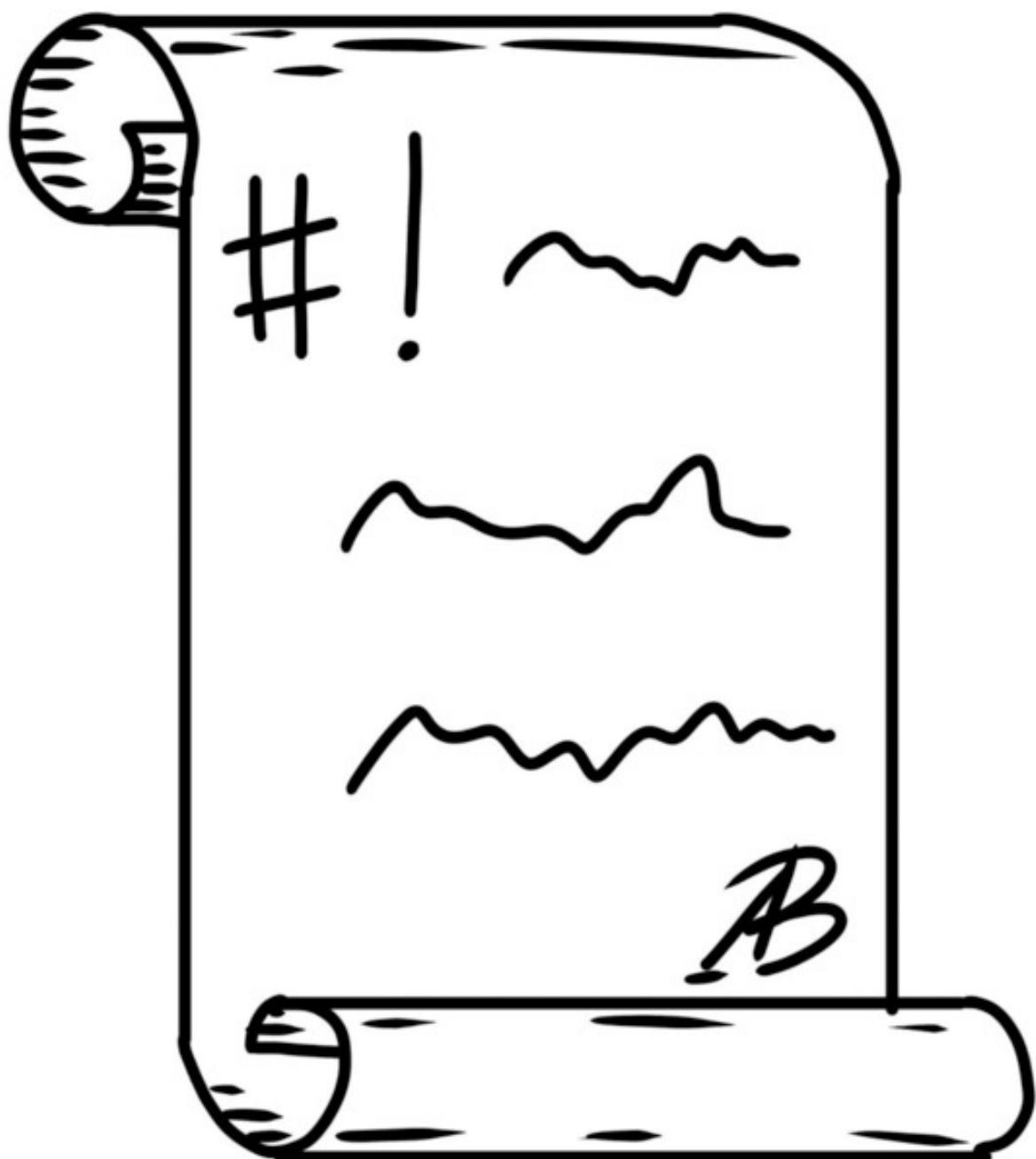


Валентин Арьков  
*Скрипты. Сценарии  
автоматизации*

Учебное пособие



Валентин Юльевич Арьков

# **Скрипты. Сценарии автоматизации.**

**Учебное пособие**

Шрифты предоставлены компанией «ПараТайп»

© Валентин Юльевич Арьков, 2025

«Скрипты-сценарии автоматизации» – это практическое руководство по автоматизации рутинных задач в популярных операционных системах. Через создание скриптов и пакетных файлов можно освоить основы программирования. Скрипты работают как в командной строке, так и в графическом интерфейсе пользователя. Скрипты можно применить внутри офисных пакетов. Скриптовые языки – это новый взгляд на мир программирования.



Создано в интеллектуальной издательской системе Ridero

# ОГЛАВЛЕНИЕ

[Скрипты. Сценарии автоматизации.](#)

[Введение](#)

[База знаний](#)

[Чат-боты](#)

[Странные названия](#)

[Среда разработки](#)

[1. Пакетные файлы BAT/CMD](#)

[CMD / CLI](#)

[Batch file](#)

[Интерпретатор](#)

[Примеры пакетных файлов](#)

[Параметры командной строки](#)

[Конвейер](#)

[Итоги](#)

[2. Скрипты VBS](#)

[MsgBox – вывод сообщения](#)

[InputBox – ввод данных](#)

[MsgBox – Параметры окна](#)

[Создаём текстовый файл](#)

[Добавим творчества](#)

[Прекращение поддержки](#)

[Итоги](#)

[3. Макросы VBA](#)

[Первый макрос](#)

[Режим Разработчика](#)

[LibreOffice + VBA](#)

[Итоги](#)

[4. Системные вызовы API](#)

[Hello, C \(часть 1\)](#)

[Компилятор](#)

[Hello, C \(часть 2\)](#)  
[Системные вызовы](#)  
[Скрипт и программа](#)  
[Итоги](#)

## [5. PowerShell](#)

[С чего начать?](#)  
[Как сменить каталог?](#)  
[Поговорим о политике](#)  
[Табуляция ускоряет](#)  
[Кто скрывается за псевдонимом](#)  
[Объекты или текст](#)  
[Список файлов](#)  
[Работа с файлом](#)  
[Кодирование свойств](#)  
[Перенаправление вывода](#)  
[Отправка в файл](#)  
[T-образный объект](#)  
[Скрипты PowerShell](#)  
[Реестр Windows Registry](#)  
[Итоги раздела](#)  
[Весёлые картинки](#)  
[Заключение](#)

# ВВЕДЕНИЕ

Усложнять – просто,

упрощать – сложно.

(Закон Мейера)

Сегодня на повестке дня пара вопросов. Вопрос первый: Не пора ли нам уже заняться программированием? Вопрос второй: Зачем? Ибо программы пишут с разными целями.

Одно из направлений в программировании – это автоматизация привычных, повторяющихся, однообразных действий. Их обычно называют словом «рутина». Между прочим, слово *routine* имеется в немецком, французском и английском языках и буквально означает «путь», «маршрут», «проторенная дорога», а также «привычная процедура». В компьютерных технологиях английское слово *routine* часто используют именно в этом, последнем значении – «процедура», «программа». Часть программы – это «подпрограмма» – *subroutine*. Здесь приставка *sub-* буквально означает «под-». Получается *sub + routine* = под + программа.

Автоматизация нужна на уровне использования операционной системы. Здесь появляются шаблоны действий, сценарии, последовательность операций. Их оформляют в виде простых программ, которые называют «скрипты». Буквально английское слово *script* означает «сценарий», «план действий», а также «рукопись» и даже просто «рукописный шрифт». Такие сценарии бывают полезны для записи последовательности операций, да

и просто для хранения длинных команд с большим количеством параметров. В качестве упражнения выясните, что может означать английское слово SCRIPT и каково его происхождение. Нет ли тут родственных связей с тем, как скрипит перо по бумаге и как скребут кошки на душе; -)

Общее знакомство со скриптами – это часть компьютерной грамотности – и для пользователя, и для программиста, и для системного администратора (сисадмина). Вот этим видом программирования мы сейчас займемся.

Попутно мы знакомимся с различными видами интерфейса пользователя – User Interface (UI). Интерфейс есть у любой программы. Разработчик-программист создает этот интерфейс. Пользователь работает, общается с программой через интерфейс. Разные виды интерфейса открывают разные возможности. Они нужны для решения разных задач. Нам нужно представлять общую картину, чтобы грамотно этим инструментом пользоваться.

## БАЗА ЗНАНИЙ

При изучении новых технологий мы будем периодически обращаться к такому ресурсу, как Википедия. Это более-менее стабильный ресурс, не самый лучший, не самый точный и не самый подробный. Относитесь к нему, как к примеру организации «базы знаний» по принципу Crowd Source. И в этом примере нам интересна сама по себе технология информационного наполнения и организация поиска материалов.

Главная особенность (достоинство и недостаток одновременно) в том, что практически любой желающий

может разместить здесь свой материал или внести исправления в существующую статью. Изменения будут опубликованы после недолгой проверки (модерации). Естественно, это влияет на скорость создания, а также качество материалов. Зачастую здесь встречаются фрагменты текста из учебников. Конечно же, никто не мешает вам найти более надежные источники по любой теме.

**Задание.** Выясните, что означает название «Википедия», какая технология использована для ее построения и где еще такая технология применяется в настоящее время.

**Задание.** Просмотрите на Википедии статьи Краудсорсинг и Crowdsourcing. Выпишите названия некоторых примеров таких проектов.

## **ЧАТ-БОТЫ**

При работе с новым материалом вам придется искать ответы на вопросы и решения для проблемных ситуаций. Традиционно мы ищем ответы с помощью поисковых машин, таких как Yandex или Google. В последнее время нам приходят на помощь интеллектуальные диалоговые сервисы – Chat bots –

**Чат-боты.** Слово Chat означает «беседа, разговор, болтовня», а слово Bot – это окончание слова Robot. Имеется в виде программы, которая работает автоматически, без участия человека, как робот – то есть «программный робот». Современные чат-боты – это на самом деле удобный программный интерфейс к системам искусственного интеллекта. Прежде всего,

к нейросетям. Существуют и другие способы работы с интеллектуальными системами, мы будем рассматривать их с последующих разделах.

В наших заданиях мы рекомендуем ориентироваться на отечественный сервис ГигаЧат – GigaChat. Можно обращаться и к любым другим инструментам. В предыдущих работах мы уже разбирали, как грамотно составлять качественные запросы к нейросетям. Напомним, что при работе с нейросетями можно получить неграмотные или неточные ответы, которые звучат очень убедительно и выглядят правдоподобно. Такая ситуация возникает при нехватке информации и называется «галлюцинация». Поэтому ответы нейросети нужно воспринимать критически и обязательно проверять их правильность. В нашем случае проверка простая: будет ли работать сгенерированная программа.

## **СТРАННЫЕ НАЗВАНИЯ**

При выполнении заданий мы обращаем внимание на названия фирм, технологий и программ. Каждое название имеет свою историю и несет какой-то смысл. Выясняется, что чаще всего эти названия появляются совершенно случайным образом. Конечно, иногда программисты даже пытаются что-то сообщить этим названием.

Известны случаи, когда названия серьезных программных продуктов придумали дети. Так можно легко получить слово, которое сможет выговорить любой ребенок. Слово простое, приятное для слуха и абсолютно бессмысленное.

Со временем люди привыкают к этим названиям, и менять их уже никто не будет. Тем более, что эти

названия уже зарегистрированы и запатентованы.

Одна из причин в том, что компьютерные технологии развиваются очень быстро. Даже слишком быстро. Быстрее, чем любая другая отрасль человеческой деятельности. И с этим приходится считаться.

## **СРЕДА РАЗРАБОТКИ**

Чтобы написать и запустить компьютерную программу, нужно выполнить хотя бы эти два действия – (1) написать и (2) запустить – звучит вроде бы наивно, просто и очевидно. И мы начинаем с «самого простого». Постепенно эту схему можно усложнить до невозможности – если, конечно, захочется.

Работать будем в широко распространенной операционной системе (ОС) Windows. И это просто примеры, иллюстрации, демонстрации. Здесь мы обсуждаем технологии разработки программ и не слишком погружаемся в тонкости языка программирования. Составление программ на разных языках программирования и в разных ОС не слишком сильно различаются.

Итак, в простейшем случае нам понадобятся два отдельных инструмента: для редактирования текстового файла и для запуска программы.

Для серьёзной работы над программами есть серьёзные инструменты. И называются они так: интегрированная среда разработки – IDE – Integrated Development Environment.

Задание. Посмотрите в Википедии статьи Интегрированная среда разработки и Integrated development environment. Выясните, какие инструменты

входят в состав современных IDE. Уточните список основных компонентов среды разработки с помощью Гигачата.

# **1. ПАКЕТНЫЕ ФАЙЛЫ BAT/CMD**

Только у нас есть  
пакет для хранения пакетов...  
(народная мудрость)

Наша первая программа – это пакетный файл Windows. Обычно ему дают расширение \*.BAT – от английского слова BATCH – «пакет». Называют такой файл «пакетным» потому, что в нем несколько команд объединяют в один «пакет» заданий. Если работать в консоли, то мы каждый раз вводим по одной команде и запускаем её на выполнение. Внутри пакетного файла можно написать несколько команд, и они выполняются как одно большое задание. Так получается «пакет» из нескольких команд.

Пакетные файлы были разработаны для ОС DOS, а затем они перекочевали в Windows. В основном, для совместимости с прежними разработками.

В других ситуациях и в других ОС такие файлы могут называть «сценарии», «скрипты» и так далее. Все зависит от фантазии и эрудиции разработчика. При этом слово «пакет» у них может иметь совсем другой смысл. Тем более, что английские слова BATCH и PACKET переводятся как «пакет». Будьте бдительны, имея дело с программистами и плодами их трудов.

**Задание.** Просмотрите в Википедии статьи Скриптовый язык и Scripting language. Обратите внимание на первоначальное предназначение скриптовых языков и список примеров таких языков.

## CMD / CLI

Запустим консоль Windows. Это «чёрное окно с белыми буквами». Один из первых интерфейсов пользователя. Другие названия: терминал, командное окно, командная строка, интерпретатор командной строки, Command Prompt, Command Processor.

Самый быстрый способ запустить командное окно – нажать комбинацию клавиш [Win + R], ввести cmd и нажать [Enter]. Здесь R – первая буква слова RUN – «запустить, выполнить». CMD – это название программы, которая обслуживает командную строку – Command Line. В последнее время стало модно называть её CLI – Command Line Interface – интерфейс командной строки.

Здесь вводим команды с помощью клавиатуры, нажимаем ENTER и сразу получаем ответ операционной системы, см. рис.

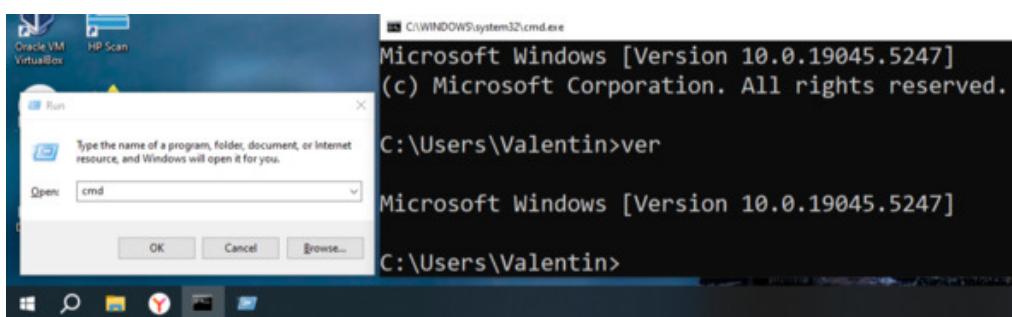


Рис. Командное окно Windows

Например, команда VER позволяет нам узнать версию операционной системы (ОС) и даже номер сборки. Есть даже специальная команда HELP, которая

выводит список основных команд ОС. В командном окне Windows можно использовать и большие, и маленькие буквы (заглавные и строчные) – система их не различает. Видимо, это наследство старой системы DOS. В других системах регистр букв нужно строго соблюдать при вводе команд.

Задание. Заучите разные названия командного окна – наизусть, как стихи. Вы должны быть готовы встретить любое из этих названий и понять, о чем идет речь.

Задание. Запустите командное окно и выведите на экран список доступных команд. Выясните, как вывести на экран список задач (то есть процессов, или запущенных программ) и как остановить выполнение выбранного процесса. Теперь введите команду, чтобы остановить выполнение процесса, который обслуживает наше текущее командное окно.

Задание. Просмотрите на Википедии статью cmd.exe – русский и английский вариант. Ознакомьтесь с примерами команд. Выясните, какой современный инструмент пришел на смену традиционному командному окну.

## BATCH FILE

Переходим к составлению и запуску пакетного файла. Начинаем с текстового редактора. Запускаем встроенный редактор Notepad, он же Блокнот. Найти его можно в меню Пуск, или через поиск, или ввести вручную: [Windows + R] – notepad. Здесь буква R по-

прежнему намекает на английское слово Run – Запустить программу.

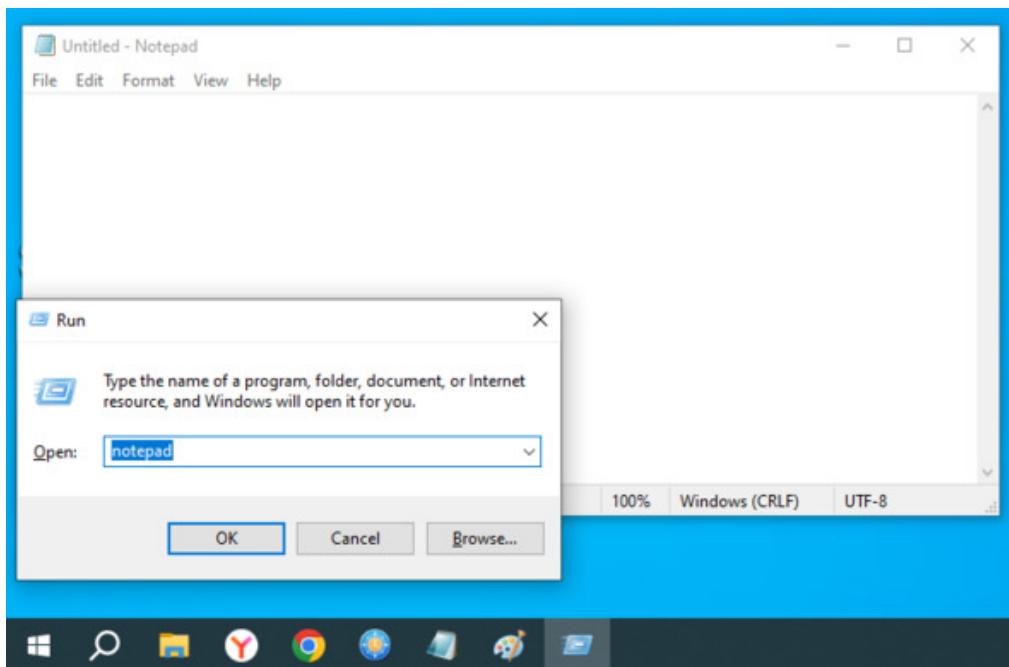


Рис. Запускаем Блокнот

Открываем Блокнот и составляем простую программу, см. рис. Сохраним ее в файле hello.bat. Все наши программы будем складывать в одну «рабочую» папку. В нашем примере это каталог Development на диске Е. Кстати, слова «папка», «каталог», «директория», «folder» – это разные названия одного и того же объекта. Просто их используют в разных ОС и в разное время. Наверное, скоро ещё что-нибудь придумают.

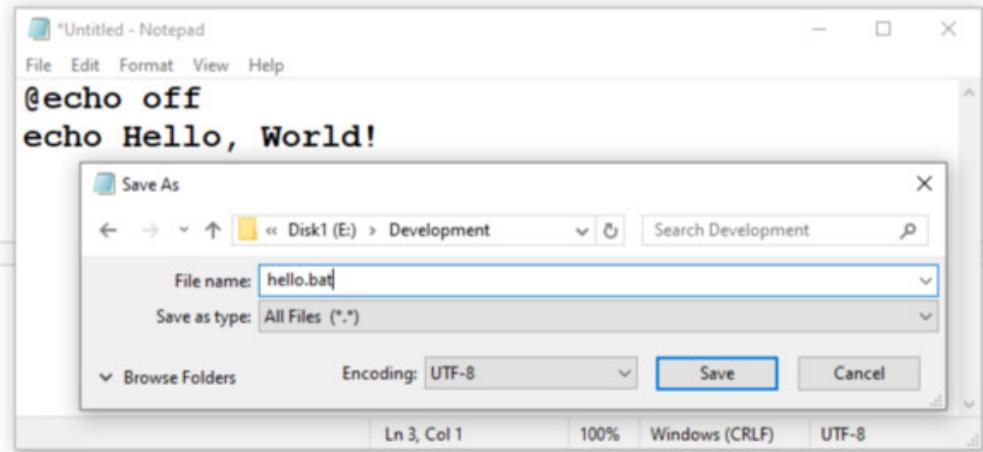


Рис. Первый пакетный файл

В нашей программе две строчки. Первая команда – вспомогательная, она отключает вывод на экран самих команд. Вторая команда – главная, она выводит на экран приветствие. Подробные объяснения по каждой команде и даже по каждой букве или символу можно получить с помощью нейросетей.

Задание. Выясните с помощью Гигачата все подробности о работе нашей программы.

Задание. Просмотрите на Википедии статьи Пакетный файл и Batch file. Выясните, какие расширения бывают у таких файлов.

Программа готова. Можно запускать. Откроем командное окно и перейдем в наш каталог, см. рис. Для начала сменим текущий диск, затем сменим каталог. Выводим на экран список пакетных файлов. Запускаем файл на выполнение, для этого вводим его имя (без расширения) и нажимаем Enter.

В этом примере появляется особая технология – шаблоны. Здесь мы используем самый простой шаблон – звездочку. Она означает любое количество символов или даже отсутствие символов. Вместо того, чтобы писать полностью название каталога Development, мы написали две буквы и поставили звездочку.

```
C:\Users\Valentin>e:  
E:\>cd De*  
  
E:\Development>dir *.bat  
Volume in drive E is Disk1  
Volume Serial Number is 6ACF-36B6  
  
Directory of E:\Development  
  
19.12.2024  13:53              31 hello.bat  
           1 File(s)            31 bytes  
           0 Dir(s)  980 512 325 632 bytes free  
  
E:\Development>hello  
Hello, World!
```

Рис. Запускаем пакетный файл

Задание. Составьте пакетный файл и запустите его на выполнение. С помощью нейросети изучите каждую команду данного примера.

Вернемся к звездочкам. Это пример того, что называется wildcard-символами или просто wildcards. Этот термин wildcard переводится как «шаблон», вместо него можно подставить любые символы. В карточных играх это «джокер», который превращается в любую карту. Это история, это происхождение термина. Может помочь в понимании компьютерных технологий. А еще

это просто любопытный факт – для расширения кругозора. При желании можно копнуть еще глубже и узнать, что «джокер» по-английски означает «шутник», но происходит это слово из совсем другого языка и с совсем другим смыслом.

Задание. Изучите историю с wildcards в компьютерных технологиях и джокером в карточных играх. Выясните, какие еще символы используют в шаблонах для имени файла или каталога.

Теперь запустим наш файл с помощью Проводника – двойным щелчком мыши. Такое ощущение, как будто ничего не происходит. Если повторить опыт и внимательно присмотреться, можно заметить, как что-то мелькнуло. Это открылось командное окно, вывело приветствие и быстренько закрылось – и все это за считанные доли секунды.

Чтобы продлить удовольствие и успеть прочитать сообщение, нужно «остановить мгновение». Для этого есть полезная команда PAUSE – то есть пауза, задержка.

Нам нужно отредактировать наш пакетный файл. Нажимаем правую кнопку мыши и в контекстном меню выбираем Edit – Изменить. Добавляем новую строчку с новой командой, сохраняем и запускаем, см. рис. Теперь мы успеем прочитать сообщение и подумать о смысле жизни. Постепенно приходим в себя и нажимаем любую клавишу. Окно закрывается.

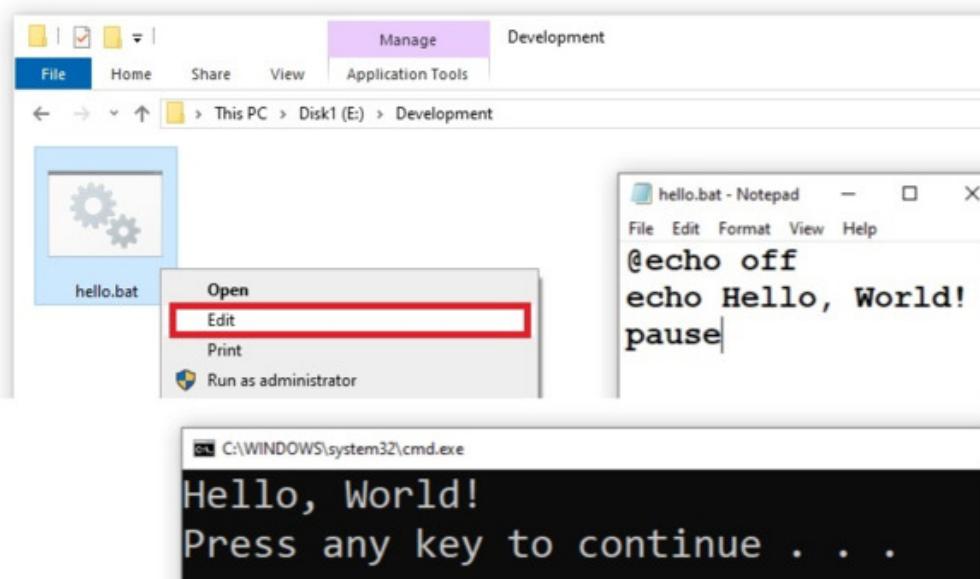


Рис. Поставим на паузу...

Задание. Составьте пакетные файлы с командами ECHO и PAUSE. Запустите их в командном окне и графической среде.

## ИНТЕРПРЕТАТОР

Обсудим наши эксперименты. Нашу программу выполняет интерпретатор командной строки ОС. Он выполняет ее постепенно – по одной строке. Читает одну команду за другой из файла по очереди и выполняет.

Интерпретатор – это такой инструмент для «пошагового» выполнения программ. Английское слово Interpreter означает «переводчик». Он переводит программу с «человеческого» языка на «компьютерный», «машинный» (имеется в виду язык, понятный процессору вычислительной машины).

Если углубиться в технологии программирования, можно встретить трансляторы. Но английское слово translator тоже означает «переводчик». А ещё бывают

компиляторы и много чего другого. Как-то надо это в одну простую схему ложить. Назовем её «Трансляторы в программировании». Для начала покопаемся в простых статьях на Вики, а потом позовем на помощь какого-нибудь чат-бота.

Задание. Просмотрите на Википедии следующие статьи:

- Интерпретатор
- Interpreter (computing)
- Компилятор
- Compiler
- Транслятор
- Translator (computing)

Постройте с помощью PlantUML схему «Трансляторы в программировании», чтобы показать, как связаны разные инструменты программирования.

Выясните, какие языки программирования работают в режиме интерпретатора и компилятора, какие есть у них преимущества и недостатки.

Итак, существуют разные инструменты для запуска программ. Все они так или иначе переводят с одного языка на другой. Это «переводчики», но с разной специализацией. В обычной жизни у переводчиков тоже есть разные направления в работе. Можно работать с написанными/напечатанными текстами. А можно переводить устную речь. Самая напряженная работа у синхронного переводчика — непосредственно во время разговора. Мы их всех называем «переводчики», а вот английские слова имеют разные

оттенки и значения. Пришло время с этой областью ознакомиться. Тем более, что слова транслятор и интерпретатор «унаследовали» эти смыслы. Такая схема улучшает наше понимание компьютерных терминов.

Задание. Выясните, какие значения имеют английские слова *translator* и *interpreter* и как это «отобразилось» на термины транслятор и интерпретатор в программировании.

## **ПРИМЕРЫ ПАКЕТНЫХ ФАЙЛОВ**

Мы познакомились с основной идеей пакетного файла Windows. Оказывается, здесь есть свой язык программирования. Теперь можно начать составлять несложные программы на этом языке.

Задание. Составьте пакетные файлы в соответствии с вариантами заданий. Запустите их в командном окне и через Проводник. Если выполнение какого-то задания вызывает трудности, зовем на помощь чат-бота. При выполнении программ может возникнуть проблема с отображением русских букв. Здесь тоже поможет чат-бот, нужно только его правильно попросить. В любом случае нужно разобраться с каждой командой, каждым обозначением и каждым символом. Конечный результат упражнения — способность написать и запустить аналогичные программы.

— выведите на экран числа от 2 до 20 с шагом 2 с помощью цикла FOR

- создайте директорию, используйте в качестве названия свою фамилию на английском. Перейдите в нее
- выведите на экран приветствие и перенаправьте это сообщение в файл под названием x.txt
- проверьте, существует ли файл под названием x.txt, и удалите его
- запустите блокнот notepad.exe и дождитесь его закрытия перед продолжением работы
- запросите имя пользователя, сохраните его в переменной и выведите на экран приветствие с использованием этой переменной
- спросите у пользователя, сколько минут он уже занимается этой лабораторной работой, и в зависимости от ответа выведите разные сообщения: меньше 30 – «маловато будет...», больше 30 – «молодец, продолжай!»

## ПАРАМЕТРЫ КОМАНДНОЙ СТРОКИ

Один из ключевых инструментов в командной строке – это параметры (аргументы) командной строки. Это просто короткие строчки текста. При начальном знакомстве можно использовать и «параметры», и «аргументы». Конечно, профессиональный разработчик скажет, что это совершенно разные вещи. Особенно, если речь идет о функциях в классических программах. Но об этом потом.

Традиционно, общение пользователя и программ в командной строке – это передача данных в виде строк текста, в виде последовательности символов. На входе и на выходе любой команды у нас используются буквы.

На входе любой команды мы сообщаем дополнительные параметры – через пробел. Это параметры (аргументы) командной строки. Например, мы можем написать команду CD без параметров и получить имя текущего каталога. Можем перейти в корневой каталог текущего диска: CD . Либо мы напишем CD TMP и перейдем в каталог под названием TMP. Каждый раз текущий каталог выводится в начале командной строки в виде приглашения к вводу команд. Дополнительные параметры мы указываем через пробел после имени команды.

Большинство команд ОС имеют дополнительные параметры. Так мы управляем поведением команд. И этот прием можно использовать при составлении пакетных файлов (скриптов).

The screenshot shows a terminal window with the following command history:

```
C:\Users\Valentin>cd  
C:\Users\Valentin  
C:\Users\Valentin>cd \  
C:\>cd tmp  
C:\TMP>
```

Annotations with red arrows point to specific parts of the commands:

- An arrow points to the first "cd" command with the text "Вывести текущий каталог".
- An arrow points to the second "cd \\" command with the text "Перейти в корневой каталог".
- An arrow points to the "cd tmp" command with the text "Перейти в каталог TMP".

Рис. Параметры командной строки

Эти дополнительные параметры описаны в справке по выбранной команде. Чтобы получить описание команды CD, напишем CD /? и нажимаем Enter. В таком описании действует общее соглашение: в квадратных скобках приводятся необязательные параметры. Их можно не указывать. Но если мы захотим их использовать, то квадратные скобки не нужны, см. рис.

```
C:\TMP>cd /?
Displays the name of or changes the current directory.

CHDIR [/D] [drive:][path]
CHDIR [...]
CD [/D] [drive:][path]
CD [...]
```

**В квадратных скобках -  
необязательные параметры**

Рис. Справка для CD

Мы можем сменить текущий каталог, или текущий диск, или и то, и другое. На рисунке приведены примеры использования параметров CD. Еще выясняется, что можно использовать команды CD и CHDIR и получить тот же самый результат. Можно сказать, что CD – это «псевдоним», сокращение, укороченный /сокращенный вариант для длинного названия команды CHDIR. А это, в свою очередь, – сокращение для фразы Change Directory – сменить каталог.

```
C:\Users\Valentin>cd c:\
c:>cd tmp
c:\TMP>chdir \
c:>chdir tmp
c:\TMP>cd /d D:
D:>cd /d c:\tmp
c:\TMP>
```

**Переходим в корневой каталог**  
**Переходим в каталог TMP**  
**Переходим в корневой каталог**  
**Переходим в каталог TMP**  
**Переходим на диск D:**  
**Смена диска и каталога  
одновременно**

Рис. Команда CD с параметрами

Задание. Изучите для команды CD. Попрактикуйтесь в использовании CD. Проверьте, что происходит при попытке перейти в несуществующий каталог или на несуществующий диск. Обратите внимание на две

точки в параметрах команды CD – разберитесь, как с ними работать.

Итак, мы можем передавать дополнительные сведения при вызове команды. Точно так же мы можем передавать параметры при запуске пакетного файла. А потом использовать их внутри нашей программы – как параметры команд.

Далее вам предлагается «придумать» с помощью какого-нибудь интеллектуального бота задания и попрактиковаться с параметрами командной строки. Вот пример:

– прочитайте два параметра командной строки и выведите на экран их значения одной строкой в обратном порядке – сначала второй параметр, затем первый и в конце – имя пакетного файла.

Мы можем попросить чат-бота написать такую программу и объяснить ее. Напомним, что наша цель – не просто «сдать и забыть», а понять новый материал и научиться самим составлять такие скрипты.

**Задание.** С помощью чат-бота составьте 10 простых заданий по созданию пакетных файлов, которые работают с параметрами командной строки Windows. Получите решение по каждому заданию и проверьте его работоспособность. Для каждого пакетного файла получите подробные объяснения каждой строки, каждой команды, каждого символа.

## КОНВЕЙЕР

В командной строке сообщения идут в виде текста. Текст на входе команды и текст на выходе команды. Мы вводим команду, она выводит на экран свой ответ в виде нескольких строчек текста. Вроде бы все понятно и очевидно.

Оказывается, можно взять текстовое сообщение одной команды и подать его на вход другой команды. Так можно построить конвейер – это цепочка команд. Между командами ставят вертикальную черту |. Есть такой символ на клавиатуре, обычно в английской раскладке.

Наш первый пример: команда DIR выводит список каталогов, команда SORT проводит сортировку списка. Каждую команду мы вызываем с дополнительными параметрами. Между командами поставлена вертикальная черта.

Здесь происходит передача и обработка нескольких строк текста.

```
c:\>dir /b /1
adwcleaner
intel
perflogs
program files
program files (x86)
tmp
users
windows
```

```
c:\>dir /b /1 | sort /r
windows
users
tmp
program files (x86)
program files
perflogs
intel
adwcleaner
```

Рис. Первый конвейер

Задание. С помощью чат-бота составьте несложные задания для организации конвейера в командной строке windows и получите решения в виде пакетных файлов с подробным описанием. Убедитесь в их работоспособности.

## ИТОГИ

В этом разделе мы поработали с пакетными файлами для традиционной командной строки Windows. Считается, что это наследие предыдущей операционной системы DOS, что оно устарело и что его скоро отключат. Но пока оно работает, мы можем его использовать — с пользой для себя и для других.

К тому же, освоение файлов BAT/CMD — это хороший первый шаг в освоении скриптов. Во многих случаях, при объяснении современных скриптовых языков на эти технологии ссылаются, как на что-то общеизвестное. Так что владение этим инструментом будет полезно во всех отношениях.

## 2. СКРИПТЫ VBS

Слышно, как скрипит  
пёрышко слегка,  
и ложатся строчки  
на листок.

(музыка С. Туликова,  
слова М. Пляцковского)

Следующий пример языка сценариев – VBScript. Полное название: Microsoft Visual Basic Scripting Edition – выпуск языка программирования Visual Basic для создания скриптов. Можно также говорить Visual Basic Script.

Скрипты \*.VBS можно запускать точно так же, как и пакетные файлы \*.BAT – из командной строки и из Проводника.

**Задание.** Просмотрите в Википедии статью VBScript. Выясните, для каких ОС и для каких задач предназначен данный язык.

Скриптовый язык VBScript построен на основе полноценного языка программирования Visual Basic. Красивое слово Visual использовано в названиях нескольких реализаций языков программирования от Microsoft: Visual Basic, Visual C и др. По-видимому, таким способом нам намекают на визуальные

возможности разработки программного обеспечения – рисование окошек на экране вместо ручного программирования. Корпорация Microsoft предлагает интегрированную среду разработки Visual Studio – здесь в названии тоже присутствует это слово.

Задание. Выясните, в названиях каких продуктов Microsoft использовано слово Visual.

Задание. Просмотрите в Википедии статью VBScript. Выясните, для каких ОС и для каких задач предназначен данный язык.

## **MSGBOX – ВЫВОД СООБЩЕНИЯ**

Программу на языке VBScript можно создать в обычном Блокноте и сохранить в виде текстового файла с расширением.vbs.

Наша первая программа выводит на экран традиционное приветствие.

Для вывода сообщения используем функцию MsgBox – Message Box. Насчет Message

всё понятно, это «сообщение», то есть текст, который печатают на экране. А вот слово Box – это не «коробка», как учили в школе, а «окно». Другими словами, это прямоугольная область на экране, куда что-то выводят.

Текст программы простой и короткий:

MsgBox «Hello, World!»

Строка для вывода записана в кавычках и через пробел от названия функции – без скобок.

Сохраняем как hello.vbs. Дважды щелкаем мышью на нашем файле и запускаем программу.

На экране появляется диалоговое окно с сообщением «Hello, World!». Нажимаем кнопку OK, и окно закрывается.

Наша программа работает в графическом интерфейсе Windows и использует готовые инструменты ОС для создания прикладных программ. В нашем примере это окно, сообщение и кнопка. Также здесь имеется кнопка закрытия окна Close.

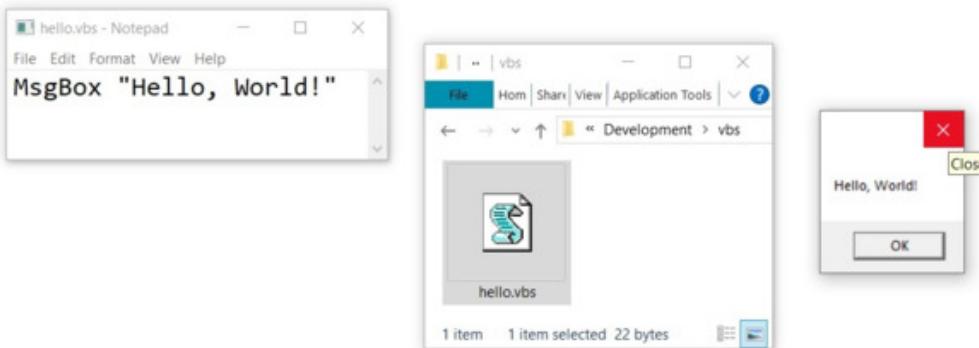


Рис. Выводим приветствие

Обратим внимание на иконку. На картинке изображен длинный бумажный свиток. Этот свиток постепенно раскручиваются, и на нём просматриваются строки текста. Такая иконка визуально передает нам идею «сценария работы».

Задание. Создайте свою программу с использованием MsgBox и выведите на экран свое собственное сообщение.

## INPUTBOX – ВВОД ДАННЫХ

Следующая функция `InputBox` – это средство для ввода текста.

Мы спросим пользователя, как его зовут, и выведем персональное приветствие.

```
name = InputBox ("Please enter your name:")
```

```
MsgBox "Hello, " & name & "!"
```

В первой строке программы мы запрашиваем имя пользователя.

Записываем полученное имя в переменную `name`.

Затем во второй строке мы выводим персональный «привет».

Для этого мы объединяем строки (приветствие + имя + восклицательный знак) с помощью символа `&` – «амперсанд».

Полученную строку выводим на экран.

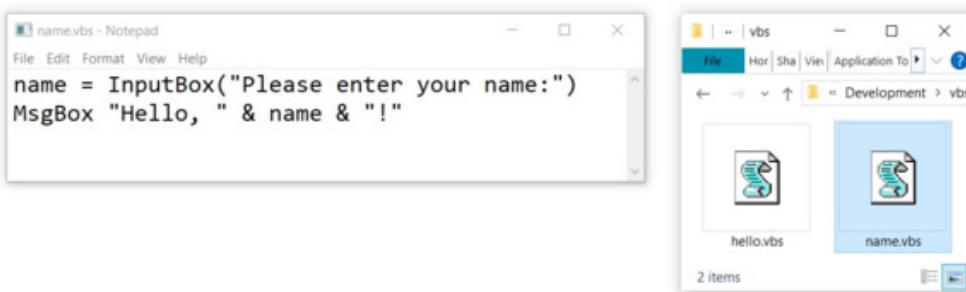


Рис. Программа с окном ввода

Задание. Просмотрите в Википедии статью Амперсанд и выясните его значение и происхождение.

Сохраняем скрипт как name.vbs. Запускаем на выполнение, вводим имя и получаем персональный привет.

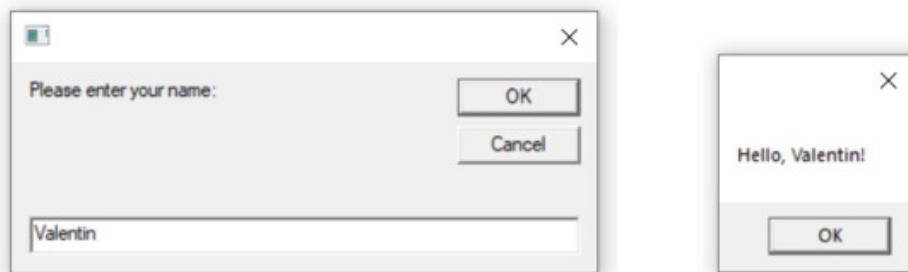


Рис. Вводим имя

Вроде бы всё просто. Но при вводе русского текста в английской версии Windows могут возникнуть проблемы.

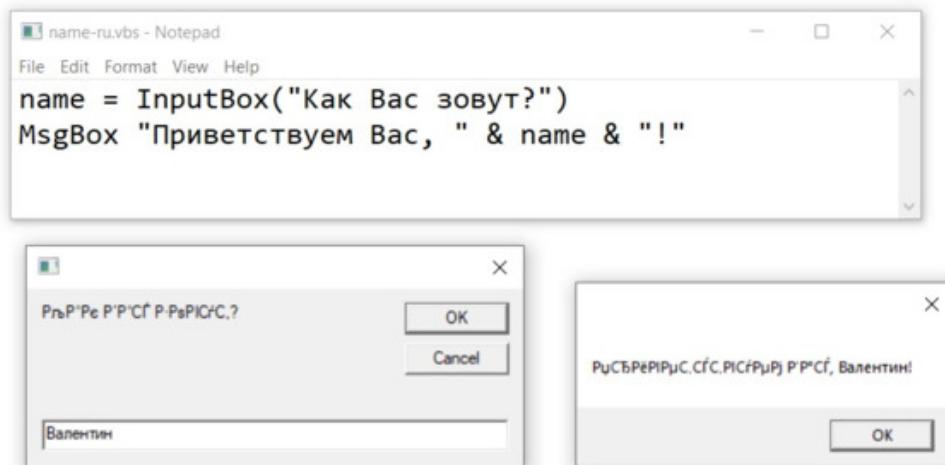


Рис. Проблемы локализации

Задание. Составьте свой скрипт по приведенному выше примеру. Проверьте, как он работает при выводе русских букв.

Возможно, вам повезет. Но на нашем компьютере пришлось повозиться и поискать решение проблемы локализации, региональных настроек и кодовых страниц. Работающее решение оказалось простым. Начинаем сохранять файл и выбираем кодировку ANSI вместо UTF-8, которая у нас была по умолчанию.

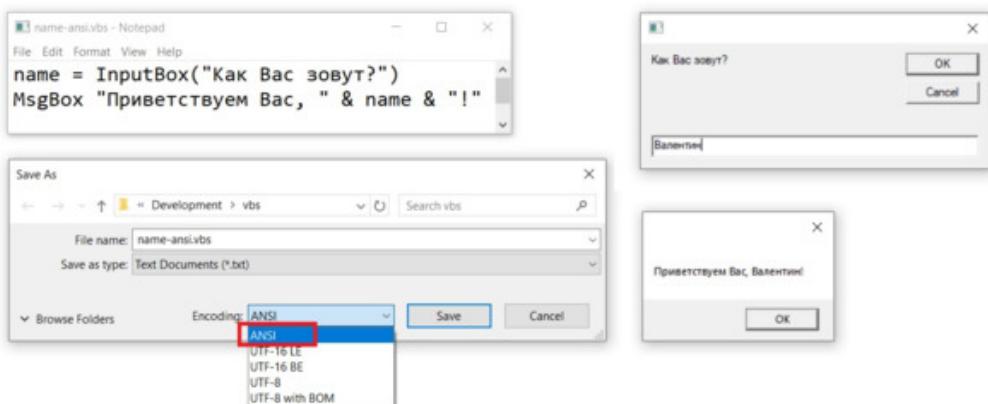


Рис. Говорите по-русски...

Задание. Сохраните свой скрипт в разных кодировках и проверьте его реакцию на русские буквы.

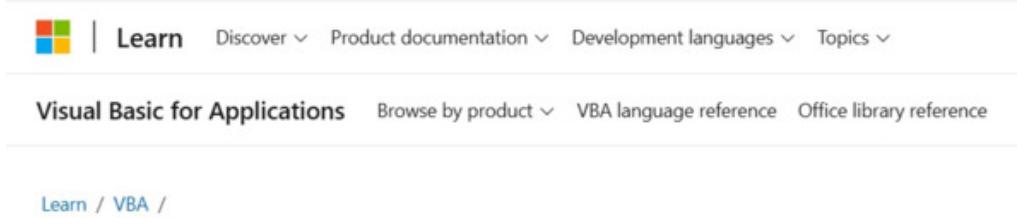
## MSGBOX – ПАРАМЕТРЫ ОКНА

Мы познакомились с функцией MsgBox и посмотрели, как работает вывод текста в окне. У этой функции есть много дополнительных возможностей – и для украшательства, и для дела. Подробное описание функции можно найти на страничке разработчиков – у корпорации Microsoft.

На рисунке приводится пример того, что можно найти через Google, Yandex или просто через поиск на сайте компании. Адрес странички:

<https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface->

## [help/msgbox-function](#)



The screenshot shows the Microsoft Learn website for Visual Basic for Applications. The top navigation bar includes links for 'Discover', 'Product documentation', 'Development languages', and 'Topics'. Below that, specific links for 'Visual Basic for Applications', 'Browse by product', 'VBA language reference', and 'Office library reference' are shown. A breadcrumb trail at the bottom left indicates the current page path: 'Learn / VBA / MsgBox function'. The main title 'MsgBox function' is prominently displayed, followed by a brief description: 'Displays a message in a dialog box, waits for the user to click a button, and returns an Integer indicating which button the user clicked.' Below this, the 'Syntax' section is introduced with the code snippet: 'MsgBox (prompt, [ buttons, ] [ title, ] [ helpfile, context ])'. It also notes that the function has named arguments.

Рис. Описание функции MsgBox

В описании указан один обязательный аргумент `prompt` – сообщение для вывода на экран. Вообще-то слово `prompt` переводится как «приглашение к действию» или «подсказка». Мы с этим словом сталкиваемся, когда составляем запрос к нейросети. Еще это слово означает «приглашение к вводу команд» в командной строке ОС. И вот теперь это означает «сообщение на экране». Интересно, почему?

Ответ на эту загадку – второй аргумент – `buttons` – кнопки. Этот аргумент указан как `optional` – необязательный. Его можно и не указывать. Поэтому он приводится в описании в квадратных скобках. Это общепринятое обозначение у программистов.

По умолчанию в нашем окне была всего одна кнопка – ОК. При желании можно заказать для этого

окна и другие кнопки. Тогда функция может вернуть соответствующее число – номер кнопки, которую нажал пользователь.

Обратим внимание, что в круглых скобках указаны arguments – «аргументы функции». А еще в программировании встречается слово «параметры». Иногда параметры и аргументы путают или считают их синонимами. Тем более, что они появляются в скобках после имени функции. Но это не одно и то же. Пришло время разобраться с этим вопросом и узнать что-нибудь новое.

Задание. Выясните, в чем состоит разница между аргументами и параметрами функции в программировании.

Итак, начинаем выбирать кнопки в окне. Можно задать их целыми числами от 0 до 5, а лучше взять готовые константы. Это поможет читать программу без справочников.

В качестве примера рассмотрим значение 1. Ему соответствует константа vbOKCancel. Здесь в названии константы сразу все ответы закодированы:

vb – константа для языка программирования Visual Basic

OK – вывести кнопку OK, то есть «Согласен»

Cancel – вывести кнопку Cancel, то есть «Отменить»

Указываем параметры в скобках, через запятую. Запускаем программу и получаем сообщение об ошибке:

Cannot use parentheses – Нельзя использовать круглые скобки.

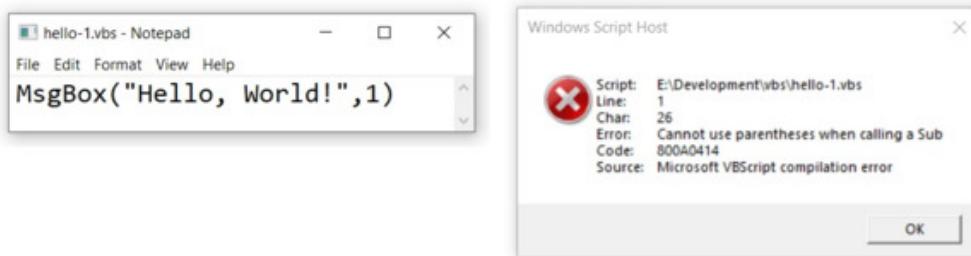


Рис. Скобки явно кому-то помешали

Присмотримся к предыдущему рисунку. Там говорится Visual Basic for Applications – VBA. И это не совсем то, что мы сейчас используем – у нас Visual Basic Script. Единственное отличие будет в том, что скобки нам не нужны. И в первоначальном примере с выводом сообщения скобок не было. В остальном всё должно сработать.

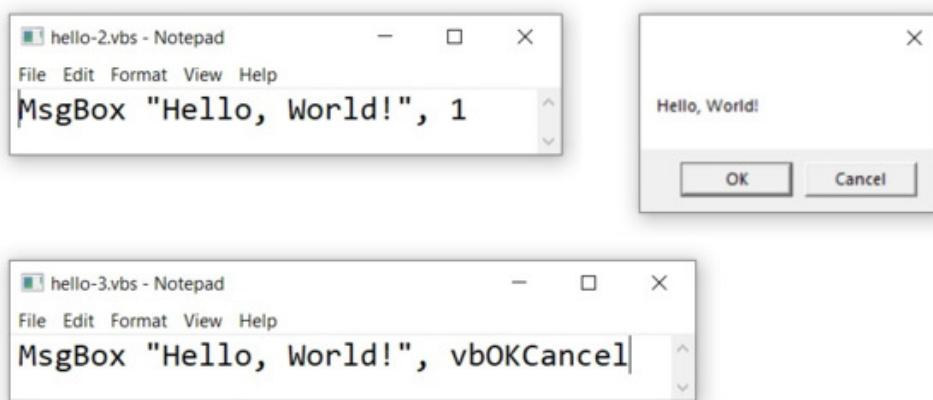


Рис. Вызываем функцию без скобок

Итак, указываем сообщение и единичку (или константу) – и получаем две обещанные кнопки. Теперь можно увидеть разницу. Если написана единичка – придётся думать и вспоминать, что это такое. Если написано название константы – оно само всё объясняет.

Задание. Составьте скрипты и испытайте все стандартные константы для задания кнопок.

Продолжаем читать описание функции. Кроме кнопок, мы можем вывести в окне различные иконки, чтобы подчеркнуть важность нашего сообщения: Для этого используют числа 16, 32, 48, 64. Опять же лучше взять константы. Так вот, выбранные числа или константы просто складывают между собой. Полученная сумма одновременно сообщает и про кнопки, и про иконки.

Берем для примера константу `vblInformation` и получаем иконку «информационное сообщение». В этом примере стандартные константы делают программу более понятной, чем число 65, то есть сумма чисел 1 и 64.



## Рис. Сумма констант

Задание. Составьте скрипты и проверьте, какие иконки выводятся в окне.

Мы «сложили» кнопки и иконки. Но это еще не все! Мы можем задать выбор кнопки по умолчанию и характер «поведения» окна. Это будут дополнительные слагаемые в нашей сумме.

Задание. Составьте скрипты и проверьте, как меняется поведение окна при задании дополнительных констант.

Переходим к третьему аргументу – title – заголовок окна. Это должна быть строка символов. Либо мы указываем строку в двойных кавычках, либо задаем «строковое выражение». То есть это будет имя переменной или какое-нибудь выражение, а в результате вычислений должна получиться строка.

И второй, и третий параметры не являются обязательными. Мы можем указать оба параметра, или только один из них, или ни одного. Поскольку buttons – это числовое выражение, а title – строковое, компьютер не запутается и всё поймёт правильно: число – это кнопки, а строка – это заголовок.

Задание. Проверьте, как работают параметры buttons и title – вместе и по отдельности.

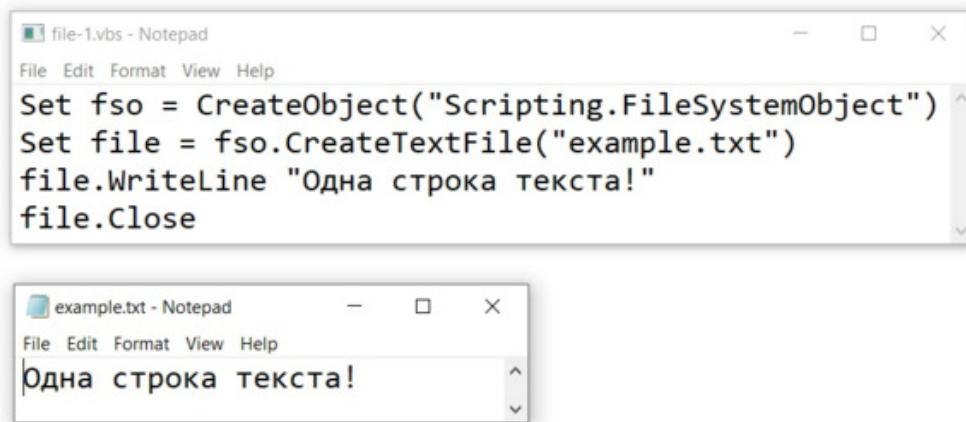
На этом мы закончим освоение функции MsgBox. У нее есть и другие параметры, но мы познакомились с самыми главными. Остальное вы сможете освоить самостоятельно — после легкой разминки и обстоятельного выполнения первых шагов.

## СОЗДАЁМ ТЕКСТОВЫЙ ФАЙЛ

Далее мы рассмотрим более сложный пример — программа из четырех строк кода!

Мы составим скрипт, который создает текстовый файл и записывает в него строчку текста.

Команды будут немного сложнее, чем в предыдущих программах. Запоминать их не нужно. А вот познакомиться и понять будет полезно. Текст программы — на рисунке.



```
file-1.vbs - Notepad
File Edit Format View Help
Set fso = CreateObject("Scripting.FileSystemObject")
Set file = fso.CreateTextFile("example.txt")
file.WriteLine "Одна строка текста!"
file.Close

example.txt - Notepad
File Edit Format View Help
Одна строка текста!
```

Рис. Запишем строчку в файл

В первой строке мы создаем «объект файловой системы». И мы видим здесь новое слово Set. В предыдущих примерах мы работали с простыми

переменными. Для них присвоение значений делается привычным оператором `=`. Для присвоения значений объектам используют оператор `Set`. Здесь идет работа со ссылками на объекты, и это организовано сложнее, чем работа с простыми переменными. Мы создаем объект `fso`; это набор инструментов (методов) для работы с файловой системой. Напомним, что «метод» в объектно-ориентированном программировании – это на самом деле функция, которая «привязана» к объекту. С помощью этих инструментов мы сможем создать конкретный файл.

Во второй строке мы создаем новый текстовый файл. Для этого мы обращаемся к объекту `fso` и вызываем метод `CreateTextFile`. Здесь мы указываем имя объекта, затем ставим точку и указываем название метода (функции) для вызова. В результате «на выходе» мы получаем новый объект `file`, который представляет вновь созданный файл. У этого объекта есть свои методы, к которым мы далее обратимся.

В двойных кавычках мы указываем имя файла. Этот параметр мы указываем в круглых скобках. И этот файл появится в текущем каталоге, в котором мы запускаем наш скрипт. При желании можно вместо имени указать полный путь к файлу. При большом желании можно создать новую переменную, записать в нее путь к файлу, а потом указать имя переменной в качестве параметра `filename`.

В наших примерах мы упрощаем все программы. Это очень помогает в понимании. Особенно при первом знакомстве с материалом. Вначале нужно добраться до сути, до основ. Когда вы поняли основную идею, можно будет добавлять второстепенные детали.

В третьей строке программы мы записываем одну строчку в файл. Здесь мы берем объект file и вызываем его метод WriteLine. Через пробел указываем строчку текста для записи. В этом случае – через пробел!

В этом примере мы видим два подхода к вызову функции и передаче аргументов. Пробелы и скобки при передаче параметров/аргументов в программах VBScript – это особенность, связанная с наследованием прежних наработок и попытками обеспечить совместимость со старыми версиями. Нечто подобное можно встретить и в разных версиях языка Python. А чтобы все запутать, одну и ту же конструкцию называют то командой, то оператором, то функцией, то методом. На этапе первого знакомства эту особенность придется просто «понять и простить».

Четвертая строка – закрытие файла. Здесь всё просто: объект file и метод Close. В других языках программирования при вызове функции нужно обязательно указывать скобки, даже если нет ни одного аргумента. В скриптах VBS некоторые моменты уже упростили. К сожалению, не все, а только некоторые.

Задание. Составьте скрипт для записи строки в файл. Запустите скрипт и убедитесь, что текстовый файл появился в текущем каталоге. Усложните программу: добавьте полный путь к файлу. Затем внесите еще больше сложности: вначале запишите путь к файлу в отдельную переменную, а потом передайте ее как аргумент функции. Наконец, добавьте еще больше сложности: объявитте каждую переменную перед ее

использованием с помощью ключевого слова Dim. Можете также добавить необязательные аргументы при вызове каждой функции (метода). Вставьте подробные комментарии. Вот теперь мы получаем красивую «грамотно оформленную» программу, на которую будет приятно посмотреть, но гораздо труднее прочитать и понять.

## ДОБАВИМ ТВОРЧЕСТВА

Мы рассмотрели буквально пару инструментов скриптового языка VBScript. Далее вам предстоит составить несколько несложных программ для освоения основных стандартных конструкций и инструментов.

**Задание.** Составьте скрипты VBScript для освоения и демонстрации следующих средств программирования:

- ввод и вывод числовых значений
- текущая дата Date
- текущее время Time
- текущая дата и время Now
- цикл For...Next
- цикл Do While... Loop
- конструкция выбора Select Case
- условная конструкция If...Then... Else
- вывод сообщения через WScript.Echo
- пауза на несколько секунд с помощью WScript.Sleep
- чтение параметров командной строки через WScript.Arguments

— сохранение команды ОС в виде пакетного (текстового) файла и запуск этого нового файла на выполнение

При составлении и запуске скриптов мы сталкиваемся с новыми названиями: Cscript и Wscript. Это не просто технология работы со скриптами. За этими программами целая история. Пришло время с этой историей познакомиться.

Задание. Выясните, как работают Cscript и Wscript, где эти программы расположены и почему их каталог до сих пор так называется.

## **ПРЕКРАЩЕНИЕ ПОДДЕРЖКИ**

На момент составления данного учебного пособия VBScript работает в ОС Windows. Однако, поддержка этого скриптового языка в продуктах MS будет постепенно прекращена. Заявлено несколько этапов — на ближайшие годы.

Нам рекомендуют использовать другие средства автоматизации, например, PowerShell. Очень скоро и до него доберемся. И увидим, как все в жизни связано.

Задание. Выясните, почему Microsoft прекращает поддержку VBScript и составьте список этих причин.

## **ИТОГИ**

Скрипты VBS работают в графической среде Windows. При этом программы на VBS достаточно короткие. Здесь нет длинного и подробного оформления программ, нет «лишних» скобок и прочего «украшательства».

Скрипты – это коротко и ясно. При этом скрипты VBS содержат основные средства традиционных языков программирования. Здесь можно освоить основные идеи программирования, не отвлекаясь на мелкие подробности.

## **3. МАКРОСЫ VBA**

Макросы могут сделать жизнь проще,  
но только если знать, как ими  
пользоваться  
(нейросетевой фольклор)

Мы уже столкнулись с названием Visual Basic, когда пытались почитать описания команд VBScript. Поясним, что есть язык программирования Basic, есть его реализация от Microsoft под названием Visual Basic, а еще есть применение этой технологии для создания скриптов-сценариев ОС под названием VBScript, или для краткости VBS. Мало того, внутри Microsoft Office появляется еще один Basic для автоматизации работы с документами и таблицами – Visual Basic for Applications, или сокращенно VBA. Теперь эти программы для «офисных» задач называют «макросы». Полное название: «макро-команды» = «макро» (большие) + «команды» (инструкции). Зная VBS, можно довольно быстро освоить VBA.

### **ПЕРВЫЙ МАКРОС**

На простом, «бытовом» уровне можно создать макрос путем записи операций с текстом. Так же, как записывают речь человека на диктофон.

Кстати, при записи макроса рядом с курсором появляется символ магнитофонной кассеты – намек на диктофонную запись. Кассеты уже вышли из моды, а символ остался.

Попутно отметим еще один символ из прошлого. Во многих программах есть изображение дискеты на кнопке Сохранить – Save. Дискеты были популярны много лет назад. С тех пор иконка дискеты до сих присутствует на экране, хотя для многих она уже изображает незнакомый музейный экспонат.

Задание. Найдите в интернете изображения кассет и дискет. В ходе работы с пакетом MS Office обратите внимание на их схематичные изображения.

Запустим Microsoft Word и создадим новый документ. Введем несколько строк текста. Если мы захотим немного украсить наш текст, можно сделать это вручную. Например, в начале каждой строки добавим какой-нибудь символ и поставим после него пробел. И здесь мы будем повторять одни и те же действия – в одном и том же порядке:

- переходим в начало строки
- печатаем символ @
- нажимаем пробел
- переходим на новую строку

Можно повторить это пять-шесть раз, но если таких строк у нас сотни или тысячи, и с каждой нужно сделать одно и то же, должна появиться мысль: а нельзя это как-то автоматизировать?

## Как записать макрос

Первый шаг  
Второй шаг  
Третий шаг  
Четвертый шаг



## Как записать макрос

@ Первый шаг  
@ Второй шаг  
@ Третий шаг  
@ Четвертый шаг

Готово!

Готово!

Рис. Рутинная работа

Запись макроса прячется где-то в глубинах «многоэтажного» меню. При большом желании все это можно найти и настроить. Мы поступим попроще. Мы заранее знаем, что макрос где-то есть. Вызываем встроенного помощника и пишем слово «макрос».

Нам дают список вариантов действий. Выбираем пункт Записать макрос. Здесь даже всплывает «очень информативная» подсказка: если нажать кнопку Записать макрос, будет запущена запись макроса.

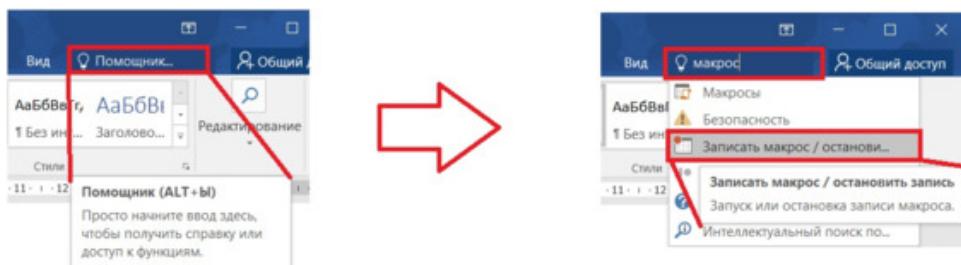


Рис. Вызываем помощника

Нажимаем кнопку Запись макроса. На экране появляется диалоговое окно Запись макроса. Здесь мы укажем имя макроса. А также выберем место для

хранения этой программы. В разделе Макрос доступен для... можно указать все документы. В этом случае макрос будет записан в шаблон Normal.dotm и будет работать во всех документах. Мы выбираем второй вариант – текущий документ. Теперь макрос будет храниться внутри нашего файла и не помешает в работе с другими документами. Добавляем краткое описание.

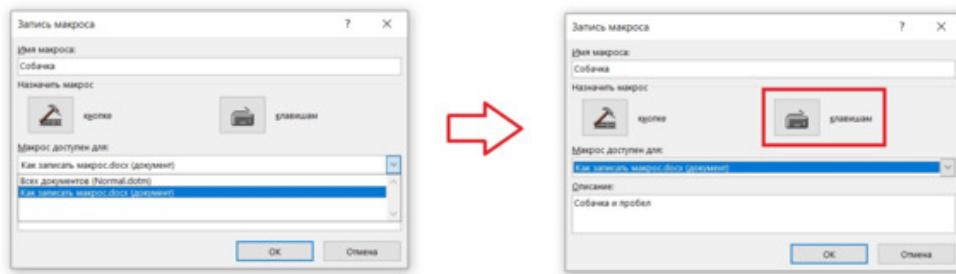


Рис. Настраиваем запись

Назначим макрос сочетанию клавиш [Crtrl + Shift + x]. Убеждаемся, что на эту комбинацию никто не претендует: Текущее назначение – нет. Сохраняем изменения в текущем документе. Нажимаем кнопку Назначить.

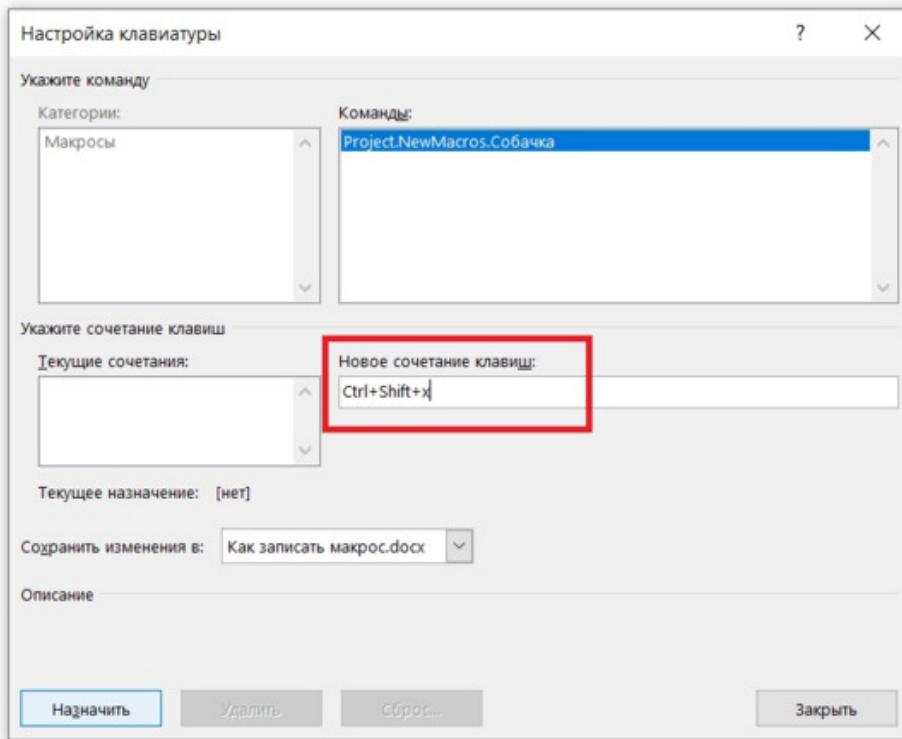


Рис. Сочетание клавиш

Начинаем запись. Нажимаем [Home], чтобы перейти в начало строки. Вводим символ @, пробел. Нажимаем стрелку вниз – Down. Останавливаем запись.

Сохраняем работу как Документ Word с поддержкой макросов \*.docm.

Проверяем наш макрос работе. Нажимаем сочетание [Ctrl + Shift + x]. С каждым нажатием выполняется макрос и делается новый шаг по украшению документа.

Макрос где-то записан, и теперь мы можем изучить текст этой программы. Снова вызываем Помощника, пишем слово Макрос и выбираем пункт Макросы. В диалоговом окне Макрос указываем наш текущий документ и выбираем наш единственный

и неповторимый макрос под кодовым названием Собачка. Нажимаем кнопку Изменить.

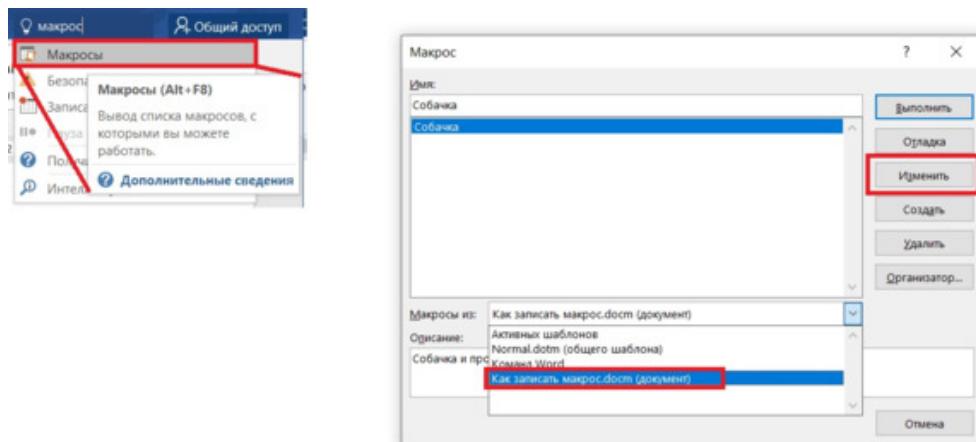


Рис. Снова вызываем Помощника

Открывается новое диалоговое окно Microsoft Visual Basic for Applications. Перед нами полноценная среда разработки программного обеспечения. Здесь есть редактор исходного текста программы. В левой части мы видим список в виде дерева. В рамках текущего проекта Project (Как записать макрос) имеется один программный модуль NewMacros.

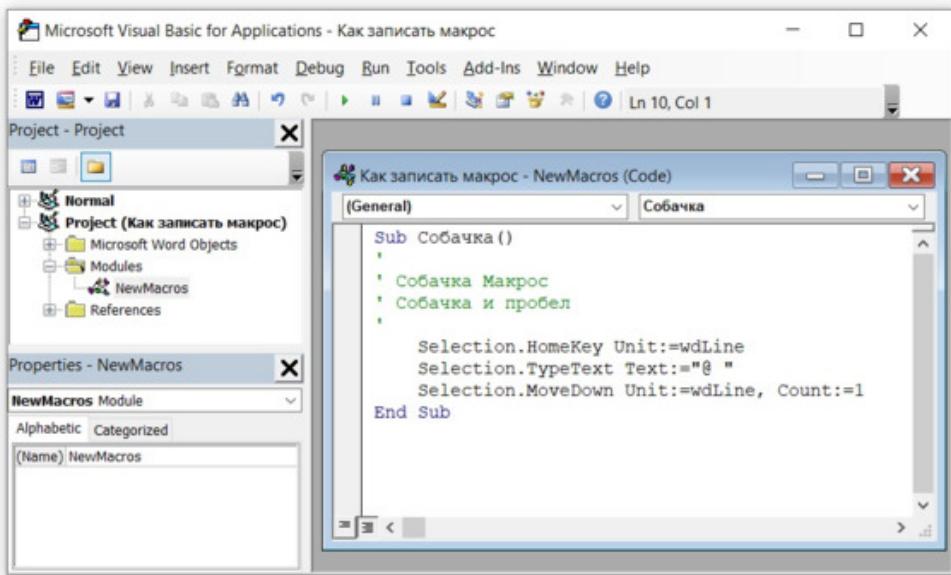


Рис. Макрос – это программа

Обсудим исходный текст нашей программы. Первая и последняя строки содержат ключевое слово `Sub`. Это объявление нашей процедуры, то есть подпрограммы, или по-английски `Subroutine`. Наша процедура только выполняет нужные действия. При этом она не возвращает никаких значений.

Далее идут комментарии. Как видим, строки комментария начинаются с символа апострофа. В этих комментариях указано название нашего макроса, затем приводится текст описания, который мы сами вводили при его создании.

Далее в трех строчках записаны наши действия. Здесь `Selection` – это объект, над которым мы выполняем действия. Название метода указано через точки после имени объекта.

Метод `HomeKey` выполняет первое действие – мы нажали клавишу `Home` (Начало), чтобы переместить курсор в начало строки. Клавиша по-английски называется `Key`. Дополнительно указана «единица

измерения» Unit. Этот параметр получает значение wdLine, что означает «текущая строка документа Word» – Word Document Line. Напомним, что слово Line может означать не только линию (черту на бумаге), но и строку текста. Присваивание значений в данном случае выполняется с помощью символов:= (двоеточие и равно). Это тоже традиция, наследие предыдущих разработок – как и скобки с пробелами.

Метод TypeText – мы ввели текст с клавиатуры. Слово Type здесь означает «печатать на клавиатуре или печатной машинке». Что именно печатать – об этом говорит параметр Text. Ему присваивают значение “@” – это строка символов в двойных кавычках (символ @ и пробел).

Переход на следующую строку:

Метод MoveDown – мы передвинули курсор вниз на одну строчку. Для этого указано, что мы работаем (Unit) со строками (wdLine) и что действие нужно выполнить один раз (Count:=1). В этой команде также можно перемещаться по словам, предложением, абзацам и т. д. – за это как раз и отвечает параметр Unit.

Задание. Запишите свой собственный макрос и изучите текст полученной программы.

## РЕЖИМ РАЗРАБОТЧИКА

Для дальнейших опытов нам нужно включить режим Разработчика. Имеется в виду разработчик программного обеспечения, то есть программист, или

просто «разраб». По-английски Developer.

Выбираем в верхнем меню Файл – Параметры – Настроить ленту – Разработчик – OK.

В разных версиях Word будут небольшие отличия, но общий принцип такой: по умолчанию в меню показано не всё. Мы сами можем включить те вкладки и кнопки, которые нам нужны.

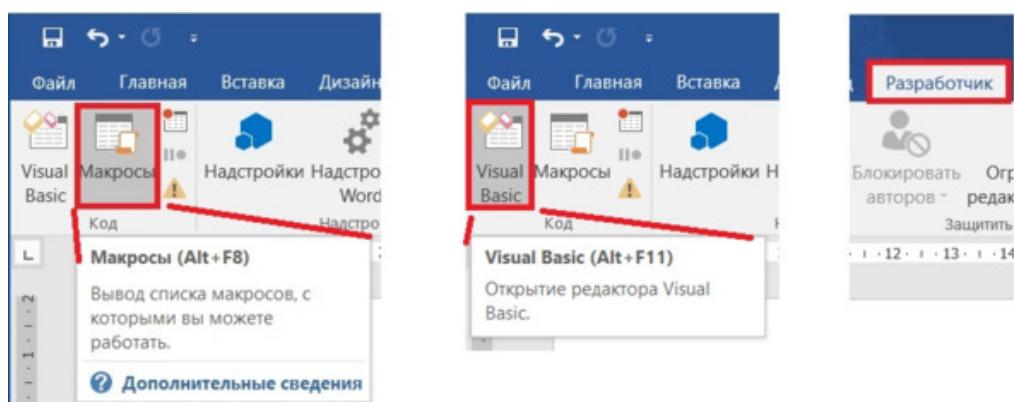


Рис. Как стать «разработом»

Теперь на ленте мы можем открыть новую вкладку Разработчик. В группе Код находим пару полезных кнопок: Макросы и Visual Basic. Наводим на них курсор и читаем всплывающую подсказку. Можно даже перейти по ссылке Дополнительные сведения и почитать описание от авторов этого офисного пакета.

Если все-таки нажать на кнопку Макросы, получим знакомое диалоговое окно Макрос. Это будет список имеющихся макросов.

Для создания нового макроса нажимаем Visual Basic и оказываемся в редакторе кода (исходного текста программы). Мы здесь уже были, когда рассматривали текст первого макроса. Теперь нам нужно добавить новый модуль: Insert – Module. Делается это либо через верхнее меню, либо нажатием правой кнопки мыши

по элементу Modules на дереве проекта в левой части окна.

Когда говорят «модуль», обычно имеют в виду «программный модуль», или «модуль программы», то есть процедуру или подпрограмму. Это опять проблема «наследования» и «совместимости». Разные названия у одной и той же вещи. Просто разные части программы разрабатывали разные программисты и в разное время. Каждая команда сделал свой вклад и оставила свой «след»: куски исходных текстов (кода) и красивые названия (терминология). Пользователи к этому уже привыкли, и что-то переделывать будет уже слишком поздно и слишком дорого.

Если рассмотреть интерфейс этого редактора, можно заметить другой стиль оформления (ленты нет, иконки другие). Похоже, что эта часть программы была полностью унаследована от предыдущей версии пакета, причем от очень предыдущей. Желающие могут провести собственное расследование и выяснить, в какой версии Windows были такие иконки.

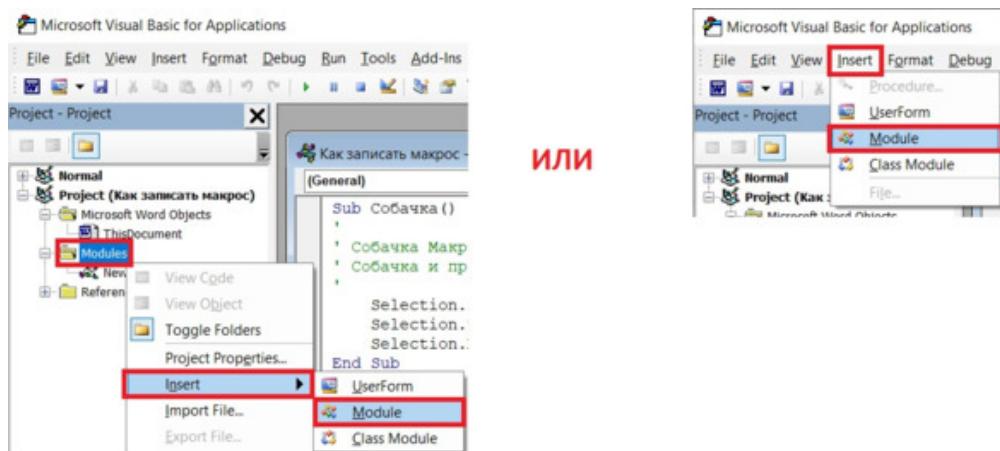


Рис. Добавляем макрос в проект

В новом окне редактирования модуля вводим текст программы. Сохраняем изменения, нажав кнопку Save. Закрываем окно редактора кода.

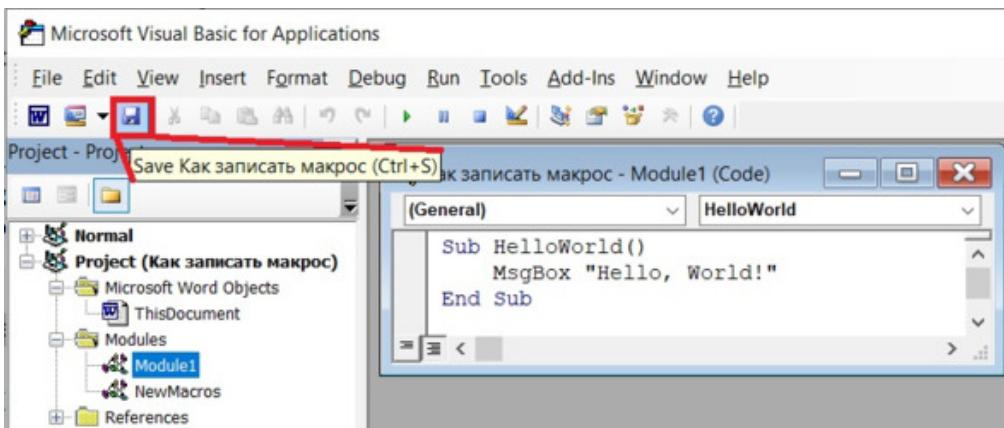


Рис. Новый модуль-макрос-процедура-подпрограмма

Обсудим текст нашей программы.

В первой строке с помощью ключевого слова `Sub` мы объявляем нашу подпрограмму (процедуру) под названием `HelloWorld`. Пустые круглые скобки сообщают, что при вызове нашей процедуры не нужны никакие параметры.

Вторая строка уже должна быть нам знакома. Это окно с сообщением – точно такое же, как в первой программе на VBScript.

Наконец, в третьей строке мы сообщаем, что на этом описание нашей процедуры завершается.

Макрос готов, можно запускать! Открываем вкладку Разработчик, раздел Код и нажимаем кнопку Макросы. Открывается диалоговое окно Макрос. Выбираем наш макрос `HelloWorld` и нажимаем кнопку Выполнить. Появляется знакомое окно с приветствием. У окна

имеется заголовок Microsoft Word. Мы его отдельно не заказывали, но и возражать не будем.

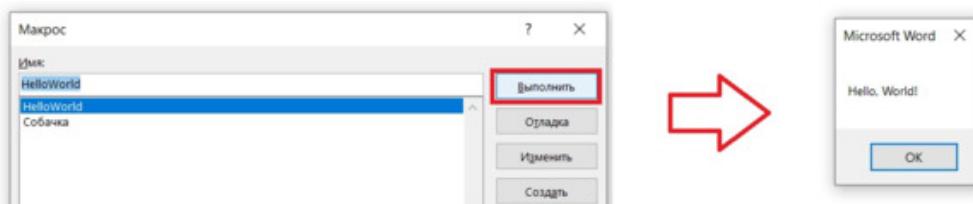


Рис. Запускаем макрос

Задание. Создайте свой модуль и определите новый макрос. Убедитесь, что он работает так же, как и программа на VBS.

Для удобства дальнейшей работы переименуем наши модули. Для этого выбираем нужный модуль, переходим в раздел Properties. Далее в графе Name (Имя) вводим новое имя модуля.

Поясним, что модуль и процедура – это разные вещи. Можно сказать, что модуль – это своего рода «контейнер», или «папка», или страничка программы для хранения процедур. Внутри одного модуля можно держать несколько процедур. Вызываем процедуры по имени, которое указано в её описании, например, HelloWorld. Название модуля может пригодиться для других целей. На данном этапе это просто средство разложить наши процедуры по «папкам».

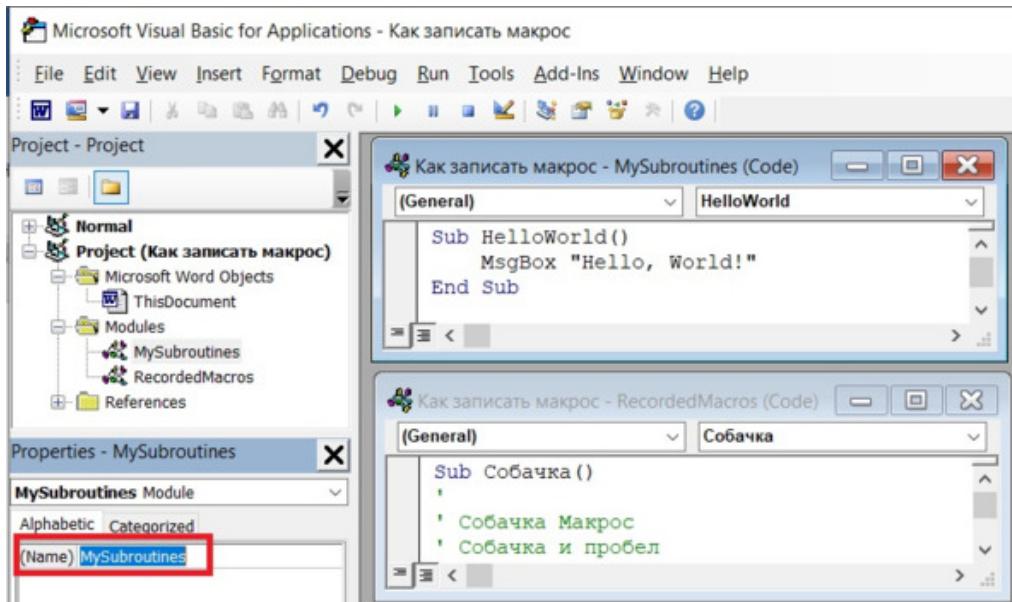


Рис. Новые названия модулей

Проверим, как работают параметры `MsgBox`. Зададим пару кнопок и «информационную» иконку с помощью суммы констант. Мы такой опыт уже делали на VBS.

Определим новую процедуру в нашем модуле `MySubroutines`. При вводе стандартных констант появляется выпадающий список, где можно выбрать нужное название. Это удобно. Помощь при вводе текста – это одна из дополнительных возможностей среды разработки.

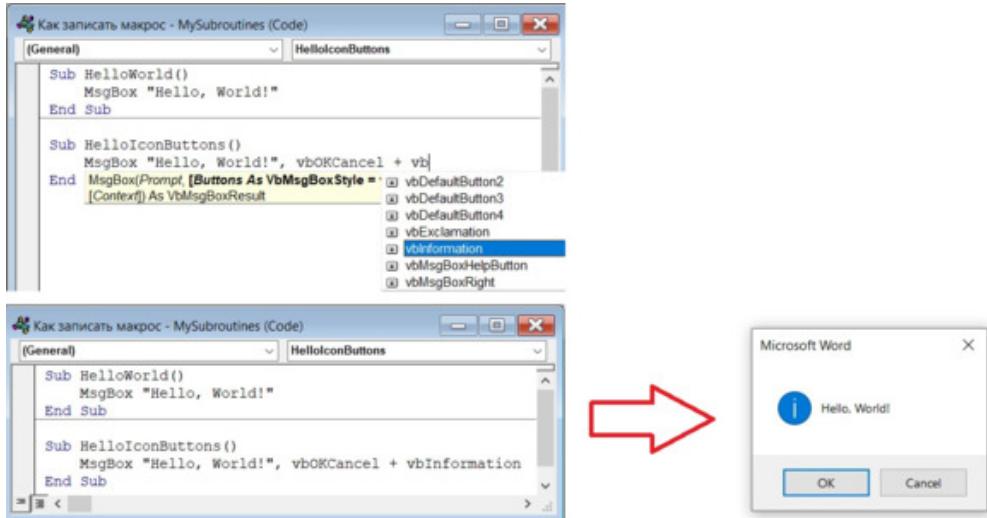


Рис. Одна иконка и две кнопочки

**Задание.** Проверьте, как работают дополнительные параметры MsgBox.

Остальные средства программирования Visual Basic должны работать так же, как в нашем предыдущем разделе в VBScript. Единственное отличие – это начало конец объявления процедуры. Проверим.

**Задание.** Проверьте работоспособность программ, составленных для VBS.

## LIBREOFFICE + VBA

Как вы наверное уже знаете, Microsoft Office – это продукт коммерческий, и для его использования нужна платная лицензия. Рассмотрим бесплатную альтернативу – Libre Office. Это название уже намекает на свободное программное обеспечение. Слово Libre переводится как «свободный». Это бесплатный продукт, который можно использовать без ограничений.

По крайней мере – на момент написания данного учебного пособия.

Этот пакет можно установить на компьютер традиционным способом. Но есть и другой вариант – Portable – переносимый. Он подойдет для наших опытов, и программу можно будет легко удалить после использования. Переносимую версию нужно скачать и развернуть в любом каталоге. Например, на рабочем диске. Оттуда мы его и запускаем. Для удаления программы просто удаляем всю папку. При этом мы не «засоряем» системный реестр. К тому же, нам не потребуются права администратора, и наши опыты можно будет проделать в любом дисплейном классе, имея ограниченные права пользователя.

Адрес странички для скачивания переносимой версии:

<https://www.libreoffice.org/download/portable-versions/>

Этот адрес со временем может измениться, но нам всегда придут на помощь любая поисковая машина.

Задание. Скачайте и разверните переносимую версию Libre Office.

Пакет Libre Office хранит документы в своем формате, но старается поддерживать совместимость с Microsoft Office. Это касается и работы с макросами. Проверим, как поведут себя наши макросы из предыдущего раздела.

Запускаем программу LibreOfficeWriter. Создаем новый документ Writer.

Начинаем работать с макросами: Tools – Macros – Edit Macros.

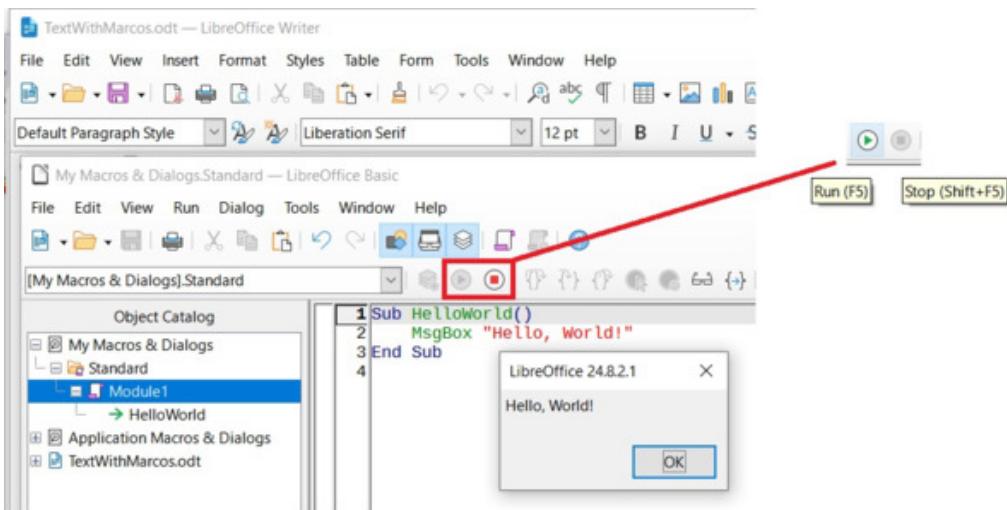


Рис. Первый макрос работает!

Перед нами среда разработки, похожая на то, что мы уже использовали. Можно скопировать и вставить текст нашей первой программы. Здесь тоже можно видеть древовидную структуру и первый модуль. Вводим текст программы и сохраняем изменения. Для запуска макроса на выполнение имеется кнопка Run, для остановки выполнения – кнопка Stop.

Проверим дополнительные параметры. Добавим заголовок окна, «информационную» иконку и две стандартные кнопки: OK и Cancel. В целом, получаем аналогичный результат.

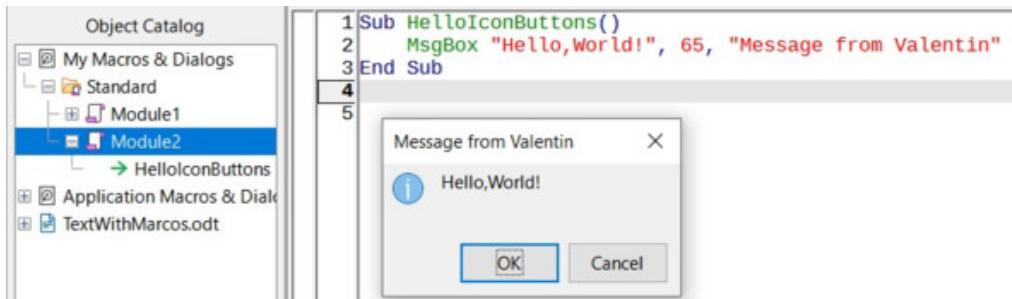


Рис. Заголовок, иконка и кнопочки

**Задание.** Проверьте, как выполняются программы на VBA из предыдущего раздела в Libre Office.

## ИТОГИ

Мы рассмотрели программы-сценарии на языке VBA. Выясняется, что это все тот же Visual Basic, но спрятанный внутри офисного пакета.

С одной стороны, это язык скриптов-сценариев. С другой стороны, это способ записи макро-команд, то есть «макросов». Мы можем записать действия пользователя и вызывать их автоматически.

Можно вначале записать макрос, а затем отредактировать этот сценарий VBA. Может быть полезно для автоматизации повторяющихся, рутинных операций.

## **4. СИСТЕМНЫЕ ВЫЗОВЫ API**

Низкоуровневое программирование  
похоже на танец с системными  
вызовами:  
очень легко споткнуться об ошибки  
(нейросетевой фольклор)

Как происходит выполнение наших программ и как на экране появляется окно? Когда мы создавали окно с сообщением, мы обращались к функциям операционной системы. Мы просто просили создать окно, а всю работу выполняла ОС. Фактически, ОС предоставляет библиотеку готовых процедур, которые мы вызываем из любой прикладной программы. Называется такой подход Application Programming Interface (API) – интерфейс составления прикладных программ. Другой перевод: интерфейс прикладного программирования.

Слова «прикладной» и «приложение» – Application – означает, что программа нужна для работы, а не для обслуживания компьютера, то есть вычислительной системы. «Приложение» в данном случае – это применение, использование компьютера для решения текущих, рабочих задач в какой-либо профессии. Эта работа может заключаться в обработке фотографий, написании рекламных объявлений, монтаже видео или расчете заработной платы. Так вот, для написания таких прикладных программ

разработчик-программист вызывает готовые функции ОС. И такие средства есть практически во всех языках программирования.

Задание. Просмотрите в Википедии статьи API и UI. Выясните, чем отличается API от UI – User Interface.

Задание. Просмотрите в Википедии русский и английский варианты статьи Windows API. Выясните, какие инструменты предоставляет Windows API разработчикам прикладного ПО.

Мы можем обращаться к функциям ОС через API самыми разными способами.

Один из примеров – это командная строка. Рассмотрим команду MSG. Это отправка текстовых сообщений пользователям компьютерной системы. В конечном счете, на экране появляется уже знакомое нам окно. При этом за кадром выполняется множество дополнительных действий.

Чтобы познакомиться с параметрами этой команды, вводим саму команду msg и нажимаем Enter. В описании команда показана как MSG – большими буквами. Напомним, что в командной строке ОС Windows большие и маленькие (заглавные и строчные) буквы не различаются. Так что можно вводить команды msg и MSG с точно таким же результатом. В других ОС будет разница, особенно в Linux и UNIX.

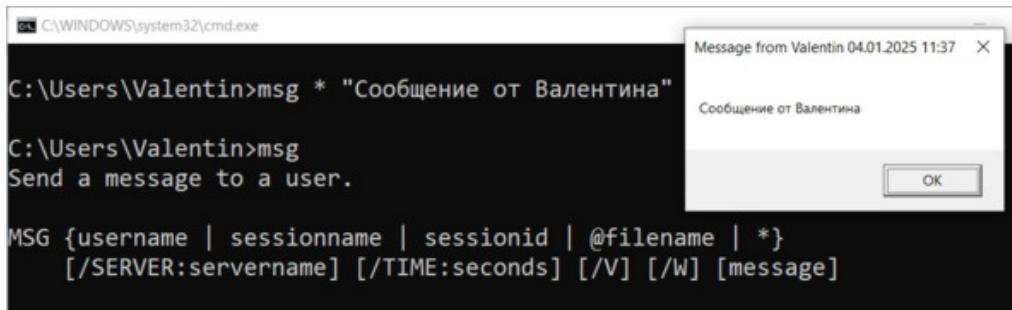


Рис. Отправляем сообщение

Задание. Ознакомьтесь с описанием команды MSG и проверьте, как работают разные её параметры.

Многие функции ОС, доступные по API, хранятся в библиотеках Dynamic Link Library – то файлы типа \*.DLL. Мы даже можем вызывать отдельные функции напрямую из DLL с помощью утилиты RUNDLL.

Задание. Просмотрите в Википедии русскую и английскую версии статьи DLL. Выясните, что общего в чем различие между EXE и DLL.

Многие средства управления ОС можно вызывать не только через меню, но и из командной строки. В следующем примере мы используем утилиту RUNDLL и обращаемся к библиотеке SHELL, чтобы вызвать функцию ShellAbout. В данном случае Shell – это оболочка (интерфейс) ОС, а About намекает на вывод сведений про текущую версию ОС.



Рис. Информация о версии ОС

**Задание.** Выведите сведения об ОС. Проверьте, как работают функции `ShellAboutA` и `ShellAbout`. Выясните, в чем заключается разница между этими функциями и что они на самом деле выводят на экран.

Далее мы рассмотрим примеры обращения к API из традиционного языка программирования. И на этих примерах мы увидим, что API работает одинаково для всех, кто к нему обращается.

## HELLO, С (ЧАСТЬ 1)

В предыдущих разделах мы познакомились с разнообразными скриптами-сценариями, которые выполняются в режиме интерпретатора. Теперь посмотрим, как построена работа на более традиционных языках программирования. Возьмем в качестве примера язык Си. Это очень успешный проект, действует с начала 1970-х годов. Популярный в настоящее время.

Само по себе название «Си» совершенно бессмысленное – это просто третья буква английского алфавита. Это некий намек, что существуют две предыдущие буквы: «Эй» и «Би»...

Задание. Просмотрите на Википедии статьи Си (язык программирования) и С (programming language) и выясните смысл названия языка программирования Си.

Задание. Найдите в интернете рейтинг языков программирования TIOBE Index. Выпишите первую десятку популярных языков.

Задание. Просмотрите на Википедии статью Индекс TIOBE. Выясните, как строится этот рейтинг и что означает это название.

Составим самую простую программу «Hello, World!» на Си.

Для этого будет достаточно простейшего текстового редактора Notepad – Блокнот.

Программа будет простая, стандартная, общеизвестная, см. рис. Найти ее можно в интернете или можно сгенерировать любой нейросетью.



```
hello.c - Notepad
File Edit Format View Help
#include <stdio.h>

int main()
{
    printf("Hello, World!\n");
}
```

Рис. Первая программа на Си

Программу мы написали без проблем. Но просто так ее запустить не получится.

Задание. Составьте программу Hello world на языке Си:

- создайте рабочий каталог для опытов;
- составьте программу в Блокноте;
- сохраните файл в рабочем каталоге.

## КОМПИЛЯТОР

Работа на таких традиционных языках программирования требует использования КОМПИЛЯТОРА. Это программа, которая берет исходный текст программы и превращает его в исполняемый файл. Например, в ОС Windows это будет файл типа \*.EXE – Executable – Исполняемый файл. Полученный файл можно будет запускать на выполнение. И теперь исходный текст программы для запуска уже не нужен. Он будет нужен только для того, чтобы внести изменения в программу.

Как всегда, перед нами английское слово COMPILER, написанное русскими буквами. Глагол TO COMPILE означает «собрать из частей, составить»: COM – «со-» + PILE – «куча, груда; скрести в кучу». Компиляция – это «собирание из готовых частей».

Компилятор берет исходный текст программы, понятный человеку, и как бы «собирает, составляет» программу на машинном языке, понятном процессору. Так что это тоже переводчик с человеческого на машинный язык. Но работает он не торопясь, как переводчик письменных текстов. Сначала мы сформируем исполняемый файл формата \*.EXE, а потом запустим его на выполнение. Этим и отличается компилятор от интерпретатора, который «переводит» на ходу, при выполнении – как синхронный переводчик.

Вот этот процесс компиляции-запуска нам предстоит освоить.

Задание. Составьте программу Hello World. Изучите текст программы и разберитесь с каждой командой и каждым символом. Можете использовать для этого интеллектуальный чат-бот.

Итак, для выполнения программы нам понадобится компилятор. Будем использовать инструмент под названием MinGW. И это пример бесплатного инструмента. Мы будем использовать переносимую версию продукта – Portable Application.

Задание. Просмотрите на Википедии статью MinGW. Выясните, какие компиляторы скрываются внутри этого продукта и что такое GNU.

Задание. Просмотрите на Википедии статьи Переносимое приложение и Portable application. Выясните, в каких ОС можно использовать такую технологию.

Займемся установкой компилятора. Здесь мы описываем шаги установки, которые работали на момент составления данного текста, причем для ОС Windows. Если у вас другая ОС или вдруг ситуация в мире изменится, найдите другой вариант переносимого приложения GCC для своей ОС.

Вводим поисковый запрос – в любой поисковой машине типа Яндекса или Гугла:

mingw windows portable free download

Выбираем в результатах поиска ссылку:

<https://www.mingw-w64.org/downloads/>

Далее выбираем в списке w64devkit и читаем, что это такое:

portable C and C++ development kit for x64 (and x86) Windows.

Перед нами переносимый вариант средств разработки для нашей ОС.

Нас перенаправляют на репозиторий проекта на GitHub.

Нам остается скачать подходящий файл, который представляет собой self-extracting 7-zip archive – самораспаковывающийся архив формата 7-zip.

Внутри этого архива будут лежать исполняемые файлы компилятора. Если взять файл с исходными текстами компилятора, то вначале придется его скомпилировать и только потом использовать. А для этого нужно будет скачать компилятор. В общем, какая-то рекурсия получается. А нам нужно работать! Так что выбираем правильный файл.

Сохраняем его в нужной папке и распаковываем. Можно просто рискнуть и запустить его на выполнение. Но лучше распаковать его архиватором 7-zip.

Конечно, придется этот архиватор предварительно скачать и установить – это делается бесплатно.

Выясняем, что компилятор в виде файлов c++ и g++ лежит в папке bin – то есть binary – двоичные исполняемые файлы. Если в архиве нет папки BIN, значит, это не тот архив. Неправильный. Нужно скачать правильный.

Для дальнейшей работы нас будет интересовать путь к этой папке bin.

Щелкаем левой кнопкой мыши по строке адреса Проводника и видим, как выделяется путь к нашему файлу. Копируем его в буфер.

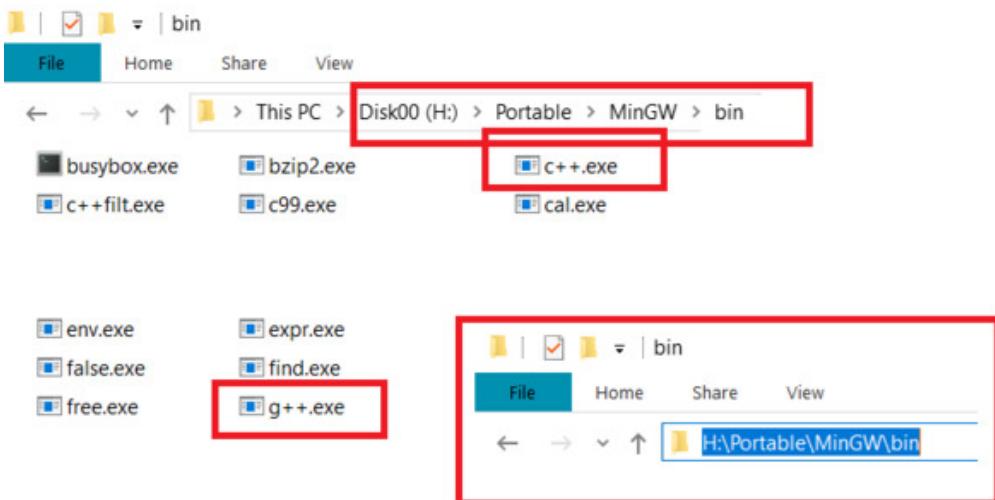


Рис. Расположение компилятора

В командном окне настроим путь для поиска, чтобы запускать компилятор из любого каталога.

```
C:\2>set path=%path%;H:\Portable\MinGW\bin  
C:\2>path  
PATH=C:\Program Files\Microsoft MPI\Bin\;C:\WINDOWS\system32;C:\WI  
stem32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS  
...  
rosoft Office\Office16;H:\Portable\MinGW\bin  
C:\2>
```

Рис. Путь к файлам

Поясним, что мы тут делаем.

Команда SET настраивает переменную окружения PATH. Это настоящая переменная. У нее есть имя – PATH. У нее есть значение – это строчка текста. Хранится она в оперативной памяти компьютера и доступна для любой программы. Программа может прочитать ее значение.

Указываем, что в переменную PATH нужно записать предыдущее ее значение. Чтобы получить значение переменной, мы «обрамляем» имя переменной

символами процента: %PATH%. Затем вставляем сюда путь к каталогу с компилятором – без пробела! Нажимаем клавишу ENTER.

Чтобы вывести значение этой настройки на экран, мы вводим команду PATH.

Можно также ввести команду «эхо»: ECHO %PATH%.

Убеждаемся, что в конце списка каталогов появилась новая папка.

Теперь можно ввести команду c++ или g++ и увидеть, что компьютер находит компилятор. Правда, ему еще нужно указать имя файла с исходным текстом программы.

```
C:\2>c++  
c++: fatal error: no input files  
compilation terminated.  
  
C:\2>g++  
g++: fatal error: no input files  
compilation terminated.
```

Рис. Не ошибается только тот, кто не работает!

Наши настройки переменной PATH будут действовать только на время текущего сеанса и только в одном командном окне. Это удобно для наших опытов. Закрыли командное окно – и никаких последствий для системы. Если что-то пойдет не так, можно закрыть это окно и открыть новое. И можно снова работать, как говорят, «с чистого листа».

Задание. Установите компилятор MinGW:

- скачайте архив;
- создайте папку на рабочем диске;
- разверните компилятор в этой папке;
- откройте командное окно;
- настройте путь для поиска;
- убедитесь, что компилятор запускается.

## **HELLO, С (ЧАСТЬ 2)**

Для работы с нашей программой нужно будет перейти в папку с исходным текстом. Смена текущего каталога (директории, папки) – команда CD – CHANGE DIRECTORY.

Теперь вызываем компилятор и сообщаем ему имя файла с исходным текстом – через пробел после названия компилятора. Если ошибок нет, то никаких сообщений мы не получим. По умолчанию, нам создают исполняемый файл А. EXE. Чтобы запустить его на выполнение, вводим имя файла а. exe или даже просто а без расширения и нажимаем ENTER. Программа выполняется и выводит свое приветствие на экран.

```
C:\2>c++ hello.c  
C:\2>a  
Hello, World!  
  
C:\2>c++ hello.c -o hello.exe  
  
C:\2>hello  
Hello, World!
```

Рис. Компилируем и запускаем

Чтобы задать другое имя выходного файла, добавляем ключ **-o** от слова OUTPUT – выходной файл. Указываем имя исполняемого файла, например, HELLO. EXE. После компиляции такой файл появляется, и его можно запустить на выполнение.

В этих примерах мы вводим в командной строке команду и ее дополнительные параметры (опции) – через пробел. Это просто символы или строки символов, разделенные пробелами. Напомним, что такой прием называется «параметры командной строки» или «аргументы командной строки». Про них у нас уже был разговор в начале этого пособия.

Задание. Составьте и запустите на выполнение программу Hello world на языке Си:

- составьте программу в Блокноте;
- сохраните файл в рабочем каталоге;
- установите компилятор;
- настройте путь для поиска;

- откройте командное окно;
- перейдите в каталог с исходным текстом;
- скомпилируйте программу;
- запустите программу на выполнение.

## СИСТЕМНЫЕ ВЫЗОВЫ

Мы научились выводить приветствие в командной строке. Теперь выведем наш «привет» в окне. Составим вторую программу – Hello World в окне Windows.

Вообще-то само слово *Windows* означает «окна». Мы создаем новое окно в этой «оконной системе», то есть в GUI – Graphical User Interface – графическом интерфейсе пользователя.

В этой работе мы используем два вида интерфейса пользователя: CLI и GUI. Это на сегодняшний день самые распространенные, стандартные интерфейсы для большинства операционных систем и для прикладных программ,

Текст программы приводится на рисунке.

Вначале мы «подключаем» библиотеку стандартных системных вызовов ОС Windows. Мы указываем компилятору, что в текст нашей программы нужно ВКЛЮЧИТЬ – INCLUDE – дополнительные строки. Тогда мы сможем использовать готовую функцию по созданию графического окна на экране.

Такая библиотека называется API – Application Programming Interface – интерфейс для составления прикладных программ. Эту игру слов не так-то просто будет перевести. Programming – это программирование, то есть написание программ. Application – это приложение, то есть прикладная программа. Выражение «прикладное программирование» – это слишком дословный перевод, он не передает основную идею.

Скорее всего, разработчики этого инструмента хотели нам сказать, что предоставляют большую библиотеку «системных вызовов», то есть сборник готовых функций, спрятанных внутри операционной системы. Опять же, SYSTEM CALL – это не просто «системный вызов». Это ВЫЗОВ готовых функций, которые заложены внутри операционной СИСТЕМЫ. Программист, он же разработчик программного обеспечения, он же «разраб», просто выбирает нужную функцию из справочника.

На сегодняшний день название API может относиться к любой библиотеке программ, когда в руководстве для программиста подробно рассказывают, как вызывать из нее готовые функции. Первоначально был один-единственный API – системный набор. Готовые функции для рисования окошек и для других действий с операционной системой. Теперь это название стало популярным. Звучит красиво и загадочно: «эй-пи-ай». Создает впечатление грамотного, опытного специалиста.

В нашем примере мы используем одну готовую функцию MessageBox. MESSAGE – это вывод сообщения.

BOX – это нечто прямоугольное, в нашем случае, это окно на экране. Мы уже вызывали такую функцию в предыдущих разделах. Немного отличалось название. Но смысл тот же. Это окно. Внутри окна текстовое сообщение. В заголовке окна строчка текста. В окне есть иконка и кнопки. Любые обращения из любых программ при создании окна в конечном счете вызывают готовую функцию ОС.



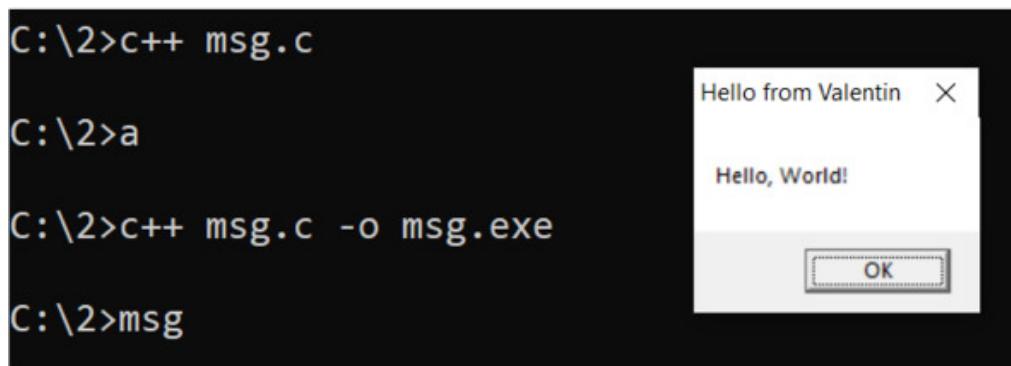
```
msg.c - Notepad
File Edit Format View Help
#include <windows.h>

int main() {
    MessageBox(NULL, TEXT("Hello, World!"), TEXT("Hello from Valentin"), 0);
}
```

Рис. Создаем приветственное окно

Открываем Блокнот и создаем новый файл. Записываем в него свою программу. Сохраняем файл в рабочем каталоге, где мы уже вовсю хозяйничаем. Вызываем компилятор и создаем исполняемый файл. Запускаем его на выполнение и получаем знакомое окно с приветствием.

Пока мы не закроем это новое окно, в командной строке будет «висеть» наша команда по запуску программы.



```
C:\2>c++ msg.c
C:\2>a
C:\2>c++ msg.c -o msg.exe
C:\2>msg
```

Terminal output:

```
Hello from Valentin X
Hello, World!
OK
```

Рис. Компилируем и запускаем

Задание. Создайте и запустите программу для создания окна.

## СКРИПТ И ПРОГРАММА

Чтобы скомпилировать и запустить программу, мы вводили наши команды вручную. Когда мы выполняем одни и те же действия много раз, мы это начинаем замечать. Появляется желание как-то автоматизировать процесс. Поручить его выполнение компьютеру. Сама собой назревает автоматизация повторяющихся, однообразных рутинных действий. Для этого как раз и предназначены скрипты.

Нам нужен скрипт, который скомпилирует нашу программу и запустит ее на выполнение. Мы будем редактировать текст программы в Блокноте, или даже в Notepad++. А затем будем вызывать один и тот же скрипт в командной строке. Это наш очень упрощенный вариант IDE – интегрированной среды разработки программного обеспечения.

Создадим пакетный файл в Блокноте. Для начала нам хватит нескольких строк.

Первая строка начинается с символа «эт коммерческое» @, чтобы не выводить первую команду на экран.

Затем в первой строке мы отключаем вывод остальных команд на экран.

Наш скрипт просто выполняется и ничего не дополнительного нам не сообщает.

Затем мы компилируем нашу программу. Получаем исполняемый файл A. EXE. Запускаем его на выполнение.

Рассматриваем новое окно с приветствием.

Закрываем окно.

Редактируем текст программы в Блокноте, и это окно редактора можно не закрывать. Просто каждый раз сохраняем изменения в файле комбинацией клавиш [Ctrl+S].

Снова вызываем скрипт. И постепенно вносим изменения, наблюдая за работой программы.

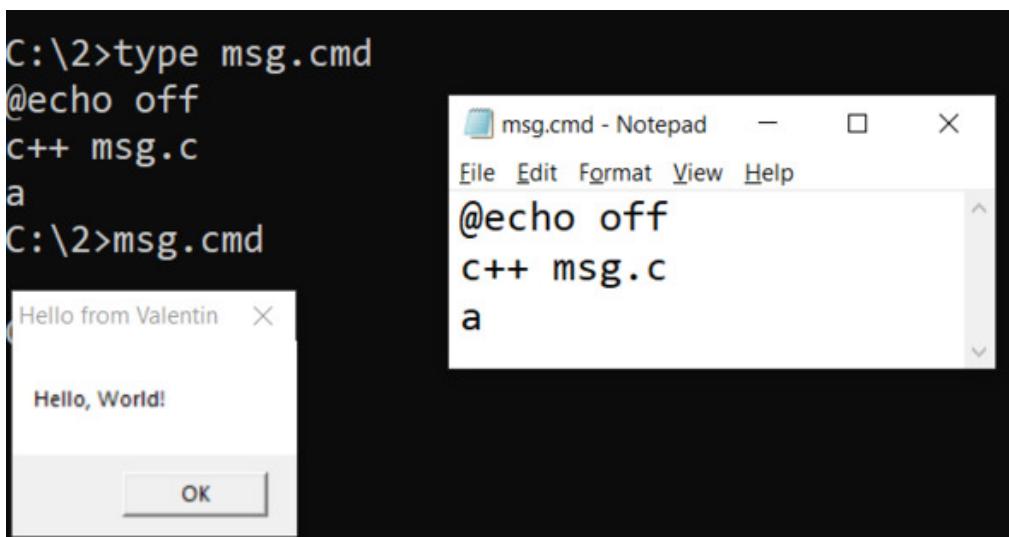


Рис. Скрипт + программа

Задание. Создайте пакетный файл (скрипт) для компиляции и запуска программы.

Теперь, когда мы отладили свой «конвейер» разработки, у нас есть инструмент. Можно заняться исследованиями.

Нас интересуют три момента.

Во-первых, мы помним, что иконку и кнопки можно задавать целым числом или с помощью системных констант. Надо бы посмотреть, как это «хозяйство» устроено и как его использовать.

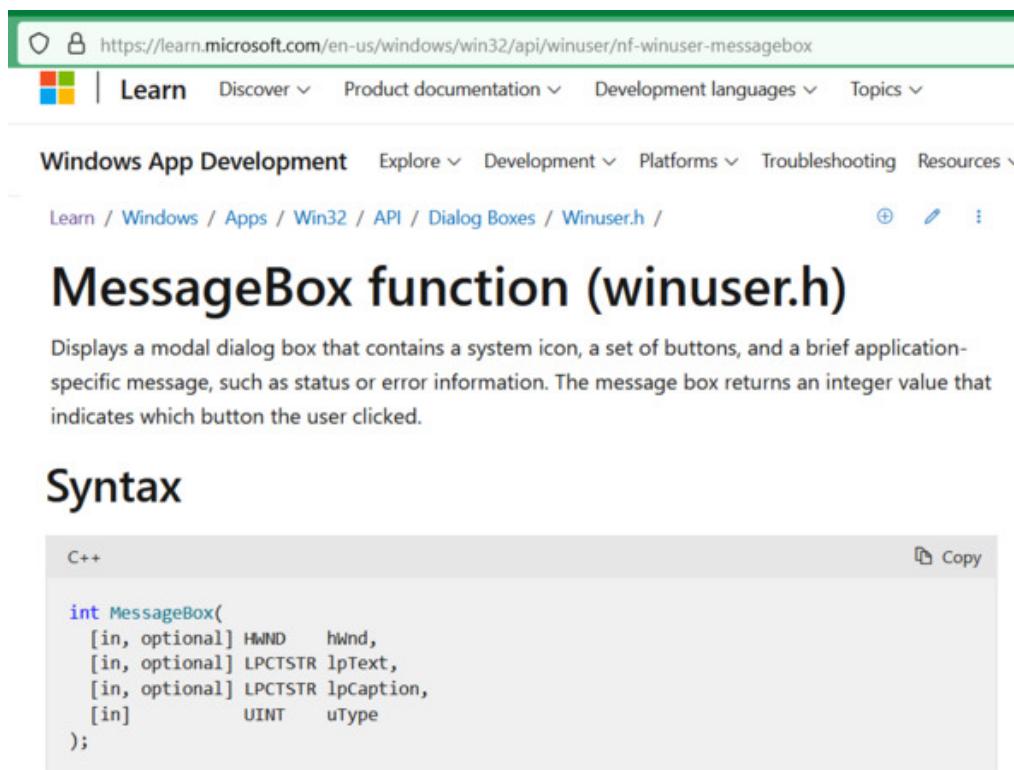
Во-вторых, функция `MessageBox` возвращает целое число, сообщающее код нажатой кнопки. Как его получить и что с ним можно делать?

В-третьих, программа на Си заканчивается строкой `return 0`. Либо мы не пишем эту строчку явным образом. Тогда ее автоматически вставляет компилятор. Это сообщение об успешном завершении программы. А можем сообщить и что-нибудь еще, например, код ошибки. Или в нашем примере – код и название нажатой кнопки. И это «возвращаемое значение» можно прочитать в пакетном файле, который вызывает нашу программу.

Для тех, кто сдавал ЕГЭ по информатике и писал программы на Python, слово `return` должно быть знакомо. Обычно таким способом мы возвращаем результат выполнения функции. В нашем примере это тоже результат – это выполнение функции `main()`. Фактически, это результат работы всей нашей программы. И это значение мы возвращаем той программе, которая вызывала нашу – то есть операционной системе.

Подробное описание функции и всех ее параметров с примерами использования можно найти на сайте корпорации. Достаточно «погуглить» в интернете. Здесь же можно переключиться на русскоязычную страницу руководства – ссылка в левом нижнем углу окна. Перевод будет, скорее всего, машинный. Качество перевода не гарантируется. Так что лучше смотреть

оригинал и при необходимости посматривать на перевод.



The screenshot shows a web browser window with the URL <https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox>. The page is part of the Windows App Development section of the Microsoft Learn site. It displays the C++ syntax for the `MessageBox` function, which is used to display a modal dialog box. The code snippet is as follows:

```
C++  
int MessageBox(  
    [in, optional] HWND     hWnd,  
    [in, optional] LPCTSTR lpText,  
    [in, optional] LPCTSTR lpCaption,  
    [in]          UINT     uType  
>;
```

Рис. Документация по Win API

Задание. Проверьте, как работают знакомые параметры для задания иконок и кнопок. Выясните, какие системные константы для этого существуют и как ими пользоваться.

Задание. Используйте код нажатой кнопки изнутри своей программы. Выведите этот код и название кнопки на экран в командном окне.

Задание. Используйте код возврата программы изнутри пакетного файла, чтобы вернуть код нажатой кнопки. Выведите этот код и название кнопки из пакетного файла на экран в командном окне.

## ИТОГИ

На примере программ на языке Си мы познакомились с компилятором GCC/MinGW и с двумя стандартными интерфейсами пользователя. CLI – это текстовый интерфейс, GUI – это графический интерфейс. Оба интерфейса – это посредники между пользователем и компьютером. Такие интерфейсы обслуживает операционная система.

Затем мы поработали с API – интерфейсом между программами. Одна программа обращается к другой программе. В наших примерах прикладная программа на Си (другие названия – приложение, Application) обращалась к операционной системе, чтобы вывести окно на экран.

Наконец, мы «подружили» программы на компилируемом языке Си и скрипты-сценарии CMD операционной системы. Как оказалось, они могут общаться между собой. Эти инструменты не исключают друг друга, а дополняют возможности своих «коллег».

Вызывая исполняемый файл EXE из скрипта CMD, мы построили миниатюрную среду разработки – IDE. И кое-что попутно автоматизировали.

## 5. POWERSHELL

Оболочка – это одежда для содержимого:

она скрывает его недостатки  
и подчеркивает достоинства  
(нейросетевой фольклор)

Технологии развиваются и совершенствуются. После простой командной разработали графический, оконный интерфейс. Параллельно с «окошками» командная строка продолжает существовать – для сисадминов и для обычных юзеров. Windows PowerShell – это дальнейшее развитие командной строки для управления ОС, или как еще говорят, для системного администрирования. Когда говорят «системное», имеется в виду работа с какой-то «системой», в нашем случае с «операционной системой» или даже с «вычислительной системой». Слово «администрирование» – это красивое название для управления и настройки компьютера. Здесь есть свой специализированный язык сценариев-скриптов для автоматизации повторяющихся действий. Это еще один пример современных скриптов. И мы с этим инструментом немного познакомимся.

Название инструмента PowerShell передает основную идею усовершенствованной командной строки: Power (мощный и удобный) + Shell (командная оболочка ОС, интерфейс командной строки). Разработчики этого программного продукта пишут, что

PowerShell – новое средство решения для автоматизации работ по управлению ОС.

В состав PowerShell входят три основных элемента: консоль (интерпретатор командной строки), язык составления сценариев (скриптов) и среда разработки. Есть также платформа управления конфигурацией – PowerShell Desired State Configuration (DSC). Это инструмент для управления корпоративной компьютерной системой (ИТ-инфраструктурой).

**Задание.** Просмотрите в Википедии статью PowerShell и выясните, в каких ОС работает этот инструмент и на каких условиях.

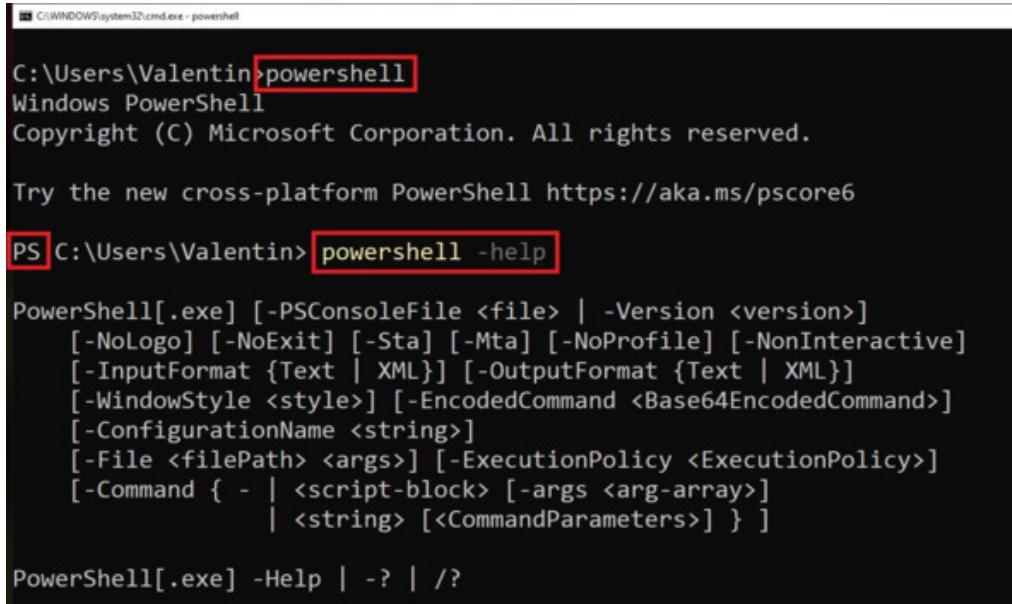
## **С ЧЕГО НАЧАТЬ?**

Работу с любой программой начинаем с запуска. Запускать PowerShell можно разными способами.

Первый вариант – ввести команду PowerShell в обычном командном окне (терминале) cmd. Мы продолжаем работу в том же самом окне, но теперь в начале приглашения к вводу команд дополнительно выводятся буквы PS. Попутно нам сообщают, что этот инструмент стал кросс-платформенным (и дают интернет-ссылку). То есть он работает не только под MS Windows.

Вводим первую команду. Вызываем справку и ознакомимся с параметрами. Это параметр (ключ, опция) -help.

Как и ранее, в квадратных скобках указывают необязательные аргументы. Вертикальная черта разделяет варианты параметров. Это аналог логической операции ИЛИ.



```
C:\Users\Valentin>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Valentin> powershell -help

PowerShell[.exe] [-PSConsoleFile <file> | -Version <version>]
    [-NoLogo] [-NoExit] [-Sta] [-Mta] [-NoProfile] [-NonInteractive]
    [-InputFormat {Text | XML}] [-OutputFormat {Text | XML}]
    [-WindowStyle <style>] [-EncodedCommand <Base64EncodedCommand>]
    [-ConfigurationName <string>]
    [-File <filePath> <args>] [-ExecutionPolicy <ExecutionPolicy>]
    [-Command { - | <script-block> [-args <arg-array>]
        | <string> [<CommandParameters>] } ]

PowerShell[.exe] -Help | -? | /?
```

Рис. Запускаем PowerShell в терминале CMD

Задание. Запустите PowerShell в командном окне CMD. Вызовите справку. Ознакомьтесь с описанием кросс-платформенной версии PowerShell и выясните, на каких plataформах работает это средство.

Задание. Обратите внимание на цветовую подсветку синтаксиса в консоли PowerShell.

Просмотрите в Википедии статью Подсветка синтаксиса. Выясните, какие элементы команд (программ) выделяют цветом и для чего.

Задание. Вызовите справку PowerShell.

Второй способ запустить PowerShell – это меню Пуск. Здесь можно пролистать список программ либо просто ввести Power в строке поиска. Можно запустить программу с правами текущего пользователя либо

с правами администратора – если нужно будет вносить изменения в настройки операционной системы.

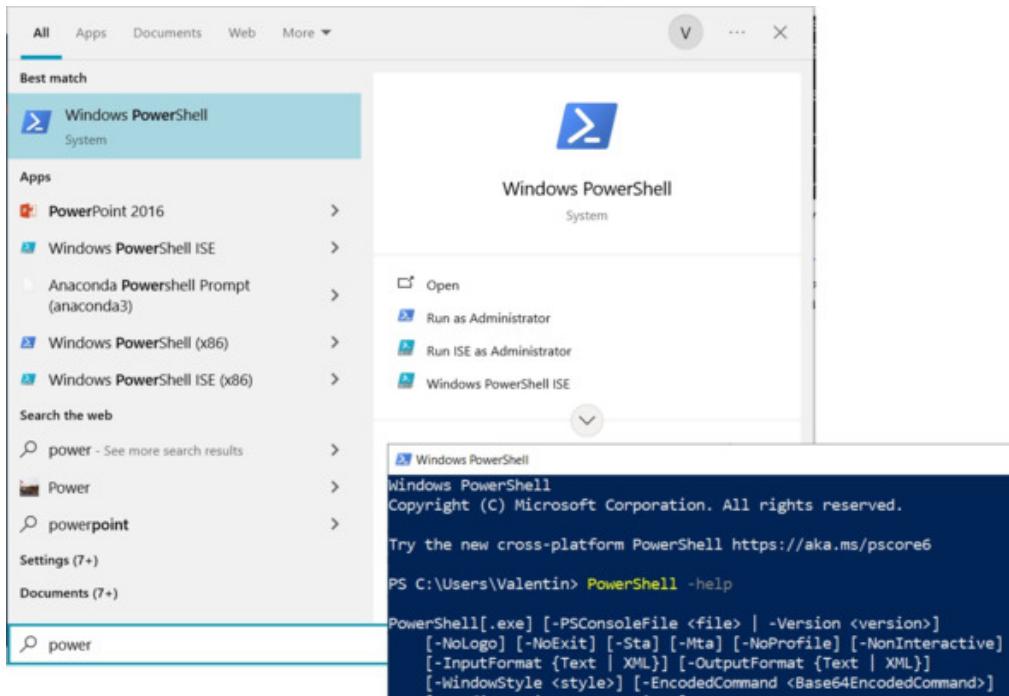


Рис. Запускаем PowerShell через меню Пуск

Обратим внимание на иконку программы – парочка символов на темно-синем фоне. Здесь можно легко узнать приглашение к вводу команд> и символ нижнего подчеркивания \_, который обычно показывает текущее положение курсора при вводе команды. Синий фон PowerShell подчеркивает отличие от привычного командного окна с черным фоном.

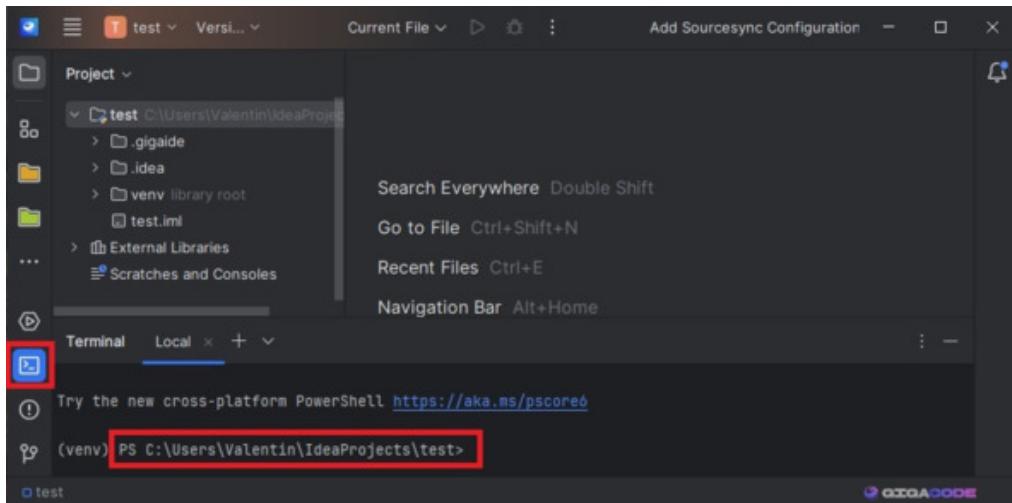


Рис. Терминал в GigaCode

Похожую иконку [>] можно увидеть во многих средах разработки типа PyCharm, VSCode, GigaCode. Это переход в терминал. То есть в командную строку. Пример приводится на рисунке.

Задание. Запустите PowerShell через меню Пуск и вызовите справку. Обратите внимание на количество доступных настроек.

При поиске Power в меню Пуск нам также предлагают PowerShell ISE – и это будет третий способ запуска PowerShell. Integrated Scripting Environment (ISE) – это интегрированная среда разработки скриптов-сценариев. Ее тоже можно запускать от имени пользователя и от имени администратора.

## КАК СМЕНИТЬ КАТАЛОГ?

Рассмотрим вывод на экран имени текущего каталога.

В командном окне CMD нам понадобится команда CD – Change Directory. Обычно это смена текущего каталога (директории, папки). Но в ОС Windows эта команда без параметров просто выводит на экран текущий каталог и ничего не меняет.

В окне PowerShell нам понадобится другой инструмент под названием Get-Location. Расшифруем название: Get (получить) + Location (текущее расположение). Это не просто команда, это cmdlet – «командлет».

Командлет – это небольшая программа для решения определенной задачи. Cmd – command – «команда», а суффикс -let придает уменьшительный оттенок существительному, по-русски это будет “-чка». Если переводить commandlet дословно и буквально, получается не просто «команда», а «командочка». Так образованы некоторые знакомые слова иностранного происхождения, например, book (книга) и booklet (книжечка, буклете). Так и получается cmdlet = «небольшая команда». Кроме красивого названия, здесь другого смысла нет. Надо же как-то отличаться от конкурентов! Обычные команды ОС тоже небольшие. Зато у команд PowerShell много дополнительных параметров, и в результате строчки получаются довольно длинными. В нашем учебном пособии мы стараемся не усложнять, а упрощать, поэтому для нас это тоже будут «команды». Но если захочется проявить компьютерную грамотность на собеседовании, конечно, надо гордо говорить «командлет» с загадочным выражением на лице.

Названия командлетов построены по схеме: Verb-Noun (например, Get-Location). Вначале написан глагол (он называет действие), затем через черточку

существительное (объект, над которым надо выполнить действие).

Итак, мы запускаем интегрированную среду ISE, и она должна нам помочь в работе. Начинаем вводить команду Get-L... Сразу появляется выпадающий список для выбора. Выбираем Get-Location и видим всплывающую подсказку с основными параметрами команды. В правой части окна ISE можно ввести начало названия команды в строке поиска Name, выбрать команду из предложенного списка и нажать Insert – Вставить. Далее можно указать дополнительные параметры либо просто нажать Enter и запустить команду на выполнение.

В результате мы получаем полный путь к текущему каталогу – с поясняющим заголовком Path – Путь.

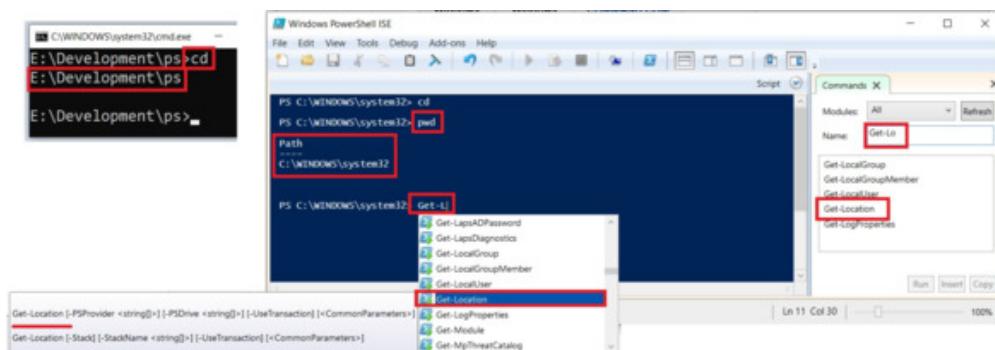


Рис. Интегрированная среда PowerShell ISE

Можно также вызвать сокращенный «псевдоним» (alias) pwd – с тем же результатом. В качестве псевдонима взяли команду из UNIX и Linux: print working directory – напечатать на экране текущую директорию (каталог, папку). И это уже намек на то, что PowerShell работает и в Windows, и в Linux.

Задание. Запустите PowerShell ISE и выведите на экран текущий каталог.

## ПОГОВОРИМ О ПОЛИТИКЕ

В консоли можно запускать отдельные команды и целые сценарии (скрипты). Чтобы обеспечить приемлемый уровень безопасности, в PowerShell настроена «политика выполнения». В ОС «политика» – это набор правил для работы, например, «пароль должен быть не короче 8 символов».

В PowerShell политика запуска скриптов защищает нас как пользователей и администраторов от случайного запуска вредоносных скриптов. К сожалению, не все скрипты «полезны для здоровья». Особенно с правами администратора.

Чтобы узнать про текущую политику, вводим команду Get-ExecutionPolicy.

Описание команды выводим с помощью Get-Help (Get – получить, вывести на экран, Help – помощь, поддержка, описание команды):

Get-Help Get-ExecutionPolicy

Попутно нам могут предложить обновить справочную систему. Не будем возражать.

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Get-ExecutionPolicy
Restricted
PS C:\WINDOWS\system32> Get-Help Get-ExecutionPolicy
Do you want to run Update-Help?
The Update-Help cmdlet downloads the most current Help files for Windows PowerSh
computer. For more information about the Update-Help cmdlet, see https:/go.micro
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
NAME
    Get-ExecutionPolicy
SYNOPSIS
    Gets the execution policies for the current session.
SYNTAX
    Get-ExecutionPolicy [[-Scope] {CurrentUser | LocalMachine | MachinePolicy |
```

Рис. Политика выполнения скриптов

Краткое описание команды – Synopsis – сообщает, что мы получаем политику для текущего сеанса работы – current session. В нашем примере она звучит как Restricted.

Выясняем, что это означает и какие еще политики бывают. В этом нам поможет Поиск с нейросетью от Яндекса.

The screenshot shows a search results page from Yandex. The search bar at the top contains the query 'Get-ExecutionPolicy'. Below the search bar, there is a navigation menu with tabs: 'поиск с нейро' (selected), 'картинки', 'видео', 'карты', 'товары', 'переводчик', and 'все'. The main content area features a red circular icon with a white 'N' and the word 'Нейро'. Below it, a note says 'На основе источников, возможны неточности'. A large text block explains how to get the current execution policy value in PowerShell: 'Чтобы получить текущее значение политики выполнения скриптов PowerShell на компьютере, нужно ввести команду: Get-ExecutionPolicy.' It includes two numbered links: '1' and '2'. Another section titled 'Некоторые возможные значения политики:' lists a bullet point: '• Restricted — запрещён запуск скриптов PowerShell, можно выполнять только интерактивные команды в консоли.' There is also a link '1' next to this point.

Рис. Опасные действия запрещены

Задание. Введите в консоли команды `Get-ExecutionPolicy` и `Get-Help Get-ExecutionPolicy`. Выясните, какая политика действует в рамках текущего сеанса работы и какие еще варианты политики существуют.

Справочная система PowerShell может вывести на экран примеры использования команды с подробными пояснениями:

`Get-ExecutionPolicy -List`

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Get-Help Get-ExecutionPolicy -Examples

NAME
    Get-ExecutionPolicy

SYNOPSIS
    Gets the execution policies for the current session.

----- Example 1: Get all execution policies -----
Get-ExecutionPolicy -List

Scope      ExecutionPolicy
-----
MachinePolicy  Undefined
UserPolicy    Undefined
Process      Undefined
CurrentUser   AllSigned
LocalMachine  Undefined
```

Рис. Примеры и пояснения

Можно даже просмотреть описание нашей команды в интернете:

`Get-Help Get-ExecutionPolicy -Online`

Как видим, в браузере открывается описание для текущей версии PowerShell на сервисе MS Learn.

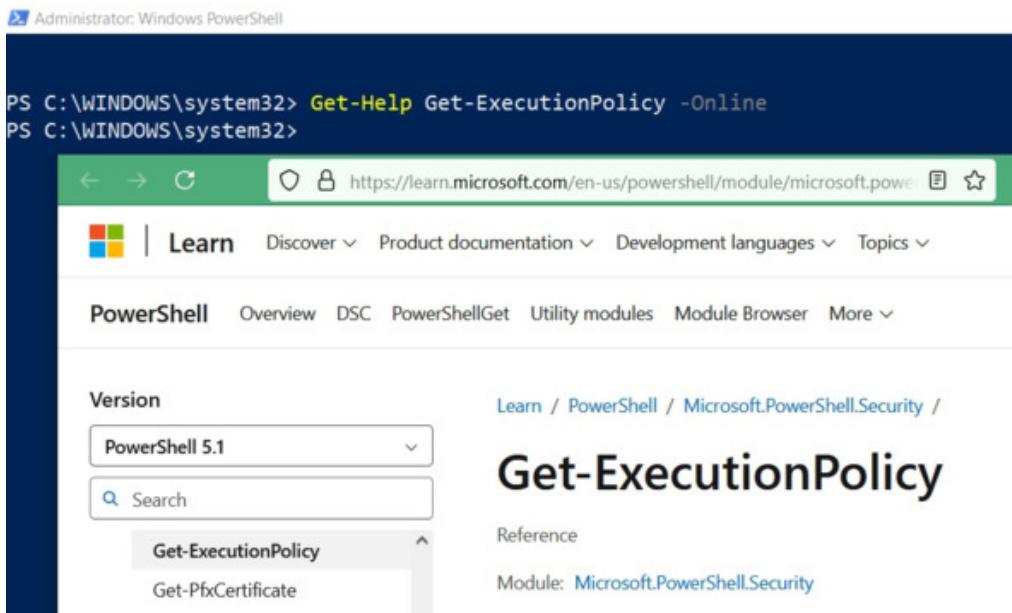
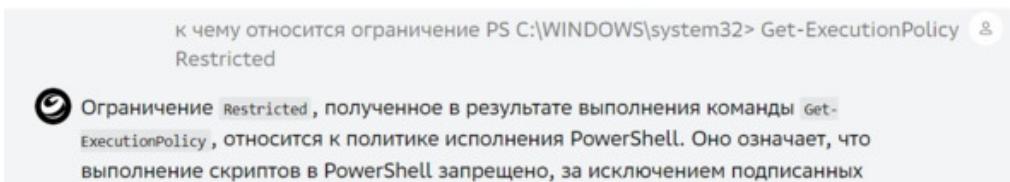


Рис. Онлайн справка для команды

Задание. Вызовите справку для Get-ExecutionPolicy в консоли и в интернете. Проверьте, как работает первый пример – опция вывода всех политик – Get all execution policies.

Чтобы перевести на человеческий язык сообщения PowerShell, нам понадобится очень интеллектуальный помощник. С той задачей хорошо справляется Гигачат. Мы отправляем ему нашу команду + сообщение компьютера и просим объяснить. Чат-бот объяснит и текст, и снимок экрана. Можно дополнительно попросить дать объяснения «простыми словами», или «шаг за шагом», или «подробно».



Задание. С помощью интеллектуального чат-бота изучите текущие ограничения по запуску скриптов на вашем компьютере.

Глаголы в названии команды указывают, какое действие нужно выполнить:

- Get – Получить
- Set – Задать, изменить
- New – Создать
- Add – Добавить
- Remove – Удалить

Задание. Просмотрите список команд в панели Commands и составьте список основных действий.

## ТАБУЛЯЦИЯ УСКОРЯЕТ

Параметры команды начинают с символа тире -. Затем через пробел указывают значение. Если значение включает пробелы, этот текст заключают в кавычки. Несколько значений для одного параметра записывают через запятую, без пробелов.

Для ускорения ввода команд здесь используют клавишу TAB. Вводим первые буквы команды в ISE или консоли и нажимаем TAB. Если есть несколько команд

с одинаковым началом, нажимаем TAB несколько раз – проходим список «по кругу». Этот инструмент особенно полезен, если нужно ввести длинную команду.

Для перебора параметров выбранной команды вводим дефис – и нажимаем TAB.

Чтобы «прокрутить» список в обратную сторону, нажимаем [Shift + TAB].

**Задание.** Введите начало команды Get-  
E и нажимайте TAB, чтобы найти Get-ExecutionPolicy. Когда появилась нужная команда, нажмите пробел и дефис, затем нажимайте TAB и просмотрите все возможные параметры. Нажмайте [Shift + TAB] и «прокрутите» список обратно.

## **КТО СКРЫВАЕТСЯ ЗА ПСЕВДОНИМОМ**

Познакомимся с псевдонимами поближе. В PowerShell есть много псевдонимов для популярных команд. Это позволяет вводить «знакомые команды». Как мы уже обсуждали, в Линуксе и Юниксе есть команда PWD – Print Working Directory – вывести на экран имя текущей директории (рабочего каталога). В Windows для этого понадобится команда CD без параметров.

Вообще-то CD означает Change Directory – сменить текущую директорию (каталог). Но в командной строке Windows команда CD без указания нового каталога просто сообщает нам, где именно мы находимся.

Напомним, синее окно – PowerShell, черное окно – CMD.

PS	CMD
<pre>PS C:\Users\Valentin&gt; Get-Location  Path ---- C:\Users\Valentin</pre>	<pre>C:\Users\Valentin&gt;cd C:\Users\Valentin  C:\Users\Valentin&gt;pwd C:/Users/Valentin</pre>
<pre>PS C:\Users\Valentin&gt; pwd  Path ---- C:\Users\Valentin</pre>	

Рис. Текущая директория

Как получить список всех псевдонимов? Используем формулу «глагол-существительное». «Получить что-нибудь» – это Get. «Псевдоним» – это Alias. Значит, нам нужна команда Get-Alias. Так мы получаем полный список. Где-то в середине списка можно найти нашу строчку: pwd -> Get-Location.

Тот же самый ответ можно получить в виде списка содержимого виртуального диска Alias::

```
PS C:\Users\Valentin> Get-Alias

 CommandType      Name
 -----          ---
 Alias           % -> ForEach-Object
 Alias           ? -> Where-Object

Alias          pushd -> Push-Location
Alias          pwd -> Get-Location
Alias          r -> Invoke-History

PS C:\Users\Valentin> ls Alias:\

 CommandType      Name
 -----          ---
 Alias           % -> ForEach-Object
 Alias           ? -> Where-Object
```

Рис. Псевдонимы

Мы уже знаем, что нас интересует Get-Location. Мы можем получить список всех псевдонимов для нее. Для этого используем параметр Definition.

Есть и обратное действие: по псевдониму найти команду – нужен будет параметр Name.

```
PS C:\Users\Valentin> Get-Alias -Definition Get-Location
 CommandType      Name
 -----          -----
 Alias           gl -> Get-Location
 Alias           pwd -> Get-Location

PS C:\Users\Valentin> Get-Alias -Name pwd
 CommandType      Name
 -----          -----
 Alias           pwd -> Get-Location
```

Рис. Псевдоним и его команда

Мы даже можем создавать свои псевдонимы Alias, изменять и удалять их. Для этого используют ключевые слова, например, New. Получаем New-Alias. Получается, что прилагательное New нарушает общее правило Verb-Noun. За этим скрывается целая история создания программного продукта, с которой можно познакомиться самостоятельно.

Попробуйте получить все команды для работы в псевдонимами:

Get-Alias -Definition \*-Alias

Однако, здесь нет команды для удаления! Найдите решение проблемы.

Задание. Попрактикуйтесь в создании и удалении псевдонимов. Выясните, почему здесь использовано прилагательное New вместо глагола Create.

## ОБЪЕКТЫ ИЛИ ТЕКСТ

Как мы с вами увидели в разделе про командную строку CMD, на входе и выходе команд используется обычный текст. Одна или несколько строк текста. Последовательность символов. В общем случае, байтов. И если нас интересует что-то конкретное, нам придется вручную искать нужную часть строки.

Разработчики PowerShell пошли дальше. Теперь на входе и выходе командлетов мы имеем объекты. Это особое понятие из объектно-ориентированного программирования (ООП). Здесь вся полезная информация разложена по полочкам и подготовлена для использования. Больше не нужно заниматься особо сложным поиском по тексту и фильтрацией результатов.

Можно сказать, что «объект» в программировании – это, как говорят программисты, «контейнер». Или своеобразная «коробочка». А внутри коробочки хранятся разные полезные и бесполезные вещи. У каждого объекта есть свойства и методы. В привычном понимании это соответствует понятиям «переменные» и «функции».

Свойство (Property) можно сравнить с конкретным значением переменной, которая хранится внутри объекта. Это значение можно посмотреть. Можно даже это значение изменить – если нам разрешили это делать. Это данные любого типа: число, или символ, или строка текста, или логическое значение Истина/Ложь.

Метод (Method) – это функция, которая работает с данными, которые хранятся внутри объекта. Как работать с методами, мы скоро увидим.

Задание. Попросите какой-нибудь чат-бот объяснить простыми словами, что такое свойства и методы объекта

в ООП. Пусть приведет простые примеры из жизни — для аналогии.

## СПИСОК ФАЙЛОВ

Чтобы поработать с объектами, разберем простой пример — получение списка файлов в текущем каталоге. В привычном командном окне это команда DIR (в Windows) или LS (в Линуксе). Переходим в нужный каталог, вызываем команду и видим, что у нас имеется пара файлов. Можно вытащить список названий файлов без подробностей. Для этого у нас есть опция /b — попробуйте выяснить, какое английское слово здесь зашифровано.

```
C:\Users\Valentin>cd /d e:\de*\sc*
e:\Development\Scripts>dir
Volume in drive E is Disk1
Volume Serial Number is 64E0-88D7

Directory of e:\Development\Scripts

20.01.2025  14:39    <DIR>      .
20.01.2025  14:39    <DIR>      ..
20.01.2025  14:18            37 x.bat
20.01.2025  14:39            36 z.cmd
                           2 File(s)   73 bytes
                           2 Dir(s)  1 998 647 808 000 bytes free

e:\Development\Scripts>dir /b
x.bat
z.cmd
```

Рис. Список файлов DIR

Теперь получим тот же список в PowerShell.

Наши действия следующие:

`Set-Location` – перейти в нужный каталог

`ls` – вывести список файлов – используем псевдоним

`Get-Alias` – узнать имя команды – находим оригинал по псевдониму

`Get-ChildItem` – получить список файлов

Так мы узнали настоящее имя нужной нам команды.

Команда `Get-ChildItem` получает дочерние элементы (файлы и подпапки) внутри текущего каталога. Можно сказать, что текущий каталог считается «родителем», а его содержимое – «потомками».

```
PS C:\Users\Valentin> Set-Location E:\Development\Scripts
PS E:\Development\Scripts> ls
    Directory: E:\Development\Scripts

Mode                LastWriteTime         Length Name
----                -----              ----- 
-a----       20.01.2025      14:18            37 x.bat

PS E:\Development\Scripts> Get-Alias -name ls
 CommandType      Name
-----          ----
 Alias           ls -> Get-ChildItem

PS E:\Development\Scripts> Get-ChildItem
    Directory: E:\Development\Scripts

Mode                LastWriteTime         Length Name
----                -----              ----- 
-a----       20.01.2025      14:18            37 x.bat
-a----       20.01.2025      14:39            36 z.cmd
```

Рис. Список файлов `Get-ChildItem`

Первое впечатление, что перед нами текст. Текст содержит список файлов. На самом деле, текст на экране – это только часть информации из объекта.

Посмотрим, что у него внутри – у полученного объекта. Организуем небольшой конвейер с помощью вертикальной черты. Еще раз вызываем нашу команду (командлет) Get-ChildItem. Но теперь полученный объект мы передаем для анализа на вход команде Get-Member. На выходе получаем список содержимого нашего объекта, нашего «контейнера».

```
PS E:\Development\Scripts> Get-ChildItem | Get-Member
 TypeName: System.IO.FileInfo

Name             MemberType      Definition
----             -----          -----
Extension       Property        string Extension {get;}
FullName        Property        string FullName {get;}
Name            Property        string Name {get;}

PS E:\Development\Scripts> (Get-ChildItem).FullName
E:\Development\Scripts\x.bat
E:\Development\Scripts\z.cmd
PS E:\Development\Scripts> (Get-ChildItem).Name
x.bat
z.cmd
PS E:\Development\Scripts> (Get-ChildItem).Extension
.bat
.cmd
PS E:\Development\Scripts> (Get-ChildItem).FullName
E:\Development\Scripts\x.bat
E:\Development\Scripts\z.cmd
```

Рис. Свойства объекта

Список большой. В нем есть и методы, и свойства. Выведем на экран несколько свойств нашего объекта. Чтобы обратиться к свойству объекта, используем круглые скобки и точку. Скобки нужны, потому что команда может содержать параметры. А мы используем результаты работы команды с параметрами, то есть объект, полученный при выполнении команды.

FullName – это полные пути к файлам: диск, каталоги, имя файла.

Name – только имя файла.

Extension – расширение имени файла – буквы после точки. Обычно эти буквы намекают на тип файла. Как видим, здесь выводится точка и три буквы после нее.

Задание. Ознакомьтесь с несколькими свойствами списка файлов.

В свою очередь, каждое полученное свойство – это тоже объект. Для него тоже можно получить список методов и свойств. Находим свойство Length. Мы можем получить длину списка файлов.

Полученный ответ тоже является объектом. И эту «матрешку» можно продолжать «раскрывать», пока не надоест.

```
PS E:\Development\Scripts> (Get-ChildItem).Name | Get-Member

TypeName: System.String

Name          MemberType      Definition
----          -----          -----
Length        Property       int Length {get;}

PS E:\Development\Scripts> ((Get-ChildItem).Name).Length
2
```

Рис. Длина списка файлов

Задание. Пройдите по цепочке свойств объектов на несколько шагов «в глубину».

Мы получили список длиной 2 элемента. Теперь из этого списка можно извлечь элемент с нужным номером (индексом). Для этого нам понадобится Select-Object с указанием индекса. Получается, что при выборе элемента списка мы снова получаем объект.

```
PS E:\Development\Scripts> (Get-ChildItem).Name | Select-Object -Index 0
x.bat
PS E:\Development\Scripts> (Get-ChildItem).Name | Select-Object -Index 1
z.cmd
PS E:\Development\Scripts> (Get-ChildItem).Name | Select-Object -Index 2
PS E:\Development\Scripts>
```

Рис. Выбор объекта

Обратим внимание, что нумерация начинается с нуля. Наши объекты имеют номера 0 и 1. Третьего элемента с индексом 2 нет, поэтому на экран ничего не выводят.

Нумерация с нуля – это удобство для программистов. Индексы хранятся в переменной типа «целое без знака». Такие переменные принимают значения 0,1,2,... и т. д.

Нормальным людям все-таки привычнее считать от единицы. Как вы, наверное, знаете, до недавнего времени в математике даже не было понятия «ноль». Числа начинались с единицы. Их называли «натуральные». Слово *natura* означает «природа», а в природе не бывает «ноль студентов». Потому что трудно отличить «ноль студентов» от «ноль преподавателей». Мы просто скажем: «никого нет», не уточняя кого именно.

Задание. Перейдите в каталог с большим количеством файлов. Выберите любой файл и получите

сведения о этом файле по его номеру.

## РАБОТА С ФАЙЛОМ

Мы получили список файлов. Теперь поработаем с одним отдельно взятым файлом.

Для доступа к файлу используем Get-Item. Указываем имя файла и просматриваем список методов и свойств.

PS E:\Development\Scripts> <b>Get-Item</b> -Path x.bat   <b>Get-Member</b>		
Name	MemberType	Definition
Delete	Method	void Delete()
Attributes	Property	System.IO.FileAttributes Attributes {get;set;}
CreationTime	Property	datetime CreationTime {get;set;}
FullName	Property	string FullName {get;}
IsReadOnly	Property	bool IsReadOnly {get;set;}
Length	Property	long Length {get;}
Name	Property	string Name {get;}

Рис. Содержимое объекта «файл»

Выведем на экран некоторые свойства нашего файла.

Как видим, у него имеется атрибут Archive – «Архивный». Название, как это часто бывает, обманчивое. Имеется в виду, что это файл не был скопирован (заархивирован) в резервную копию ОС. И поэтому при очередном резервном копировании его нужно будет добавить в этот «архив».

CreationTime – время создания файла (включая дату) совпадает с предыдущими сведениями.

FullName – Полное имя файла, то есть полный путь к файлу: диск-каталоги-файл.

IsReadOnly – Установлен ли для этого файла атрибут «только для чтения»? Нет.

Length – длина, то есть размер файла в байтах – совпадает.

Name – имя файла.

```
PS E:\Development\Scripts> (Get-Item -Path x.bat).Attributes
Archive
PS E:\Development\Scripts> (Get-Item -Path x.bat).CreationTime
20 января 2025 г. 14:17:52

PS E:\Development\Scripts> (Get-Item -Path x.bat).FullName
E:\Development\Scripts\x.bat
PS E:\Development\Scripts> (Get-Item -Path x.bat).IsReadOnly
False
PS E:\Development\Scripts> (Get-Item -Path x.bat).Length
37
PS E:\Development\Scripts> (Get-Item -Path x.bat).Name
x.bat
```

Рис. Свойства объекта t

Рассмотрим атрибуты нашего файла. Запускаем Проводник: [Win + E]. Здесь Е – первая буква слова Explorer – Проводник. Переходим в наш каталог. Выбираем вывод списка файлов в виде таблицы с подробными сведениями: View – Details. Затем добавляем отображение столбца атрибутов: View – Current view – Add columns – Choose column – Attributes.

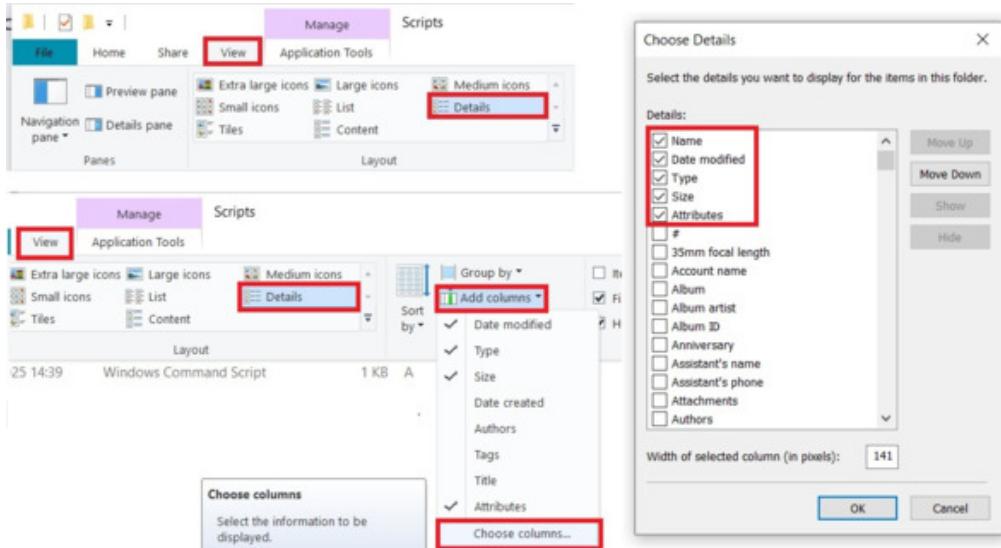


Рис. Атрибуты файла

И снова перед нами «странные названия». Properties – Свойства и Attributes – Атрибуты. В других ОС то же самое могут называть Permissions – Разрешения. Или Access rights – Права доступа. Или еще как-нибудь. Даже в разных версиях одной и той же ОС будут разные названия для одного и того же.

В любом случае для нашего файла мы находим или разрешения, или свойства. Речь идет про действия, которые можно выполнять над файлом, или про сведения о файле. К примеру, владелец файла разрешает или запрещает нам редактировать, изменять его содержимое. Это можно назвать разрешением. А еще у нас есть дата создания файла. Это просто сведения о файле.

В колонке Атрибуты можно будет увидеть разные буквы. Эти буквы – начало соответствующих английских слов. Подробнее можно с ними познакомиться и поуправлять разрешениями в командном окне. Для просмотра и установки атрибутов есть команда attrib. Она доступна и в CMD, и в PowerShell.

```
E:\Development\Scripts>attrib /?
Displays or changes file attributes.

ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [+O | -O] [+I | -I]
[+X | -X] [+P | -P] [+U | -U]
[drive:][path][filename] [/S [/D]] [/L]

+ Sets an attribute.
- Clears an attribute.
R Read-only file attribute.
A Archive file attribute.
```

```
E:\Development\Scripts>attrib x.bat
A E:\Development\Scripts\x.bat

E:\Development\Scripts>attrib +R +H +S x.bat

E:\Development\Scripts>attrib x.bat
A SHR E:\Development\Scripts\x.bat
```

Name	Date modified	Type	Size	Attributes
x.bat	20.01.2025 14:18	Windows Batch File	1 KB	RHSA
z.cmd	20.01.2025 14:39	Windows Command Script	1 KB	A

Рис. Управляем атрибутами в CMD

Некоторыми «буквами» можно управлять и в контекстном меню Проводника: наводим курсор на иконку файла, нажимаем правую кнопку мыши (ПКМ) и выбираем Properties – Attributes, затем Advanced.

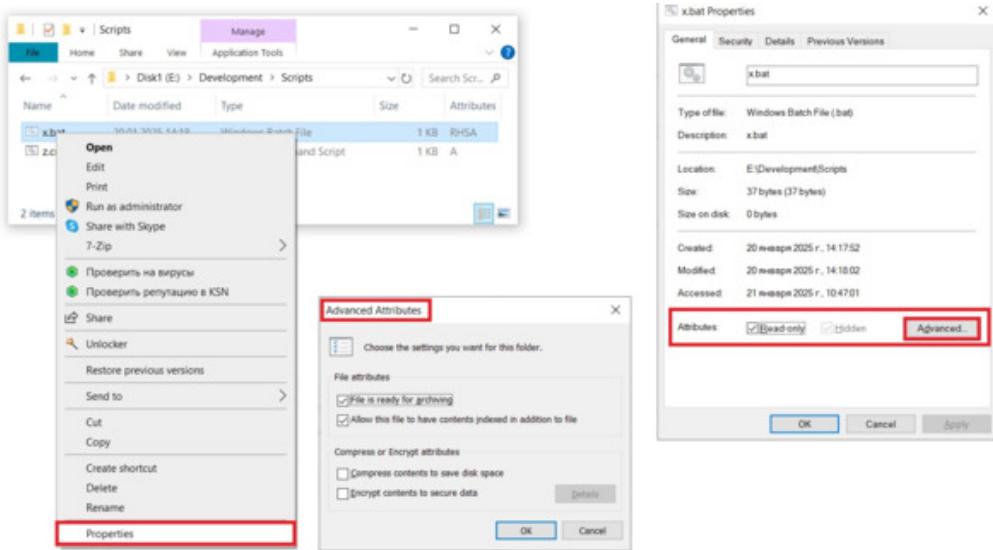


Рис. Управляем атрибутами в Проводнике

Скрытые файлы можно будет увидеть в Проводнике, но не сразу.

Нужно выполнить некоторые настройки:

View – Show/Hide – File name extensions + Hidden items

View – Options – Change folder and search options – Folder options – View – Advanced settings – Show hidden files / Hide extensions for known file types / Hide protected operating system files.

В один случаях мы ставим галочки, в других – снимаем.

Нажимаем Apply to folders и OK.

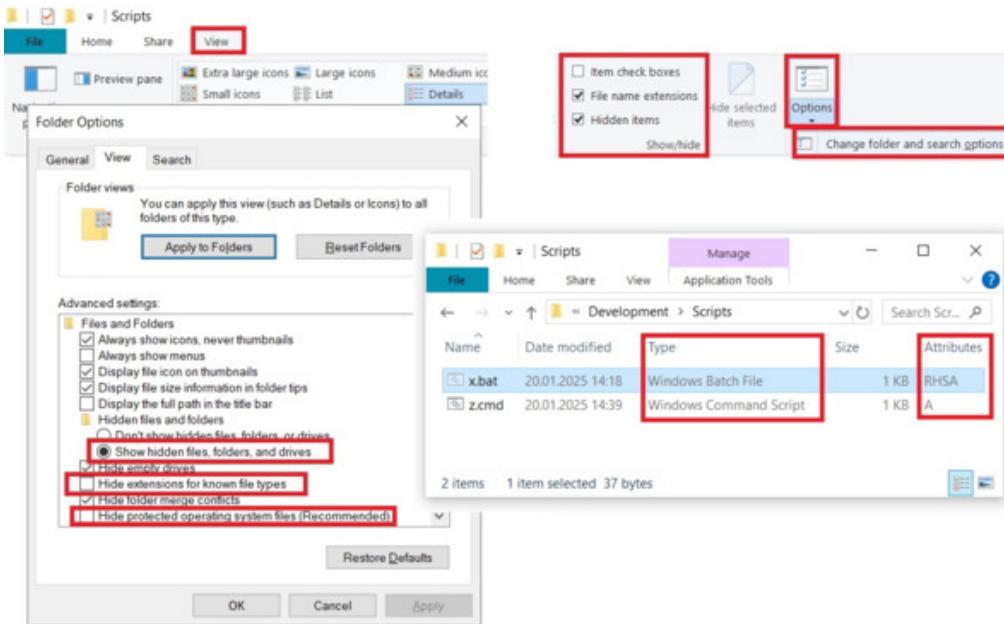


Рис. Настроим Проводник

Мы снова наблюдаем оба наших файла.

В колонке Attributes указаны загадочные буквы. Мы можем частично управлять этими атрибутами в Проводнике, и частично – в командной строке.

Обратим внимание на тип. В зависимости от расширения имени получаем:

\*.BAT – Windows Batch File

\*.CMD – Windows Command Script

При этом внутри оба файла могут быть совершенно одинаковыми, и работать они могут в простейшем случае одинаково.

Задание. Выясните различие файлов \*.BAT и \*.CMD.

Выводим на экран список файлов: Get-ChildItem. Один файл пропал.

Чтобы вывести на экран список всех файлов, нужен параметр -Force.

Атрибуты выводятся в колонке Mode – Режим доступа.

```
PS E:\Development\Scripts> Get-ChildItem  
Directory: E:\Development\Scripts  


| Mode   | LastWriteTime    | Length | Name  |
|--------|------------------|--------|-------|
| -a---- | 20.01.2025 14:39 | 36     | z.cmd |

  
PS E:\Development\Scripts> Get-ChildItem -Force  
Directory: E:\Development\Scripts  


| Mode   | LastWriteTime    | Length | Name  |
|--------|------------------|--------|-------|
| -arhs- | 20.01.2025 14:18 | 37     | x.bat |
| -a---- | 20.01.2025 14:39 | 36     | z.cmd |


```

Рис. Находим все файлы

Чтобы вывести на экран атрибуты нашего файла, тоже придется применить параметр -Force.

Нам расшифровали все четыре буквы: r-h-s-a.

```
PS E:\Development\Scripts> Get-Item x.bat  
Get-Item : Could not find item E:\Development\Scripts\x.bat.  
At line:1 char:1  
+ Get-Item x.bat    + CategoryInfo          : ObjectNotFound: (E:\Development\Scripts\x.bat:String) [Get-Item], IOException  
    + FullyQualifiedErrorId : ItemNotFound,Microsoft.PowerShell.Commands.GetItemCommand  
  
PS E:\Development\Scripts> Get-Item -Force x.bat  
Directory: E:\Development\Scripts  

```

Рис. Атрибуты скрытого файла

Задание. Измените атрибуты файла с помощью команды `attrib` или в Проводнике. Проверьте с помощью `Get-Item`, как эти разрешения проявляются в виде свойств объекта.

## КОДИРОВАНИЕ СВОЙСТВ

Все эти атрибуты (свойства, права доступа, разрешения) закодированы нулями и единицами. Чтобы с этими единицами познакомиться, придется погрузиться в тонкости программирования PowerShell.

Получим свойство `Attributes` для наших файлов. Преобразуем тип этого значения в `int` – целое число. Преобразование типов здесь выполняется с помощью квадратных скобок. Пишем `[int]` перед нашим значением. Получаем 39 и 32.

Чтобы расшифровать эти значения, посмотрим на готовые «системные константы».

Более красиво это формулируется так: «статический член `ReadOnly` из перечислимого типа `FileAttributes` в пространстве имен `System.IO`». Фактически, мы вытащили готовое значение из системной библиотеки. И вывели его на экран в виде целого числа.

Так мы проделали с каждым из четырех атрибутов.

```
PS E:\Development\Scripts> [int](Get-Item -Force x.bat).Attributes  
39  
PS E:\Development\Scripts> [int](Get-Item -Force z.cmd).Attributes  
32  
PS E:\Development\Scripts> [IO.FileAttributes]::ReadOnly  
ReadOnly  
PS E:\Development\Scripts> [int][IO.FileAttributes]::ReadOnly  
1  
PS E:\Development\Scripts> [int][IO.FileAttributes]::Hidden  
2  
PS E:\Development\Scripts> [int][IO.FileAttributes]::System  
4  
PS E:\Development\Scripts> [int][IO.FileAttributes]::Archive  
32  
PS E:\Development\Scripts> [int](Get-Item -Force z.cmd).Attributes  
32
```

Рис. Двоичная запись атрибутов

Теперь понятно, что 32 – это один бит, означающий «Только для чтения». И это всего лишь двойка в пятой степени.

Если нужно, освежите в памяти двоичную систему и перевод между двоичной и десятичной системами счисления.

Тогда число 39 получается как сумма атрибутов 32 +4 +2 +1. Это все степени двойки. То есть это отдельные биты, в которые записали единички.

Внутреннее устройство команд PowerShell иногда перекликается с языком запросов к базам данных SQL. Это прежде всего ключевые слова SELECT и LIKE. Поскольку SQL был разработан раньше, можно подозревать, что это он повлиял на скрипты PS.

С другой стороны, с помощью команд PowerShell можно выполнять SQL-запросы. Для этого используется команда Invoke-Sqlcmd.

Можно даже обнаружить регулярные выражения. Их тоже позаимствовали у своих предшественников.

Задание. Выясните, насколько похожи запросы SQL и формат команд PowerShell. Как работает SELECT и можно ли найти здесь какую-нибудь аналогию.

Задание. Ознакомьтесь с использованием регулярных выражений в PowerShell. Получите с помощью чат-бота пару простых примеров и проверьте, как они работают.

## ПЕРЕНАПРАВЛЕНИЕ ВЫВОДА

Обычно команды выводят свои сообщения на экран. Можно также записать тот же самый текст в обычный текстовый файл.

Простейший общепринятый метод сводится к перенаправлению вывода. Это означает, что результаты выполнения очередной команды вместо экрана отправляют в файл с указанным названием. Перенаправление во многих ОС делается с помощью значка «больше», который в данном случае изображает из себя «стрелочку вправо». Такой способ традиционно работает в командном окне (терминале, консоли) DOS, Windows, Linux и т. п.



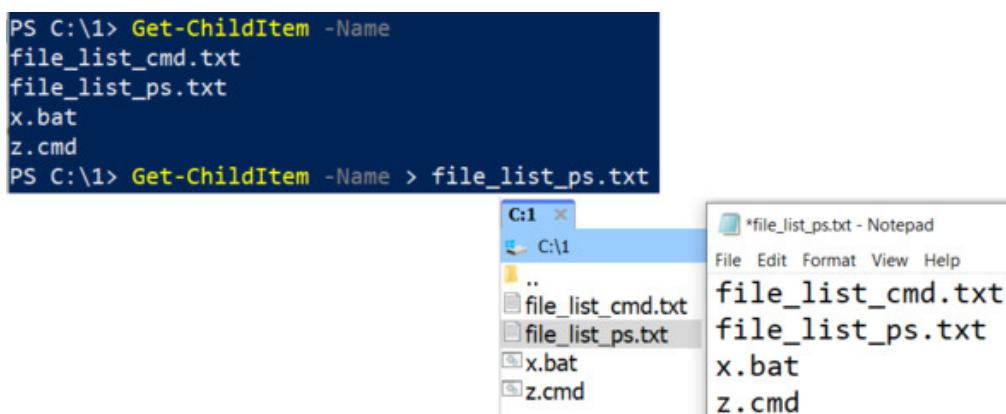
Рис. Перенаправление вывода для CMD

В данном примере команда DIR выводит содержимое текущей директории (папки, каталога) в виде списка файлов. Ключ (опция) /b дает только список имен файлов без подробностей – от английского bare format – краткий формат вывода (буквально – «голый»).

Аналогичный результат получаем в PowerShell. Список файлов получаем командой Get-ChildItem – получить список «потомков» для текущего (родительского) объекта. Конечно же, здесь речь идет о каталогах и файлах. То есть мы получаем список файлов и директорий, которые расположены в текущей директории.

Чтобы получить только имена (названия) файлов и каталогов, используем параметр -Name.

Затем перенаправляем вывод в файл значком >. Это традиционный, проверенный десятилетиями подход.



```
PS C:\1> Get-ChildItem -Name
file_list_cmd.txt
file_list_ps.txt
x.bat
z.cmd
PS C:\1> Get-ChildItem -Name > file_list_ps.txt
```

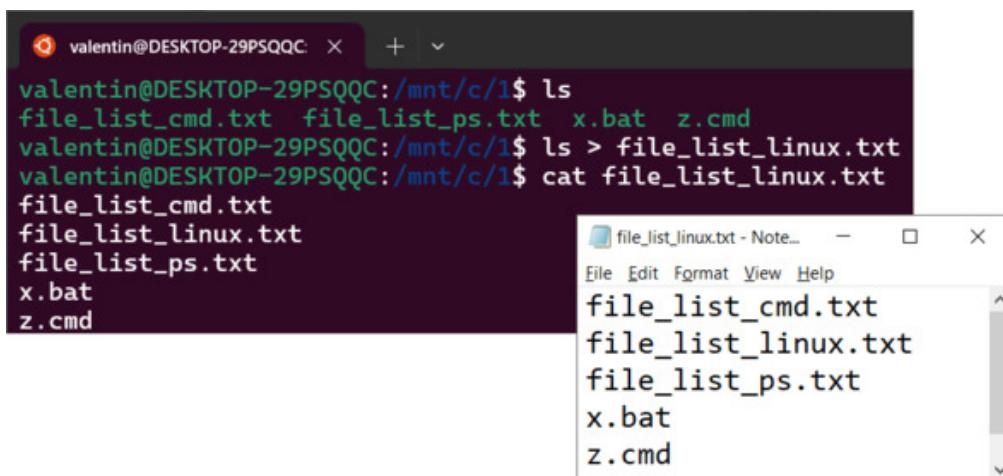
The screenshot shows a Windows command-line interface (cmd.exe) window and a Notepad application window. The cmd window displays the command 'Get-ChildItem -Name' followed by a list of files: 'file\_list\_cmd.txt', 'file\_list\_ps.txt', 'x.bat', and 'z.cmd'. Below this, the command 'Get-ChildItem -Name > file\_list\_ps.txt' is shown. To the right, a Notepad window titled '\*file\_list\_ps.txt - Notepad' shows the contents of the file, which are identical to the list in the cmd window: 'file\_list\_cmd.txt', 'file\_list\_ps.txt', 'x.bat', and 'z.cmd'.

Рис. Перенаправление вывода для PowerShell

Следующий пример выполняется в Ubuntu Linux, который можно установить внутри Windows через магазин приложений Microsoft Store. Здесь список

файлов выводим командой LS – от английского слова LIST – «список».

Такой «внутренний», виртуальный вариант Линукса получает доступ к дискам Windows. В нашем примере через папку /mnt/c/1. Это каталог 1 на диске С:, который «смонтирован» в папку /mnt – от слова MOUNT – монтировать. В операционных системах «монтажирование» – это действие по «подключению диска», чтобы получить к нему доступ.



The screenshot shows a terminal window on a Linux desktop. The terminal output is as follows:

```
valentin@DESKTOP-29PSQQC:~$ ls
file_list_cmd.txt file_list_ps.txt x.bat z.cmd
valentin@DESKTOP-29PSQQC:~$ ls > file_list_linux.txt
valentin@DESKTOP-29PSQQC:~$ cat file_list_linux.txt
```

Below the terminal, a note application window titled "file\_list\_linux.txt - Note..." displays the same list of files:

File
file_list_cmd.txt
file_list_linux.txt
file_list_ps.txt
x.bat
z.cmd

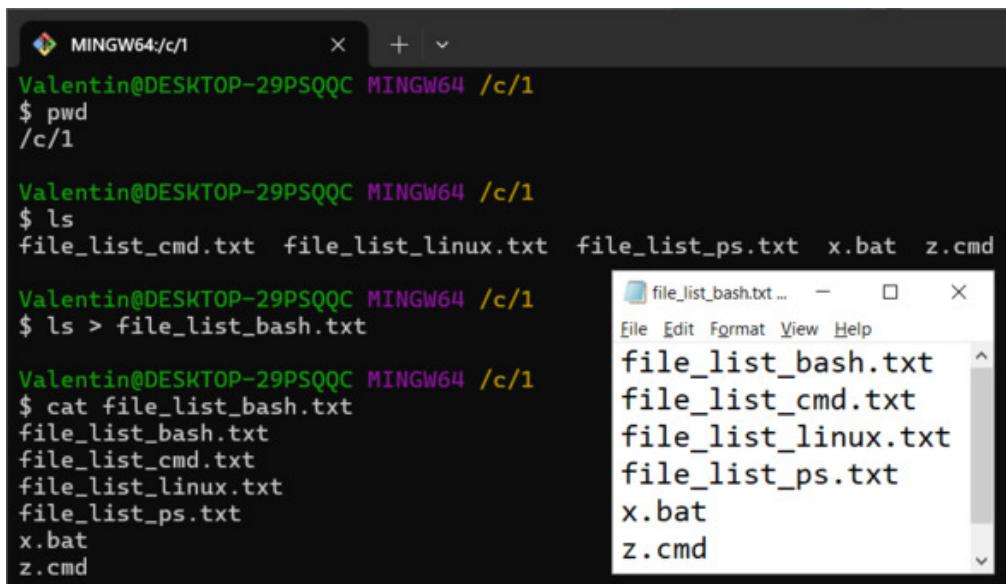
Рис. Перенаправление вывода в Linux

Аналогично работает консоль Bash, которая устанавливается вместе с пакетом Git для управления версией программного продукта. Git – это отдельная тема, но нас тут пока интересует «кусочек Линукса». И в этом окне можно вводить основные команды Линукса.

В этом примере доступ к диску С: получаем через папку /c.

Обратим внимание, что в Линуксе имена каталогов разделяются наклонной чертой типа «прямой слэш» – с наклоном вправо. А в ОС Windows используют «обратный слэш» – черта с наклоном влево. Видимо, разработчики решили показать, что у них всё по-

другому – даже наклонная черта другая. Хотя за основу, за образец явно был взят UNIX.



The screenshot shows a terminal window titled 'MINGW64:/c/1' with the command history:

```
Valentin@DESKTOP-29PSQQC MINGW64 /c/1
$ pwd
/c/1

Valentin@DESKTOP-29PSQQC MINGW64 /c/1
$ ls
file_list_cmd.txt  file_list_linux.txt  file_list_ps.txt  x.bat  z.cmd

Valentin@DESKTOP-29PSQQC MINGW64 /c/1
$ ls > file_list_bash.txt

Valentin@DESKTOP-29PSQQC MINGW64 /c/1
$ cat file_list_bash.txt
file_list_bash.txt
file_list_cmd.txt
file_list_linux.txt
file_list_ps.txt
x.bat
z.cmd
```

To the right of the terminal is a small window titled 'file\_list\_bash.txt ...' containing the same list of files:

file_list_bash.txt
file_list_cmd.txt
file_list_linux.txt
file_list_ps.txt
x.bat
z.cmd

Рис. Перенаправление вывода для Bash

Итак, мы рассмотрели несколько примеров перенаправления вывода в файл с помощью значка>. Это традиционный способ. Но даже здесь разработчики PowerShell нашли, что можно усовершенствовать.

## ОТПРАВКА В ФАЙЛ

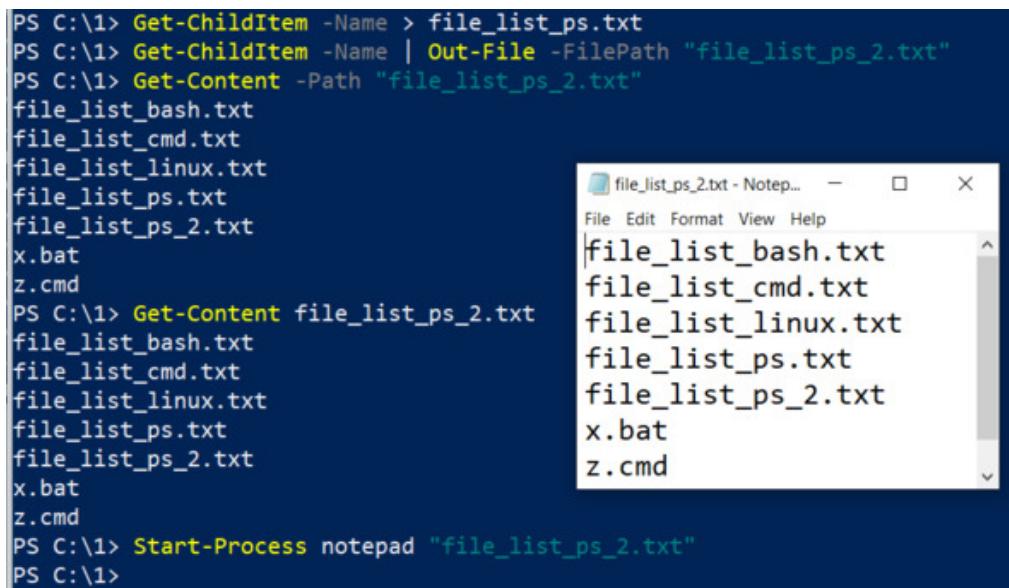
В рамках PowerShell идея записи сообщений в файл получила дальнейшее развитие. Теперь это готовые команды, которые работают с разными форматами.

Обычный текст можно записать командой Out-File. Мы передаем ей на вход список файлов. Делаем это с помощью конвейера – через вертикальную черту. Далее, в помощью параметра -FilePath сообщаем имя файла или даже полный путь к выходному файлу.

Полученный текстовый файл выводим на экран командой Get-Content. Говорящее название намекает,

что мы получаем содержимое файла: Get (получить) + Content (содержимое).

Для надежности откроем тот же самый файл с помощью Блокнота и получим тот же результат – в отдельном окне. Для запуска приложения Блокнот есть отдельный инструмент – команда Start-Process. Расшифруем и это название: Start (запустить на выполнение, начать исполнять) + Process (процесс – это программа, которая выполняется прямо сейчас). Пока программа лежит на диске, она ничего не делает, это просто ФАЙЛ. А вот когда мы запустили этот файл на выполнение, он отправляется в оперативную память и теперь это называется ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС.



The screenshot shows a Windows Command Prompt window with the following PowerShell session:

```
PS C:\1> Get-ChildItem -Name > file_list_ps.txt
PS C:\1> Get-ChildItem -Name | Out-File -FilePath "file_list_ps_2.txt"
PS C:\1> Get-Content -Path "file_list_ps_2.txt"
file_list_bash.txt
file_list_cmd.txt
file_list_linux.txt
file_list_ps.txt
file_list_ps_2.txt
x.bat
z.cmd
PS C:\1> Get-Content file_list_ps_2.txt
file_list_bash.txt
file_list_cmd.txt
file_list_linux.txt
file_list_ps.txt
file_list_ps_2.txt
x.bat
z.cmd
PS C:\1> Start-Process notepad "file_list_ps_2.txt"
PS C:\1>
```

Next to the command prompt is a screenshot of a Notepad window titled "file\_list\_ps\_2.txt - Notepad...". The window contains the same list of files and executables as the command prompt output:

- file\_list\_bash.txt
- file\_list\_cmd.txt
- file\_list\_linux.txt
- file\_list\_ps.txt
- file\_list\_ps\_2.txt
- x.bat
- z.cmd

Рис. Запись текста в файл

**Задание.** Выясните, как дописывать вывод в существующий файл, не удаляя его. Соответствующее английское название: APPEND. Попрактикуйтесь дописывать файл в разных средах.

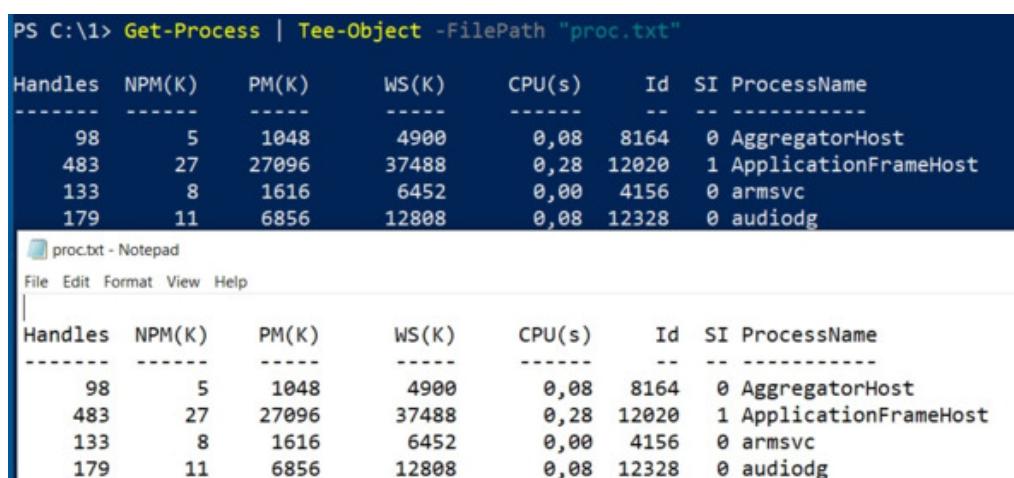
## Т-ОБРАЗНЫЙ ОБЪЕКТ

Кроме перенаправления вывода в файл, мы можем одновременно выводить сообщения на экран и записывать их в текстовый файл. Для этого существует команда Tee-Object.

Английское ТЕЕ – это произношение буквы Т. В названии команды использована аналогия с Т-образным соединением труб или проводов. Здесь на входе имеется один поток (в данном случае поток текста). А на выходе получаем два потока.

Задание. Найдите в интернете картинки на тему «Tee connector», «т-коннектор», «т-образное соединение труб» и «т-образное соединение проводов».

В следующем примере мы получаем список процессов (запущенных программ). С помощью Tee-Object мы отправляем этот список на экран и в файл – одновременно. Имя файла указываем с помощью параметра -FilePath.



```
PS C:\1> Get-Process | Tee-Object -FilePath "proc.txt"

Handles  NPM(K)    PM(K)      WS(K)      CPU(s)      Id  SI ProcessName
-----  -----    -----      -----      -----      --  -- -----
    98      5       1048       4900       0,08     8164  0 AggregatorHost
   483     27      27096      37488       0,28    12020  1 ApplicationFrameHost
   133      8       1616       6452       0,00     4156  0 armsvc
   179     11      6856      12808       0,08   12328  0 audiodg

proc.txt - Notepad
File Edit Format View Help
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
98	5	1048	4900	0,08	8164	0	AggregatorHost
483	27	27096	37488	0,28	12020	1	ApplicationFrameHost
133	8	1616	6452	0,00	4156	0	armsvc
179	11	6856	12808	0,08	12328	0	audiodg

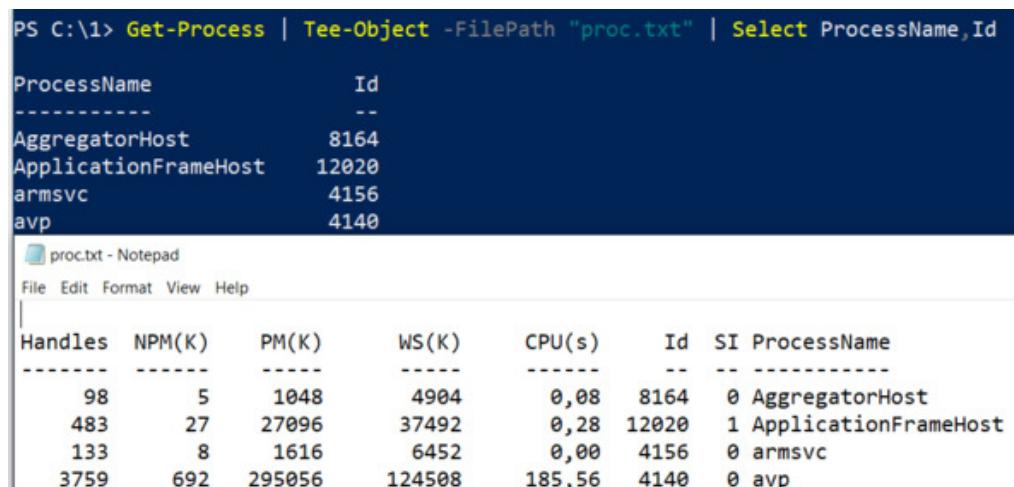
Рис. Одновременный вывод на экран и в файл

Фактически, Tee-Object не просто выводит данные в файл и на экран, а позволяет отправить их дальше по конвейеру (pipeline). Таким образом, мы можем сохранить полученные сведения и продолжить их обработку.

В следующем примере мы записываем в файл подробные сведения о процессах, а на экран выводим только два столбца:

- Id – идентификатор процесса;
- ProcessName – имя процесса.

Для выбора столбцов у нас есть команда Select. Названия столбцов перечисляем через запятую – в нужном нам порядке. Это чем-то напоминает SELECT в запросах к базам данных на языке SQL.



```
PS C:\1> Get-Process | Tee-Object -FilePath "proc.txt" | Select ProcessName,Id
```

ProcessName	Id
AggregatorHost	8164
ApplicationFrameHost	12020
armsvc	4156
avp	4140

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
98	5	1048	4904	0,08	8164	0	AggregatorHost
483	27	27096	37492	0,28	12020	1	ApplicationFrameHost
133	8	1616	6452	0,00	4156	0	armsvc
3759	692	295056	124508	185,56	4140	0	avp

Рис. Запись в файл и продолжение обработки

Но и это еще не все. Мы можем сохранить полученные результаты в переменную – вместо записи в файл. Используем параметр -Variable и указываем имя переменной для вывода.

Теперь мы сможем к ней обращаться через символ \$ и использовать в других командах. Можно просто

вывести содержимое переменной на экран.

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
98	5	1048	4900	0,08	8164	0	AggregatorHost
483	27	27096	37488	0,28	12020	1	ApplicationFrameHost
133	8	1616	6452	0,00	4156	0	armsvc
3711	690	295000	184760	188,28	4140	0	avp

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
98	5	1048	4900	0,08	8164	0	AggregatorHost
483	27	27096	37488	0,28	12020	1	ApplicationFrameHost
133	8	1616	6452	0,00	4156	0	armsvc
3711	690	295000	184760	188,30	4140	0	avp

Рис. Сохранение данных в переменную

Кроме текстового файла, мы можем создать файл в формате электронных таблиц CSV – Comma Separated Values – Значения, разделенные запятыми. Такие файлы можно будет обрабатывать в Excel и любых других электронных процессорах, а также в программах на Python.

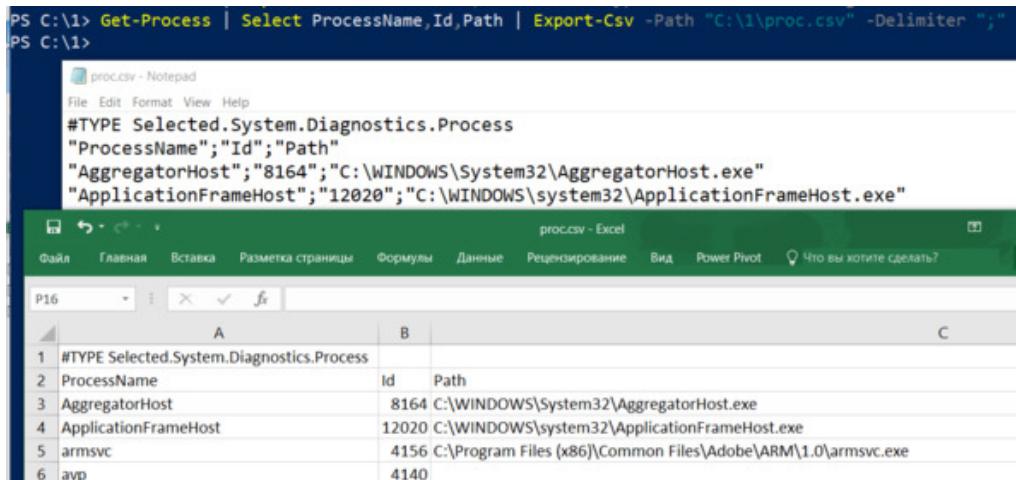
Для вывода в CSV можно использовать Export-Csv.

В следующем примере мы выводим список процессов с помощью Get-Process.

В полученном списке с помощью Select оставляем только столбцы ProcessName, Id и Path (путь к исполняемому файлу). Выводим результаты в CSV командой Export-Csv. С помощью параметра -Path указываем полный путь к файлу. При работе в текущем каталоге можно ограничиться только именем файла. Наконец, для работы с локализованной «русской» версией Excel мы указываем разделитель значений (полей) с помощью параметра -Delimiter. Нас устроит точка с запятой";».

Содержимое полученного файла можно изучить в Блокноте.

Если открыть этот файл в Excel, мы сможем обработать данные, расположенные по столбцам.



	A	B	C
1	#TYPE Selected.System.Diagnostics.Process		
2	ProcessName	Id	Path
3	AggregatorHost	8164	C:\WINDOWS\System32\AggregatorHost.exe
4	ApplicationFrameHost	12020	C:\WINDOWS\system32\ApplicationFrameHost.exe
5	armsvc	4156	C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\armsvc.exe
6	avp	4140	

Рис. Запись в файл CSV

Задание. Попрактикуйтесь выводить информацию в файл – разными способами и в разных форматах.

## СКРИПТЫ POWERSHELL

Мы познакомились с командами PowerShell. Постепенно наши команды становятся все длиннее и сложнее. Для удобства работы можно организовать длинные команды в скрипты-сценарии. Их легко хранить в виде текстовых файлов. Их можно многократно запускать. Их можно редактировать. Для работы со скриптами PowerShell служит ISE – Integrated Scripting Environment. Это интегрированная среда разработки скриптов.

Нажимаем кнопку Start – Пуск и начинаем вводить Power...

Выбираем запуск ISE с правами администратора.

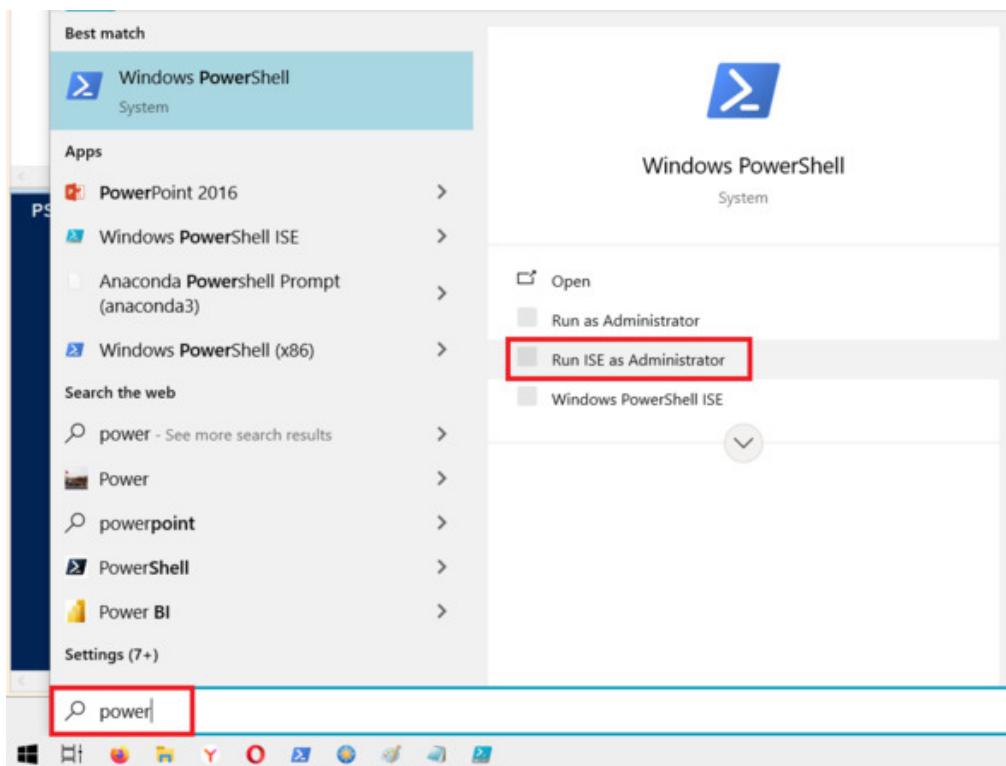


Рис. Запуск интегрированной среды ISE

Напишем программу для вывода приветствия в отдельном окне. Мы уже создавали окно с текстовым сообщением MsgBox.

Наш сценарий будет состоять из двух строк. Для украшения скрипта добавим комментарии.

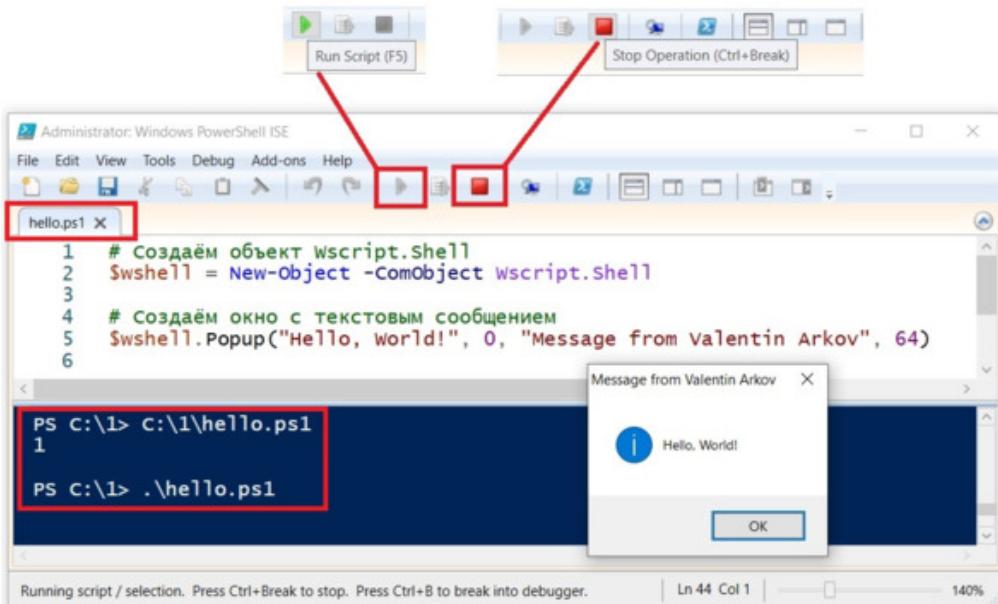


Рис. Выводим сообщение в окне

Обсудим текст нашей программы-сценария.

(1) В первой строке мы даем комментарий – очень коротко. Комментарии начинаются с символа решетки #. Это встречается во многих языках программирования, включая Python. Слишком подробные комментарии затрудняют чтение программы. Подробные пояснения лучше давать в документации. Что мы сделаем далее.

(2) Затем мы создаем новый объект Wscript.Shell и сохраняем его в переменную \$wshell. Команда New-Object позволяет создавать различные объекты. COM-объекты (Component Object Model).

В нашем примере мы указываем параметр -ComObject. Это означает, что мы создаем объект через интерфейс COM (Component Object Model). Технология COM позволяет программам взаимодействовать друг с другом.

Наконец, мы указываем имя COM-объекта, который мы хотим создать. Wscript.Shell – это объект оболочки Windows Script Host. Через него мы можем вызывать

готовые функции операционной системы. У нас это будет функция вывода сообщения.

(3) Отделяем разные части программы пустой строкой. Это делается для удобства и лёгкости чтения – Readability – «читабельности» кода.

(4) Краткий комментарий перед следующей командой.

(5) Создаём окно с текстовым сообщением с помощью метода `PopUp`. Мы обращаемся нашему объекту `$wshell` и вызываем его метод (встроенную функцию) – через точку после имени объекта. Название метода `PopUp` расшифровывается как «a pop-up window on a computer screen» – «всплывающее окно на экране компьютера». Фактически, мы таким способом обращаемся к готовой функции ОС, то есть используем System Call – «системный вызов» типа Message Box. Можно также сказать API Call – «вызов функции через API».

Параметры метода `PopUp`:

- Текст сообщения: «Hello, World!»;
- Время до автоматического закрытия окна: 0 секунд, то есть ожидаем нажатия кнопки;
- Заголовок окна: «Message from Valentin Arkov»;
- Тип иконки и кнопок: 64 – Информационная иконка (восклицательный знак на синем фоне).

Сохраняем скрипт под коротким информативным названием: Save as – PowerShell Scripts (\*.ps1) – `hello.ps1`. Обращаем внимание на расширение файла:

PS1. Здесь можно было бы ожидать всего две буквы: PS – от слов Power и Shell.

Однако, это расширение уже занято: PS означает PostScript. Это особый язык описания документов, предназначенный для вывода на печать. Есть даже специальные принтеры, которые его понимают. И разработан он был гораздо раньше.

Для дальнейших опытов нужно разрешить выполнение скриптов. Установим политику выполнения: Unrestricted – «без ограничений». Для этого выполняем следующую команду – с правами администратора:

```
Set-ExecutionPolicy Unrestricted
```

Подтверждаем изменение политики, нажав букву Y и Enter.

Для запуска программы есть две полезные кнопки и соответствующие «горячие клавиши»:

- зеленая стрелочка – Run Script – F5;
- красный квадратик – Stop Operation – Ctrl+Break.

Запустить скрипт можно и вручную. Для этого переходим в папку, где он находится, и вводим его имя. Правда, вначале придётся указать точку и «обратный слэш». Это напоминает традиции Linux. Мы явно сообщаем, что файл находится в текущем каталоге. Символ «точка» означает текущий каталог.

После завершения скрипта мы получаем единичку «1» в окне терминала. Это результат выполнения

последней команды. В нашем примере метод `PopUp()` возвращает целое число – идентификатор кнопки, нажатой в диалоговом окне.

Теперь можно проверить полученный код и определить логику поведения программы. Для этого используется конструкция `if-elseif`.

Для составления таких программ можно привлечь любой чат-бот, но потом сгенерированную программу нужно вручную довести до работающего состояния, или, как говорят, «допилить». При генерации таких скриптов можно будет убедиться на собственном опыте, что здесь потребуется написать очень подробный промпт. Иначе нейронка обязательно что-нибудь сделает по-своему.

**Задание.** Поэкспериментируйте с параметрами метода `PopUp`. Сравните результаты с аналогичными экспериментами на VBScript. Задайте в окне две кнопки: `OK` и `Cancel`. Проверьте, какое значение возвращается при нажатии каждой из кнопок.

**Задание.** Составьте скрипт, который проверяет код нажатой клавиши и выводит соответствующее сообщение в консоль.

**Задание.** Просмотрите на Википедии статью `Component Object Model` и обратите внимание на красивые названия технологий, которые были разработаны в процессе развития COM.

**Задание.** Просмотрите на Википедии статью `PostScript` и обратите внимание на примеры заголовка

и текста программы.

## РЕЕСТР WINDOWS REGISTRY

Множество настроек ОС Windows хранится на компьютере особым образом в виде «реестра». Эти сведения находятся в нескольких файлах и организованы в виде «дерева». Более красивая формулировка звучит так: «Реестр Windows – это иерархическая база данных».

Обычно мы получаем доступ к реестру через программу regedit – Registry Editor – редактор реестра. Его можно запустить вручную: [Windows+R] – regedit.

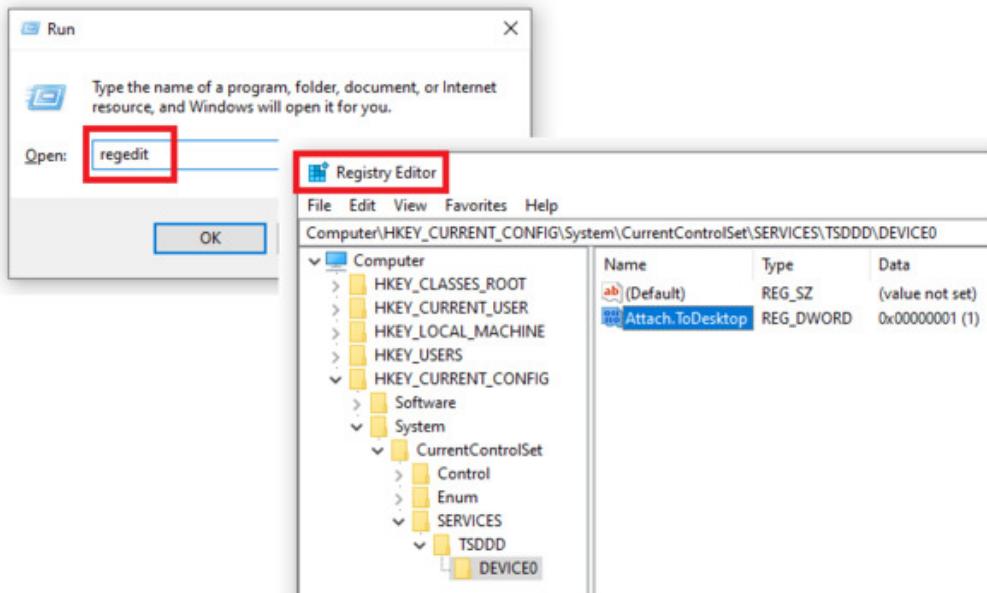


Рис. Редактор реестра

В этом редакторе мы можем просматривать и изменять настройки. Разделы реестра организованы как отдельные «ветви» большого «дерева». Разделы называют словом HIVE, что буквально означает «улей, в котором обитают пчёлы».

Настройки здесь хранятся в виде пар ключ-значение (key-value). «Ключ» соответствует имени переменной (или объекта), «значение» – это содержимое нашей «переменной». Внутри «значения» тоже могут быть пары «ключ-значение». Получается своеобразная «матрешка».

Например, настройки текущего пользователя хранятся в разделе HKEY\_CURRENT\_USER, сокращено HKCU.

Задание. Просмотрите в Википедии статью Реестр Windows и выясните назначение разделов реестра. Запустите редактор реестра и найдите эти разделы.

Если раскрыть «ветку» и добраться до «листьев» нашего «дерева», можно увидеть в правой части окна редактора таблицу с колонками Name – Type – Data. На рисунке можно увидеть тип DWORD – «двойное машинное слово», то есть целое число длиной четыре байта. В колонке «значение» выводится его значение в шестнадцатеричном (начинается с 0x) и в десятичном виде (приводится в скобках).

Задание. Раскройте любые ветки реестра и обратите внимание на тип и значение любых параметров.

Мы можем добраться до реестра и с помощью PowerShell. Доверяем вам самостоятельно с этим разобраться.

Задание. С помощью любого интеллектуального помощника выясните, как просматривать ветки реестра.

Найдите настройки, отвечающие за автоматический запуск программ на сменных носителях и выведите их на экран. Используйте редактор реестра и PowerShell.

## ИТОГИ РАЗДЕЛА

В этом разделе мы познакомились с усовершенствованной командной строкой под названием PowerShell. Перед нами не просто командное окно, а полноценный язык программирования. И для него имеется своя интегрированная среда разработки.

## ВЕСЁЛЫЕ КАРТИНКИ

Одна картинка говорит больше,  
чем тысяча слов  
(народная пословица)

Завершим наши упражнения на весёлой ноте.  
Зададим значок-иконку «ручной работы» для нашей  
флешки.

Будем использовать файл autorun.inf. Он работает  
на сменных носителях — компакт-дисках и USB-  
флешках. Позволяет автоматически запускать  
приложения при подключении устройства к компьютеру.  
Фактически, этот файл содержит настройки  
конфигурации. Можно провести определённую аналогию  
между файлом autorun.inf и скриптами. Оба инструмента  
предназначены для автоматизации действий  
пользователя. Но работают они по-разному.

Задание. Просмотрите на Википедии статью  
Autorun.inf и выясните, как обычно выглядит  
содержимое такого файла.

Задание. Просмотрите на Википедии статью Иконка  
(графический интерфейс) и выясните, какого размера  
обычно бывают эти «значки».

Пусть наша иконка будет уникальной, авторской. Для этого можно сгенерировать любую картинку. Есть много нейросетей, которые бесплатно генерируют картинки по текстовому описанию, например, Кандинский от Сбера или Шедеврум от Яндекса.

Исходная картинка может быть любого размера. Главное требование – она должна быть квадратной. В конце концов, можно взять свою фотографию и подрезать ее до нужного размера – в любом графическом редакторе.

Сохраняем картинку в формате иконки \*.ICO. Для этого можно использовать любой бесплатный онлайн конвертер иконок, например:

<https://image.online-convert.com/ru/convert-to-ico>

Далее создаем на флешке текстовый файл autorun.inf в корневом каталоге. В этом тексте нет никакого форматирования, поэтому можно использовать обычный Блокнот.

В этом файле организуем секцию [autorun]. Задаем параметр icon и указываем имя файла. Если нужно, укажем полный путь к файлу.

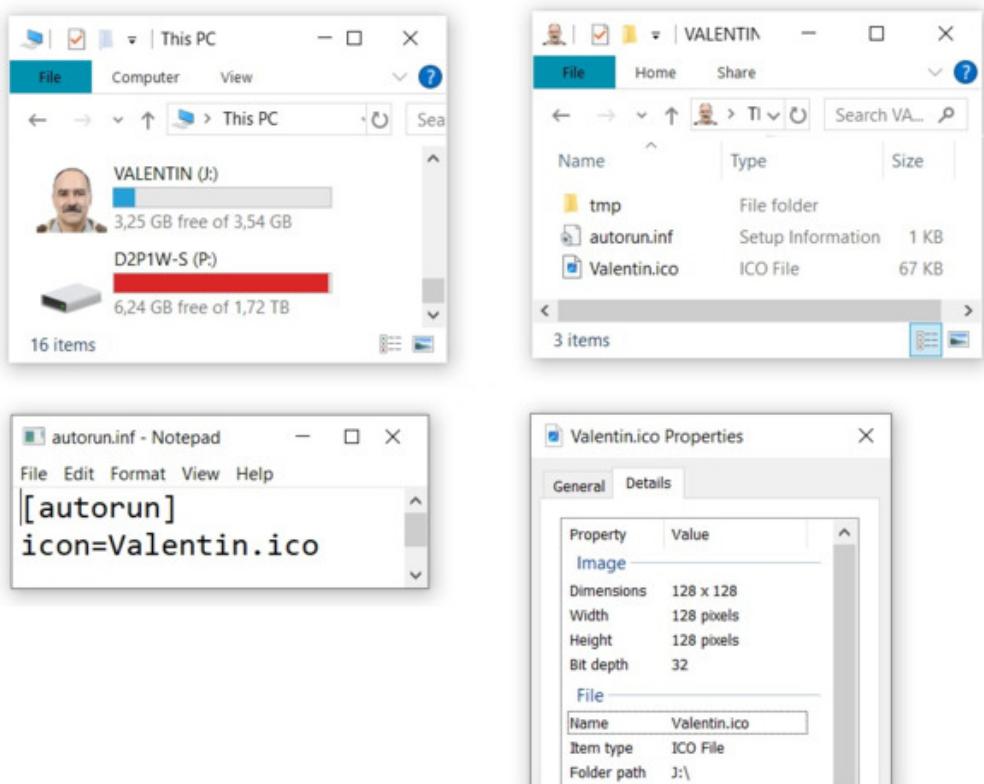


Рис. Иконка на флешке

# **ЗАКЛЮЧЕНИЕ**

А напоследок я скажу...

(Белла Ахмадуллина)

Скрипты – это сценарии работы. Автоматической работы без участия пользователя. Скрипты бывают разными, но выполняются они «шаг за шагом», с помощью интерпретатора. И в этом их отличие от традиционных языков программирования, которым нужен компилятор, чтобы создать исполняемый файл.

В любом случае, мы имеем дело с инструментами программирования, которые могут быть полезными во многих компьютерных профессиях. А если они лично вам не пригодятся прямо сейчас, вы получили очередную порцию компьютерной грамотности и развили свои практические навыки. Будем считать, что это своеобразный умственный «фитнес».

Для самых наблюдательных мы оставили одну нерешенную загадку на обложке книги. Попробуйте её найти, разгадать и применить эту технологию.

Спасибо за внимание. И до новых встреч.