Создание скрипта для автоматизации установки и настройки ПО

Выполнили: Шарманов Д.А., Иванишин Д.Е.

Введение

В ходе выполнения задания была создана программа для автоматизации установки и настройки ПО в учебных аудиториях для Windows. Разработанная нами программа позволяет быстро установить требуемое программное обеспечение и настроить его. Программа устанавливает программное обеспечение из списка представленного по ссылке: https://disk.yandex.ru/i/lmPLlMgm7NfsBQ.

Существует несколько методов автоматизации установки программного обеспечения: WinGet, Ansible, ручные установщики/скрипты, Chocolatey. Нами был выбран именно последний вариант, по следующим причинам:

- Обилие пакетов;
- Простота в написании скрипта;
- Установка только из проверенных репозиториев;
- Минимизация ручной работы.

Инструкция по пользованию

1. Установите Chocolatey. Для этого откройте PowerShell от имени администратора и введите команду:
   Set-ExecutionPolicy Bypass -Scope **Process** -Force;
   [System.Net.ServicePointManager]::SecurityProtocol =
   [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

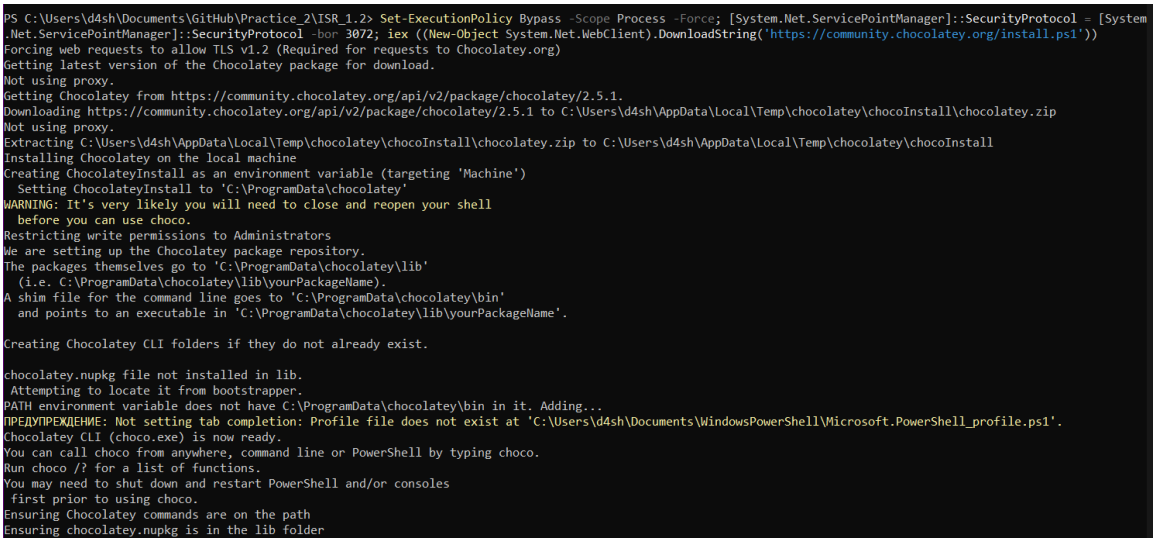

Рис. 1. Демонстрация выполнения скрипта установки Chocolately.

2. Скачайте скрипт software-setup.ps1 из репозитория и поместите его в любую папку.
3. Откройте PowerShell от имени администратора и перейдите в директорию со скриптом с помощью команды cd *путь*.

