- и заполняем ее значениями от 1 до 10
12 for i = 1, 10 do
13     myNumbers[i] = i
14     print(myNumbers[i])
15 end
16
17 -- то же самое можно сделать с помощью списка значений
18 myNumbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
19 for i = 1, 10 do
20     print(myNumbers[i])
21 end
22
23 -- матрица 2x2
24 myMatrix = {}
25 for col = 1,2 do
26     myMatrix[col] = {}

```

Рис. 1.4 ♦ Область ввода макрокоманды
в окне редактора макрокоманд

Область ввода поддерживает операции копировать (**Ctrl+C**), вырезать (**Ctrl+X**), вставить (**Ctrl+V**) через буфер обмена и операцию отмены последнего действия (**Ctrl+Z**).

Для ввода кода макрокоманды необходимо создать и назвать ее, а затем набрать код макроса в области 3. Чтобы редактировать макрокоманду, необходимо выбрать ее в перечне макрокоманд и внести изменения в ее текст в области 3.

1.2.4. Кнопки выполнения и отладки макрокоманд

Кнопки для выполнения и отладки становятся активными, изменяя цвет, после ввода текста макрокоманд в области 3.

Чтобы выполнить код макрокоманды, необходимо нажать кнопку **Выполнить** (► Выполнить). Результат или статус выполнения макрокоманды отображается в области вывода результата выполнения макрокоманд (область 5 на рис. 1.2).

Макрокоманды также можно запускать, не открывая редактор. Для этого необходимо открыть список макросов, нажав на иконку **Макрокоманды** на правой боковой панели. Далее выбрать из списка макрос и нажать кнопку **Выполнить** (см. рис. 1.5). Произойдет выполнение макроса без открытия редактора. Появится всплывающее сообщение «Макрокоманда выполнена».

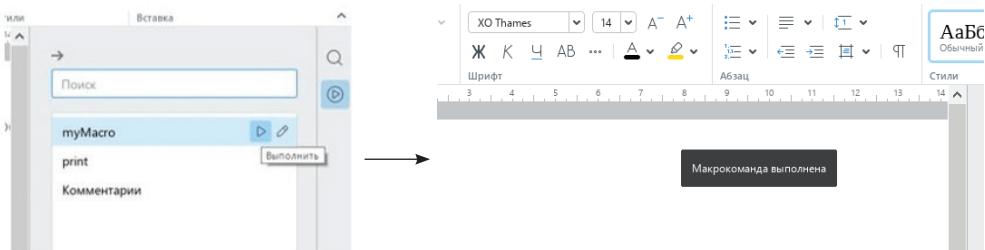


Рис. 1.5 ♦ Запуск макроса из списка на правой панели редактора

При написании кода могут возникать ошибки. Помочь их выявить и локализовать может процесс отладки кода. Подробнее процесс отладки будет рассмотрен в занятии «Отладка кода макрокоманд».

1.2.5. Область вывода результата исполнения макрокоманды и процесса отладки

Область вывода результата также называется **консоль**. В консоль выводятся любые значения, которые пользователь может напечатать с помощью функции `print()`.

После нажатия кнопки **Выполнить** начинается выполнение исходного кода макрокоманды. Когда в исходном коде встречается вызов функции `print`, происходит немедленный вывод на печать значения функции. Например, выражение

```
print("МойОфис!")
```

приведет к тому, что в консоли будет напечатан текст

МойОфис!

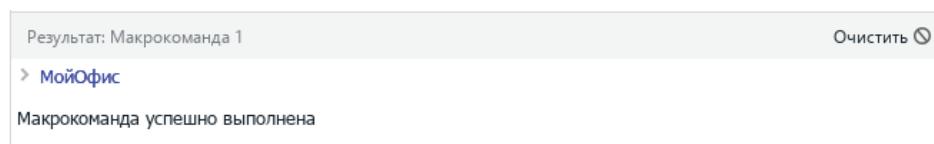


Рис. 1.6 ♦ Консоль редактора макрокоманд

Если в консоли есть результат выполнения макроса, то в правом верхнем углу появляется кнопка **Очистить**. Нажатие на эту кнопку очищает консоль (удаляет ее содержимое), не затрагивая код макрокоманды.

Также в область результата выводится информация о выполнении макрокоманды. Если макрос выполнен успешно, то в консоль будет выведена информация «*Макрос выполнен успешно*». Если предполагается, что код мак-

роса вносит изменения в документ без вывода в консоль, то результаты макроса можно будет увидеть в самом файле после закрытия окна редактирования.

При выполнении процесса отладки в области вывода результатов отображаются дополнительные поля, содержащие информацию о процессе выполнения команд кода. О них подробнее будет рассказано в следующих занятиях.

1.3. Запуск простой макрокоманды

Теперь давайте создадим первую макрокоманду и запустим ее. В окне редактирования макросов нажмите на кнопку «+» в перечне макрокоманд. В качестве имени макрокоманды укажем слово «Привет».

В области ввода текста макрокоманды введем следующий текст:

```
print("Привет, МойОфис! ")
```

Нажмем кнопку **Выполнить** для запуска макрокоманды. В консоли будет напечатан текст

Привет, МойОфис!

