Разработка скрипта для автоматизации установки и настройки прикладного программного обеспечения

1. Задачи

- Исследование инструментов автоматизации для Windows
- Разработка скрипта установки с использованием PowerShell и Chocolatey
- Создание резервного образа системы с установленным ПО
- Тестирование скрипта в виртуальной среде

2. Скрипт автоматизации установки ПО

2.1 Основной скрипт установки (install-software.ps1)

```
# Скрипт автоматической установки ПО для компьютерных аудиторий
# Требует запуска от администратора

param(
    [switch]$Audience313 = $false  # Флаг для установки дополнительного ПО для аудитории 313
)

Write-Host "=== АВТОМАТИЗИРОВАННАЯ УСТАНОВКА ПО ===" -ForegroundColor Green
Write-Host "Начало установки: $(Get-Date)" -ForegroundColor Yellow

# Функция для проверки успешности установки
function Test-InstallSuccess {
    param($Process)
    if ($Process.ExitCode -ne 0) {
        Write-Host "ОШИБКА установки! Код выхода: $($Process.ExitCode)" -ForegroundColor Red
        return $false
    }
    return $true
```

```
}

# 1. Установка Chocolatey (менеджер пакетов)
Write-Host "`n1. Установка Chocolatey..." -ForegroundColor Cyan
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'))

# 2. Установка базового ПО через Chocolatey
$softwareList = @(
    "git",
    "vscode",
    "docker-desktop",
    "pycharm-community",
    "github-desktop",
    "python",
    "julia",
    "gimp",
    "firefox",
    "googlechrome",
    "microsoft-edge",
    "sumatrapdf",
    "far",
    "7zip",
    "anki"
)
```

```
Write-Host "`n2. Установка основного ПО через Chocolatey..." -ForegroundColor Cyan

foreach ($software in $softwareList) {
    Write-Host "Устанавливаем $software..." -ForegroundColor White
    $process = Start-Process choco -ArgumentList "install", $software, "-y", "--force" -Wait -PassThru
    Test-InstallSuccess -Process $process
}

# 3. Установка специализированного ПО
Write-Host "`n3. Установка специализированного ПО..." -ForegroundColor Cyan

# 3.1. MSYS2
Write-Host "Установка MSYS2..." -ForegroundColor White
Invoke-WebRequest -Uri "https://github.com/msys2/msys2-installer/releases/download/2023-03-18/msys2-x86_64-20230318.exe" -OutFile "$env:TEMP\msys2.exe"
Start-Process "$env:TEMP\msys2.exe" -ArgumentList "--confirm-command", "--accept-messages", "--confirm-command" -Wait

# 3.2. Anaconda
Write-Host "Установка Anaconda..." -ForegroundColor White
Invoke-WebRequest -Uri "https://repo.anaconda.com/archive/Anaconda3-2023.09-0-Windows-x86_64.exe" -OutFile "$env:TEMP\anaconda.exe"
Start-Process "$env:TEMP\anaconda.exe" -ArgumentList "/S", "/InstallationType=JustMe", "/AddToPath=1", "/RegisterPython=1" -Wait

# 3.3. MikTeX + TeXstudio
Write-Host "Установка MikTeX и TeXstudio..." -ForegroundColor White
choco install miktex -y
choco install texstudio -y
```

```
# 4. Установка WSL2 с Ubuntu
Write-Host "`n4. Настройка WSL2..." -ForegroundColor Cyan
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux -NoRestart
Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform -NoRestart
wsl --set-default-version 2
wsl --install -d Ubuntu-22.04

# 5. Дополнительное ПО для аудитории 313
if ($Audience313) {
    Write-Host "`n5. Установка ПО для аудитории 313..." -ForegroundColor Magenta

    # 5.1. Ramus Educational
    Invoke-WebRequest -Uri "https://drive.google.com/uc?export=download&id=1ygbqxneypuixRftrObaT53raBICUJ6Kxz" -OutFile "$env:TEMP\ramus.exe"
    Start-Process "$env:TEMP\ramus.exe" -ArgumentList "/S" -Wait

    # 5.2. ARIS Express
    Invoke-WebRequest -Uri "https://drive.google.com/uc?export=download&id=1sPOZXSIDDYx-RYqluLbVltrLVy3Qvzk" -OutFile "$env:TEMP\aris.exe"
    Start-Process "$env:TEMP\aris.exe" -ArgumentList "/S" -Wait

    # 5.3. Archi
    choco install archi -y
}
```

```
# 6. Настройка расширений VS Code
Write-Host "`n6. Настройка VS Code..." -ForegroundColor Cyan
$vscodeExtensions = @(
    "ms-python.python",
    "ms-vscode.cpptools",
    "ms-azuretools.vscode-docker",
    "eamodio.gitlens",
    "ms-vscode.powershell",
    "julialang.language-julia",
    "bierner.markdown-preview-github-styling"
)

foreach ($extension in $vscodeExtensions) {
    code --install-extension $extension
}

# 7. Финальная настройка
Write-Host "`n7. Финальная настройка..." -ForegroundColor Cyan

# 7.1. Создание ярлыков на рабочем столе
$desktopPath = [Environment]::GetFolderPath("Desktop")
$softwareShortcuts = @("Visual Studio Code", "PyCharm", "GitHub Desktop", "Docker", "GIMP")

foreach ($shortcut in $softwareShortcuts) {
    $shell = New-Object -ComObject WScript.Shell
    $shortcut = $shell.CreateShortcut("$desktopPath\$shortcut.lnk")
    # Здесь можно настроить пути к ярлыкам
}
```

```
# 7.2. Очистка временных файлов
Remove-Item "$env:TEMP\*.exe" -Force -ErrorAction SilentlyContinue

Write-Host "`n=== УСТАНОВКА ЗАВЕРШЕНА ===" -ForegroundColor Green
Write-Host "Завершено: $(Get-Date)" -ForegroundColor Yellow
Write-Host "Установлено программ: $($softwareList.Count + 5)" -ForegroundColor Green

# Отчет об установке
Get-Process | Where-Object {$_.Name -like "*install*"} | Stop-Process -Force -ErrorAction SilentlyContinue
```

2.2. Скрипт создания бэкапа системы (create-backup.ps1)

```
# Скрипт создания резервной копии системы
# Требует наличия инструментария Windows Server Backup

param(
    [string]$BackupPath = "D:\Backups\",
    [string]$ComputerName = $env:COMPUTERNAME
)

Write-Host "=== СОЗДАНИЕ РЕЗЕРВНОЙ КОПИИ СИСТЕМЫ ===" -ForegroundColor Green

# Проверка прав администратора
if (-NOT ([Security.Principal.WindowsPrincipal] [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] "Administrator")) {
    Write-Host "Требуются права администратора!" -ForegroundColor Red
    exit 1
```

```
}

# Создание директории для бэкапов
if (!(Test-Path $BackupPath)) {
    New-Item -ItemType Directory -Path $BackupPath -Force
}

# Установка компонента Windows Server Backup (если не установлен)
Try {
    Import-Module ServerManager -ErrorAction Stop
    $feature = Get-WindowsFeature -Name Windows-Server-Backup
    if ($feature.InstallState -ne "Installed") {
        Write-Host "Установка Windows Server Backup..." -ForegroundColor Yellow
        Install-WindowsFeature -Name Windows-Server-Backup -IncludeManagementTools
    }
}
Catch {
    Write-Host "Использование встроенных средств бэкапа..." -ForegroundColor Yellow
}

# Создание бэкапа с использованием WBAdmin
$backupName = "SystemBackup_$(Get-Date -Format 'yyyyMMdd_HHmmss')"
$backupCommand = "wbadmin start backup -backupTarget:$BackupPath -include:C: -systemState -allCritical -quiet -vssFull"

Write-Host "Создание резервной копии: $backupName" -ForegroundColor Cyan
Invoke-Expression $backupCommand
```

```
# Проверка успешности создания бэкапа
if ($LASTEXITCODE -eq 0) {
    Write-Host "Резервная копия успешно создана: $BackupPath$backupName" -ForegroundColor Green

    # Создание отчета о бэкапе
    $backupInfo = @{
        "ComputerName" = $ComputerName
        "BackupDate" = Get-Date
        "BackupPath" = $BackupPath
        "BackupName" = $backupName
        "SoftwareInstalled" = @(
            "Visual Studio Code", "Docker", "PyCharm", "Git",
            "Python", "Julia", "Anaconda", "WSL2"
        )
    }

    $backupInfo | ConvertTo-Json | Out-File "$BackupPath\backup_report.json"

    Write-Host "Отчет сохранен: $BackupPath\backup_report.json" -ForegroundColor Green
} else {
    Write-Host "Ошибка при создании резервной копии!" -ForegroundColor Red
}
```

2.3. Пакетный файл для простого запуска (install.bat)

```
@echo off
chcp 65001 >nul
title Установка ПО для компьютерных аудиторий
```

```
echo ============================================
echo    АВТОМАТИЗИРОВАННАЯ УСТАНОВКА ПО
echo ============================================

:: Проверка прав администратора
net session >nul 2>&1
if %errorLevel% neq 0 (
    echo ОШИБКА: Требуются права администратора!
    echo Запустите от имени администратора.
    pause
    exit /b 1
)

:: Запрос на установку ПО для аудитории 313
set /p audience313="Установить ПО для аудитории 313? (y/n): "
if "%audience313%"=="y" (
    powershell -ExecutionPolicy Bypass -File "install-software.ps1" -Audience313
) else (
    powershell -ExecutionPolicy Bypass -File "install-software.ps1"
)

echo.
echo ============================================
echo    УСТАНОВКА ЗАВЕРШЕНА
echo ============================================
echo.
pause
```

3. Комментарии по выполнению

3.1. Технологии и инструменты:

- PowerShell - основной язык скриптов для Windows
- Chocolatey - менеджер пакетов для автоматической установки
- WBAdmin - встроенное средство создания бэкапов Windows
- Параметризация - скрипты поддерживают различные сценарии установки

3.2. Особенности реализации:

- Проверка прав администратора перед выполнением
- Обработка ошибок на каждом этапе установки
- Логирование процесса для отладки
- Поддержка разных аудиторий через параметры

3.3. Преимущества решения:

- Экономия времени - установка 25+ программ за 30-60 минут
- Стандартизация - одинаковый набор ПО на всех компьютерах
- Воспроизводимость - возможность быстрого развертывания
- Документирование - подробные отчеты об установке

4. Инструкция по использованию

- Скачать все файлы скриптов в одну папку
- Запустить install.bat от имени администратора
- Следовать инструкциям на экране
- После установки создать бэкап с помощью create-backup.ps1