4. Запустите скрипт командой
   .\software-setup.ps1

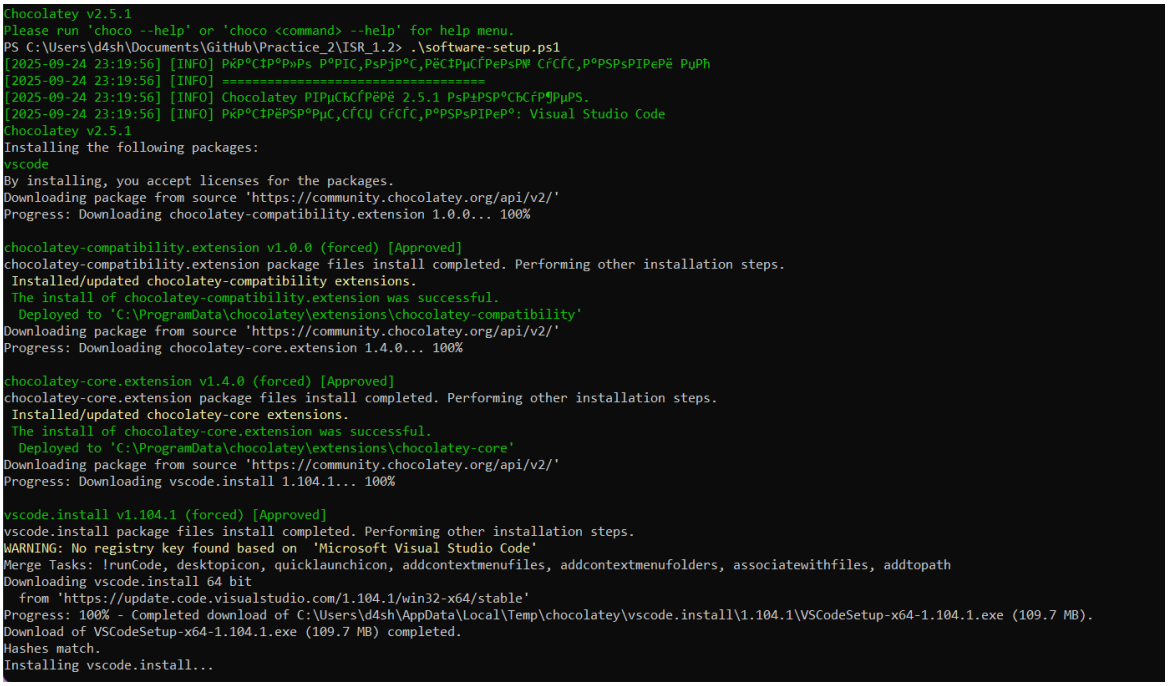

Рис. 2. Демонстрация выполнения скрипта software-setup

5. Дождитесь завершения установки.

Примечание.

- Процесс выполнения скрипта может занять значительное время (около 30 минут, зависит от скорости интернета).
- Некоторые программы (Docker, WSL) могут требовать перезагрузки
- Некоторые программы могут требовать дополнительной ручной настройки после установки

Листинг кода с комментариями

```
<#
.SYNOPSIS
    Скрипт автоматической установки и настройки ПО для разработки и аналитики.
.DESCRIPTION
    Устанавливает Visual Studio Code с расширениями, Docker, PyCharm, Git, GitHub Desktop, Maxima, KNIME, GIMP, Julia, Python, Rust, MSYS2, Zettlr, MiKTeX, TeXstudio, Anaconda, Far Manager, SumatraPDF, Chrome, Flameshot, WSL2, Qalculate, Yandex.Telemost, Sber Jazz, Arc, 7Zip, Firefox, Yandex Browser.
.NOTES
    Требует запуска от имени администратора.
    Требует предварительной установки Chocolatey.
#>

#Requires -RunAsAdministrator

# Функция для логирования
function Write-Log {
    param([string]$Message, [string]$Type = "INFO")
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
```

```
    Write-Host "[$timestamp] [$Type] $Message" -ForegroundColor $(if ($Type -
eq "ERROR") { "Red" } elseif ($Type -eq "WARNING") { "Yellow" } else {
"Green" })
}

# Функция проверки установки Chocolatey
function Test-Chocolatey {
    try {
        $chocoVersion = choco --version
        Write-Log "Chocolatey версии $chocoVersion обнаружен."
        return $true
    }
    catch {
        Write-Log "Chocolatey не установлен. Пожалуйста, установите
Chocolatey перед запуском скрипта." -Type "ERROR"
        Write-Host "Инструкция по установке: https://chocolatey.org/install"
        return $false
    }
}

# Функция установки пакета
function Install-Package {
    param(
        [string]$PackageName,
        [string]$DisplayName = $PackageName
    )

    Write-Log "Начинается установка: $DisplayName"
    try {
        choco install $PackageName -y --force
        if ($LASTEXITCODE -eq 0) {
            Write-Log "Успешно установлено: $DisplayName"
        } else {
            Write-Log "Возникла проблема при установке $DisplayName (код
выхода: $LASTEXITCODE)" -Type "WARNING"
        }
    }
    catch {
        Write-Log "Ошибка при установке $DisplayName : $_" -Type "ERROR"
    }
}

# Функция установки расширений VS Code
function Install-VSCodeExtensions {
    $extensions = @(
        "ms-python.python",
        "ms-vscode.cpptools",
        "ms-azuretools.vscode-docker",
        "ms-vscode.vscode-json",
        "ms-vscode.vscode-typescript-next",
        "bradgashler.htmltagwrap",
        "ecmel.vscode-html-css",
        "ms-vscode.PowerShell",
        "eamodio.gitlens",
        "GitHub.vscode-pull-request-github"
    )

    Write-Log "Установка расширений VS Code..."
    foreach ($extension in $extensions) {
```

```
        Write-Log "Установка расширения: $extension"
        try {
            code --install-extension $extension --force
            Write-Log "Успешно: $extension"
        }
        catch {
            Write-Log "Ошибка при установке расширения $extension : $_" -Type
"WARNING"
        }
    }
}

# Функция настройки WSL
function Setup-WSL {
    Write-Log "Настройка WSL 2..."

    # Включение компонентов Windows
    try {
        Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-
Subsystem-Linux -NoRestart -ErrorAction Stop
        Enable-WindowsOptionalFeature -Online -FeatureName
VirtualMachinePlatform -NoRestart -ErrorAction Stop
        Write-Log "Компоненты WSL включены"
    }
    catch {
        Write-Log "Ошибка при включении компонентов WSL: $_" -Type "ERROR"
        return
    }

    # Установка WSL 2 по умолчанию
    try {
        wsl --set-default-version 2
        Write-Log "WSL 2 установлен как версия по умолчанию"
    }
    catch {
        Write-Log "Предупреждение: не удалось установить WSL 2 как версию по
умолчанию" -Type "WARNING"
    }

    # Установка дистрибутивов Ubuntu
    $ubuntuVersions = @("22.04", "24.04")

    foreach ($version in $ubuntuVersions) {
        Write-Log "Установка Ubuntu $version..."
        try {
            # Проверяем, не установлен ли уже дистрибутив
            $distroName = "Ubuntu-$version"
            if (wsl -l -q | Select-String -Pattern $distroName) {
                Write-Log "Ubuntu $version уже установлен"
                continue
            }

            # Скачиваем и устанавливаем через Winget (альтернативный метод)
            winget install --id Canonical.Ubuntu.$version --source winget --
accept-package-agreements --accept-source-agreements
            Write-Log "Ubuntu $version установлен"
        }
        catch {
```

```
            Write-Log "Ошибка при установке Ubuntu $version : $_" -Type
"WARNING"
        }
    }
}

# Главная функция
function Main {
    Write-Log "Начало автоматической установки ПО"
    Write-Log "=================================="

    # Проверка Chocolatey
    if (-not (Test-Chocolatey)) {
        exit 1
    }

    # Список пакетов для установки через Chocolatey
    $packages = @(
        @{Name = "vscode"; DisplayName = "Visual Studio Code"},
        @{Name = "docker-desktop"; DisplayName = "Docker Desktop"},
        @{Name = "pycharm-community"; DisplayName = "PyCharm Community
Edition"},
        @{Name = "git"; DisplayName = "Git"},
        @{Name = "github-desktop"; DisplayName = "GitHub Desktop"},
        @{Name = "maxima"; DisplayName = "Maxima"},
        @{Name = "knime"; DisplayName = "KNIME Analytics Platform"},
        @{Name = "gimp"; DisplayName = "GIMP"},
        @{Name = "julia"; DisplayName = "Julia"},
        @{Name = "python"; DisplayName = "Python"},
        @{Name = "rust"; DisplayName = "Rust"},
        @{Name = "msys2"; DisplayName = "MSYS2"},
        @{Name = "zettlr"; DisplayName = "Zettlr"},
        @{Name = "miktex"; DisplayName = "MiKTeX"},
        @{Name = "texstudio"; DisplayName = "TeXstudio"},
        @{Name = "anaconda3"; DisplayName = "Anaconda"},
        @{Name = "far"; DisplayName = "Far Manager"},
        @{Name = "sumatrapdf"; DisplayName = "SumatraPDF"},
        @{Name = "googlechrome"; DisplayName = "Google Chrome"},
        @{Name = "flameshot"; DisplayName = "Flameshot"},
        @{Name = "qalculate"; DisplayName = "Qalculate!"},
        @{Name = "yandex-telemost"; DisplayName = "Yandex.Telemost"},
        @{Name = "sber-jazz"; DisplayName = "Sber Jazz"},
        @{Name = "arc"; DisplayName = "Arc Browser"},
        @{Name = "7zip"; DisplayName = "7-Zip"},
        @{Name = "firefox"; DisplayName = "Mozilla Firefox"},
        @{Name = "yandex"; DisplayName = "Yandex Browser"}
    )

    # Установка пакетов
    foreach ($package in $packages) {
        Install-Package -PackageName $package.Name -DisplayName
$package.DisplayName
    }

    # Установка расширений VS Code
    Install-VSCodeExtensions

    # Настройка WSL
    Setup-WSL
```

```
    Write-Log "==================================="
    Write-Log "Автоматическая установка завершена!"
    Write-Log "Некоторые программы могут требовать перезагрузки системы."
    Write-Log "Рекомендуется перезагрузить компьютер."
}

# Запуск главной функции
Main
```

Заключение

Возможные дальнейшие улучшения:

- Создание резервных копий;
- Добавление модульной архитектуры;
- Расширение обработки ошибок.

Нами было проверена работа данного скрипта, и результат успешный. Все этапы работы были выполнены совместно. Написанием инструкции по установке занимался Иванишин Д.Е, а Шарманов Д.А. занимался тестированием. Все остальные этапы были выполнены в равномерном объеме.