Закроем окно редактора макрокоманд, нажав крестик в правом верхнем углу окна.

Сохраним изменения (мы ведь создали макрокоманду!) в открытом документе нажатием кнопки **Сохранить** на панели команд или с помощью комбинации клавиш **Ctrl+S**. В качестве имени документа укажем «Макрокоманда-1».

Мы создали и выполнили первую макрокоманду, используя стандартную функцию `print()` языка программирования Lua. Функция `print()` предназначена для печати значения, переданного в качестве аргумента функции (между круглых скобок «()»). Аргументом команды может быть как текст, так и числа, переменные или целые выражения. Текст в качестве аргумента всегда пишется в кавычках, числа и переменные – без кавычек.

Пример использования команды `print()` с различными аргументами:

| |
|--|
| Ввод в окне редактора макрокоманд |
| <pre>print(8) print("Фрукты: Яблоки, Груши, Апельсины") print(2 + 2 + 2)</pre> |
| Результат в консоли после нажатия кнопки Выполнить |
| <pre>> 8 > Фрукты: Яблоки, Груши, Апельсины > 6</pre> |
| |

На следующем занятии будет рассмотрено более подробно, что называют переменными и с какими типами данных работает язык Lua. Для закрепления пройденного материала ответьте на вопросы и выполните практические задания в конце данного занятия.

Контрольные вопросы и задания

1. В каких ситуациях рационально использовать макрокоманды?
2. Чем обоснован выбор языка программирования Lua для написания макросов?
3. Как запустить окно редактирования макросов в редакторах МойОфис?
4. Наберите коды макрокоманд из примеров в редакторе макрокоманд. Попробуйте ввести свои данные в код и выполнить его.

Дополнительная информация

Для начала работы с редакторами МойОфис и созданием макрокоманд в них вы можете установить бесплатную версию «МойОфис Стандартный. Домашняя версия» (myoffice.ru/products/standard-home-edition) с сайта компании-разработчика. Все задания, связанные с изучением языка Lua и написанием макрокоманд, можно выполнить, используя данные редакторы.

Если вы хотите больше узнать о работе в редакторах, то посетите бесплатные вебинары или посмотрите их в записи (myofficehub.ru/events/), а также изучите обучающие материалы (myofficehub.ru/materials/): статьи, учебные пособия, видеоуроки и др.

Занятие 2

Переменные и типы данных

В ХОДЕ ЗАНЯТИЯ ВЫ:

- узнаете, какие бывают переменные и особенности их именования;
- узнаете, какие типы данных существуют в языке программирования Lua;
- научитесь определять тип данных с помощью встроенной функции;
- познакомитесь с видами комментариев и ситуациями их использования.

СОДЕРЖАНИЕ ЗАНЯТИЯ

- 2.1. Понятие переменной и правила использования
 - 2.2. Типы данных в языке программирования Lua
 - 2.3. Множественное присваивание
 - 2.4. Комментарии к коду макрокоманд
- Контрольные вопросы и задания

Здесь мы кратко познакомимся с особенностями типов данных в Lua, в следующих занятиях более подробно разберем каждый из них.

2.1. Понятие переменной и правила использования

Словом «переменная» называют поименованную область памяти компьютера. Адрес этой области используется для получения доступа к данным. Данные, которые находятся в переменной (то есть данные, расположенные по этому адресу в памяти), называются **значением переменной**.

В ходе выполнения программы, при выполнении вычислений, данные могут изменяться, отсюда название «переменная» (variable).

В языке программирования Lua имена переменных могут содержать буквы, цифры и знаки подчеркивания. Имя переменной обязательно должно начинаться с буквы.

Следующие имена переменных являются допустимыми:

a, id, Name, tab_name, column_b12.

Использование следующих имен переменных недопустимо:

12b, @_count, мойофис.

Кроме того, следующие слова зарезервированы и не могут быть использованы в качестве имен переменных в языке Lua:

| | | | | | |
|-------|-------|----------|-------|--------|--------|
| and | break | do | else | elseif | end |
| false | for | function | goto | if | in |
| local | nil | not | or | repeat | return |
| then | true | until | while | | |

Все имена, начинающиеся с символа подчеркивания, за которым идут заглавные буквы (например, _VERSION), также являются зарезервированными.

Хорошей практикой считается выбирать короткие названия переменных, которые при этом понятны не только автору кода. Например, total_score – итоговый счет или max_width – максимальная ширина.

Язык Lua чувствителен к регистру символов, поэтому abc, Abc, ABC являются различными именами.

Примеры использования переменных и присвоения значений:

| Ввод в окне редактора макрокоманд |
|--|
| <pre>-- переменная с именем name содержит строковое значение -- (текст) "Сергей" name = "Сергей" -- переменная my_number содержит целое число 10 my_number = 10 -- распечатаем значение my_number print(my_number) -- переменная price_total содержит число с плавающей точкой 124.89 price_total = 124.89 -- переменная print_msg содержит функцию с одноименным названием print_msg = function(msg) print(msg) end -- эту функцию можно вызвать так print_msg("МойОфис!")</pre> |
| Результат в консоли после нажатия кнопки Выполнить |
| <pre>> 10 > МойОфис!</pre> |

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru