Лекція 4

ОПЕРАТОРИ ПЕРЕХОДУ У CMD

Завдання на повторення матеріалу Лекції 3. Написати командний файл, який би створював директорію з ім'ям, зазначеним в якості першого параметра командного рядка в директорії, що зазначена в якості другого параметра командного рядка. Якщо директорія не створена, має бути виведено повідомлення про помилку.

За допомогою команди:

IF ... ELSE

(ключове слово ELSE може бути відсутнім) в пакетних файлах можна виконувати обробку умов декількох типів. При цьому, якщо задана після IF умова приймає істинне значення, система виконує наступну за умовою команду (або кілька команд, укладених в круглі дужки), в іншому випадку виконується команда (або кілька команд в дужках), що записана за ключовим словом ELSE (рис. 4.1.).

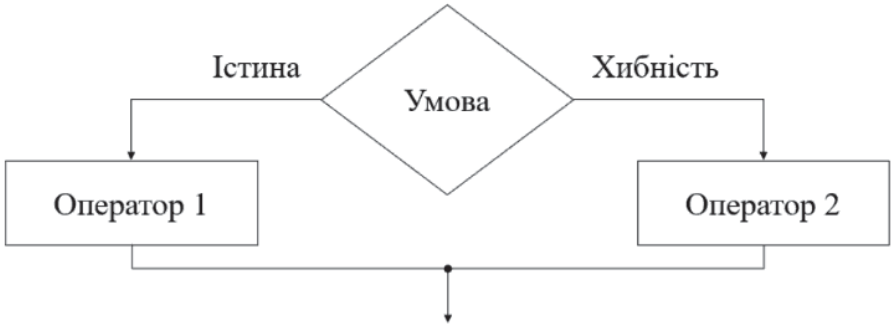

Рис. 4.1. — Принцип роботи умовного оператора IF ... ELSE

Перший тип умови використовується зазвичай для перевірки значення змінної. Для цього застосовуються два варіанти синтаксису команди IF:

IF [NOT] рядок1==рядок2 команда1 [ELSE команда2]

(квадратні дужки вказують на необов'язковість укладених в них параметрів).

Умова *рядок1==рядок2* (тут необхідно писати саме два знака рівності) вважається істинною при точному збігу обох рядків. Параметр NOT вказує на те, що задана команда виконується лише в тому випадку, коли рядки, що порівнюються, не збігаються. Рядки можуть бути літеральнимі або являти собою значення змінних.

Розглянемо приклад перевірки правильності введеного користувачем ключа при запуску командного файлу.

Нехай, якщо користувач ввів перший параметр Hello, то командний файл повинен привітатися з ним. Інакше, командний файл повинен видати повідомлення про помилку (лістинг 4.1.).

```
script.cmd
@ECHO OFF

IF %1==Hello (
ECHO Hello, user
) ELSE (
ECHO User must say hello first
)
```

```
D:\>script.cmd Hello
Hello, user
D:\>script.cmd Action
User must say hello first
```

Лістинг 4.1. — Приклад перевірки правильності введеного користувачем ключа

Важливі зауваження:

1. Оператори команди IF і ключове слово ELSE повинні розташовуватися в одному рядку. Якщо ця умова не виконується, команди після оператора IF і ELSE повинні бути укладені в круглі дужки. Круглі дужки обов'язково повинні бути використані, якщо після оператора IF або оператора ELSE використовуються кілька команд.

2. Можливе застосування вкладених операторів IF. В цьому випадку, вкладений оператор повинен відзначатися символом @ (лістинг 4.2.).

```
script.cmd
@ECHO OFF
IF "%1"=="1" (
@IF "%2"=="2" ( ECHO true ) )
```

```
D:\>script.cmd 1 2
true
D:\>script.cmd 1 3
```

Лістинг 4.2. — Приклад використання вкладеного оператора IF

Відзначимо, що якщо один з аргументів не буде визначено користувачем — це призведе до помилки синтаксису. Щоб обійти це, аргументи потрібно брати в лапки.

Завдання. Написати командний файл, який виводить на екран вітання, якщо введена в якості першого параметра командного рядка дата відповідає сьогоднішній. Інакше командний файл повинен видати повідомлення про помилку.

Існують ще варіанти використання команди IF:

IF [/I] [NOT] рядок1 оператор_порівняння рядок2 команда

В якості оператора порівняння можна використовувати наступні оператори, наведені в табл. 4.1.

Таблиця 4.1. — Допустимі оператори порівняння команди IF

Оператор	**Значення**
EQU	Дорівнює
NEQ	Не дорівнює
LSS	Менше
LEQ	Менше або дорівнює
GTR	Більше
GEQ	Більше або дорівнює

При цьому, якщо рядки містять лише цифри, вони сприймаються як числа.

Наведемо приклад використання операторів порівняння (лістинг 4.3.).

```
script.cmd
@ECHO OFF
CLS

IF "%1" EQU "John" ECHO Hello John!

IF "%1" NEQ "John" ECHO Hello, but you are not John!
```

```
D:\>script.cmd Fox
Hello, but you are not John!

D:\>script.cmd John
Hello John!
```

Лістинг 4.3. — Приклад використання операторів порівняння

Ключ /I, якщо він зазначений, задає порівняння текстових рядків без урахування регістру. Ключ /I можна також використовувати і в формі рядок1==рядок2 команди IF. Наприклад, умова:

IF /I DOS==dos ...

буде істинною.

Відзначимо, що оператором IF і спеціальним записом команди SET з ключем /P зручно створювати меню командних файлів:

SET /P variable=[promptString]

де:

— variable — ім'я змінної куди буде записана інформація з клавіатури;

— promptString — рядок запрошення введення.

Наведемо приклад запиту введення значення змінної (лістинг 4.4.).

```
script.cmd
@ECHO OFF
CLS

SET /p input="Please, enter the value:"
ECHO %input%
```

```
D:\>script.cmd
Please, enter the value:Hello
Hello
```

Лістинг 4.4. — Приклад запиту введення значення змінної

Інший спосіб використання команди IF — це перевірка існування заданого файлу. Синтаксис для цього випадку має вигляд:

IF [NOT] EXIST файл команда1 [ELSE команда2]

Умова вважається істинною, якщо зазначений файл існує. Наведемо приклад командного файлу, в якому за допомогою такого варіанту команди IF перевіряється наявність файлу, вказаного в якості першого параметра командного рядка (лістинг 4.5.).

```
script.cmd
@ECHO OFF
IF "%1"=="" ( ECHO No file Specified
) ELSE (
@IF NOT EXIST "%1" ECHO File do not exist )
```

```
D:\>script.cmd
No file Specified
D:\>script.cmd c:\Windows\System32\drivers\etc\hosts
D:\>script.cmd c:\Windows\System32\drivers\etc\hosts1
File do not exist
```

Лістинг 4.5. — Приклад командного файлу, в якому перевіряється наявність файлу

Аналогічно до файлів команда IF дозволяє перевірити наявність в системі певної змінної середовища:

IF DEFINED змінна команда1 [ELSE команда2]

Тут умова DEFINED застосовується подібно умові EXIST наявності заданого файлу, але приймає в якості аргументу ім'я змінної середовища і повертає істинне значення, якщо ця змінна визначена (лістинг 4.6.).

```
script.cmd
@ECHO OFF
SET a=0
IF DEFINED %1 (
ECHO Variable %1 is defined
) ELSE (
ECHO Variable %1 is not defined )
```

```
D:\>script.cmd a
Variable a is defined
D:\>script.cmd b
Variable b is not defined
```

Лістинг 4.6. — Приклад командного файлу, в якому перевіряється, чи задана змінна

Ще один спосіб використання команди IF — це перевірка коду завершення (коду виходу) попередньої команди. Синтаксис для IF в цьому випадку має такий вигляд:

IF [NOT] ERRORLEVEL число команда1 [ELSE команда2]

Тут умова вважається істинною, якщо остання запущена команда або програма завершилася з кодом повернення, рівним або таким, що перевищує вказане число.

Складемо, наприклад, командний файл, який би копіював файл my.txt на диск C:\ без виведення на екран повідомлень про копіювання, а в разі виникнення будь-якої помилки він має видавати попередження (лістинг 4.7.).

```
script.cmd
@ECHO OFF
COPY my.txt D:\1 > NUL
REM Check of the exit code of the copying
IF ERRORLEVEL 1 (
ECHO An error occurred while executing the command COPY!
) ELSE (
ECHO Copying completed without errors. )
```

```
D:\>script.cmd
An error occurred while executing the command COPY!
D:\>dir > my.txt
D:\>script.cmd
Copying completed without errors.
```

Лістинг 4.7. — Командний файл копіювання файлу з перевіркою успішності виконання операції

Завдання. Видалити файл, вказаний в якості першого параметра командного рядка. Якщо файл не існує, вивести відповідне повідомлення. Якщо перший параметр командного рядка не вказано, вивести відповідне повідомлення.

Розглянемо приклад створення меню командного файлу (лістинг 4.8.).

```
script.cmd
@ECHO OFF
CLS
COLOR A0

ECHO *1* Antivirus scanning
ECHO *2* Removing of temporaly files
ECHO *3* Disk formatting
SET /p input="Please, enter the task number:"
ECHO %input%
COLOR
IF "%input%"=="1" ECHO The first item is selected
IF "%input%"=="2" ECHO The second item is selected
IF "%input%"=="3" ECHO The third item is selected
```

Лістинг 4.8. — Приклад командного файлу з меню

```
*1* Antivirus scanning
*2* Removing of temporaly files
*3* Disk formatting
Please, enter the task number:3
3
The third item is selected
```

Лістинг 4.8. (закінчення) — Приклад командного файлу з меню

Команда GOTO використовується для виконання безумовного переходу в командному файлі. Формат команди:

GOTO мітка

Мітка є рядком символів, що починається з двокрапки. Наприклад:

@ECHO OFF

. . .

GOTO M1

. . .

:M1

GOTO M1 в даному прикладі виконує перехід до мітки M1. Мітка знаходиться в окремому рядку командного файлу і починається з двокрапки.

Відзначимо, що команда GOTO може приймати в якості мітки переходу рядок :EOF (End Of File), який викликає передачу управління в кінець поточного пакетного файлу, що дозволяє легко завершити його виконання без реального задання даної мітки.

Використання даної техніки дозволяє легко завершити командний файл в разі, наприклад, виконання певної умови (лістинг 4.9.).

```
script.cmd
@ECHO OFF

COPY my.txt D:\1 > NUL

REM Check of the exit code of the copying
IF ERRORLEVEL 1 (
ECHO An error occurred while executing the command COPY!
GOTO :EOF )

ECHO Copying completed without errors.
```

Лістинг 4.9. — Приклад командного файлу для копіювання файлу з перевіркою успішності виконання операції з використанням оператора безумовного переходу GOTO

Повернемося до нашого командному файлу (лістинг 4.8.). За допомогою оператора GOTO ми можемо зробити код більш зрозумілим (лістинг 4.10).

```
script.cmd
@ECHO OFF
CLS

COLOR A0

ECHO *1* Antivirus scanning
ECHO *2* Removing of temporaly files
ECHO *3* Disk formatting

SET /p input="Please, enter the task number:"
ECHO %input%

COLOR

IF "%input%"=="1" GOTO :m1
IF "%input%"=="2" GOTO :m2
IF "%input%"=="3" GOTO :m3
```

Лістинг 4.10. — Командний файл з меню із застосуванням оператора безумовного переходу GOTO

```
:m1
ECHO The first item is selected & GOTO m4
:m2
ECHO The second item is selected & GOTO m4
:m3
ECHO The third item is selected & GOTO m4
:m4
COLOR
```

Лістинг 4.10. (закінчення) — Командний файл з меню із застосуванням оператора безумовного переходу GOTO

Відзначимо, що з одного командного файлу можна викликати інший, просто вказавши шлях до нього. Однак, в цьому випадку після виконання викликаного файлу управління в файл з якого здійснено виклик не передається. Для того, щоб викликати зовнішній командний файл з наступним поверненням в початковий файл, потрібно використовувати спеціальну команду:

CALL файл

Приклади виклику командного файлу з використанням команди CALL, а також безпосереднього виклику показані у лістингу 4.11. При цьому у script2.cmd строка коду COPY A:*.* C:\ не буде виконана.

```
script1.cmd
@ECHO OFF

CLS
REM Listing the log-files
DIR C:\*.log

REM The execution will be transfered to f.bat
CALL f.bat
COPY A:\*.* C:\
PAUSE
```

Лістинг 4.11. — Виклик командного файлу з використанням команди CALL, а також безпосередньо

```
script2.cmd
@ECHO OFF
CLS
REM Listing the log-files
DIR C:\*.log
REM The execution will be transfered to f.bat
f.bat
COPY A:\*.* C:\
PAUSE
```

Лістинг 4.11. (закінчення) — Виклик командного файлу з використанням команди CALL, а також безпосередньо

Зауваження. За допомогою команд IF і SHIFT можна в циклі обробляти всі параметри командного рядка, навіть не знаючи заздалегідь їх кількості. Наприклад, командний файл (лістинг 4.12.) виводить на екран ім'я файлу, що запускається і всі параметри командного рядка.

```
script.cmd
@ECHO OFF
ECHO File started with the following parameters...
REM Begining of the loop
:BegLoop
IF "%1"=="" GOTO ExitLoop
ECHO %1
REM shifting parameters
SHIFT
REM Returning to the begining of the loop
GOTO BegLoop
:ExitLoop
REM Exiting from the loop
ECHO.
ECHO This is all.
```

```
D:\>script.cmd 1 2 3
File started with the following parameters...
1
2
3

This is all.
```

Лістинг 4.12. — Командний файл для виводу на екран імені файлу, що запускається і всіх параметрів командного рядка

Завдання. Написати командний файл, який буде записувати в зазначений в першому параметрі командного рядка файл всі рядки, що вводяться користувачем. Кінець введення — рядок, що містить лише одну крапку.

У командних файлах для організації циклів використовуються кілька різновидів оператора FOR, які забезпечують такі функції:

— виконання заданої команди для всіх елементів зазначеної множини;

— виконання заданої команди для всіх відповідних імен файлів;

— виконання заданої команди для всіх відповідних імен каталогів;

— виконання заданої команди для певного каталогу, а також всіх його підкаталогів;

— отримання послідовності чисел із заданими початком, кінцем і кроком збільшення;

— читання і обробка рядків з текстового файлу;

— обробка рядків виводу певної команди.

Найпростіший варіант синтаксису команди FOR для командних файлів має такий вигляд:

FOR %%змінна IN (множина) DO команда [параметри]

Увага! Перед назвою змінної повинні стояти саме два знака відсотка (%%). Відразу наведемо приклад (лістинг 4.13.).

script.cmd
```
@ECHO OFF
FOR %%i IN (One,Two,Three) DO ECHO %%i
```

```
D:\>script.cmd
One
Two
Three
```

Лістинг 4.13. — Приклад використання циклу FOR ... IN ... DO

Параметр *множина* в команді FOR задає один або більше текстових рядків, розділених комами, які Ви хочете обробити за допомогою заданої команди. Дужки тут обов'язкові. Параметр *команда [параметри]* задає команду, виконувану для кожного елемента множини. Якщо в рядку, що входить в множину, використовується кома, то значення цього рядка потрібно взяти в лапки (лістинг 4.14.).

```
script.cmd
@ECHO OFF
FOR %%i IN ("One,Two",Three) DO ECHO %%i
```

```
D:\>script.cmd
"One,Two"
Three
```

Лістинг 4.14. — Приклад використання циклу FOR ... IN ... DO при наявності символу «,» у значенні, що обробляється

Зауваження:

— при виконанні команда FOR послідовно підставляє замість змінної текст кожного рядка в заданій множині, поки команда, що стоїть після ключового слова DO, не обробить всі такі рядки;

— параметр *%%змінна* представляє змінну, що підставляється (лічильник циклу), причому тут можуть використовуватися тільки імена змінних, що складаються з однієї літери;

— щоб уникнути плутанини з параметрами командного рядка %0,…,%9, для змінних слід використовувати будь-які символи крім 0,…,9.

Параметр *множина* в команді FOR може також представляти одну або кілька груп файлів. Наприклад, щоб вивести в файл список всіх файлів з розширеннями exe і dll, що знаходяться в каталозі C:\Windows, без використання команди DIR, можна використовувати командний файл такого змісту (лістинг 4.15.).

```
script.cmd
@ECHO OFF
FOR %%f IN (C:\Windows\*.exe, C:\Windows\*.dll) DO (
ECHO %%f >> list.txt )
```

Лістинг 4.15. — Приклад використання циклу FOR ... IN ... DO для обробки файлів

При такому використанні команди FOR процес обробки триває, поки не будуть оброблені всі файли (або групи файлів), зазначені у множині.

Завдання. Дописати деякий рядок в усі текстові файли в каталозі, вказаному користувачем в якості першого параметра командного рядка.

Наступний варіант команди FOR реалізується за допомогою ключа /D:

FOR /D %%змінна IN (набір) DO команда [параметри]

У разі, якщо *набір* містить групові символи, то команда виконується для всіх відповідних імен каталогів, а не імен файлів. Виконуючи наступний командний файл, ми отримаємо список всіх каталогів на диску C:\ (лістинг 4.16.).

```
script.cmd
@ECHO OFF
CLS
FOR /D %%f IN (C:\*.*) DO ECHO %%f
```

```
D:\>script.cmd
C:\1
C:\Intel
C:\Octave
C:\PerfLogs
C:\Program Files
C:\Program Files (x86)
C:\Temp
C:\Users
C:\VAGTool
C:\VCDS RUS 14.10.0
C:\WCH.CN
C:\Windows
```

Лістинг 4.16. — Приклад використання циклу FOR ... IN ... DO для обробки директорій

За допомогою ключа /R можна задати рекурсію в команді FOR:

FOR /R [[диск:]шлях] %%змінна IN (набір) DO команда [параметри]

В цьому випадку задана команда виконується для каталогу *[диск:]шлях*, а також для всіх підкаталогів цього шляху. Якщо після ключа /R не вказано ім'я каталогу, то виконання команди починається з поточного каталогу. Наприклад, для роздрукування всіх файлів з розширенням txt в поточному каталозі і всіх його підкаталогах можна використовувати наступний пакетний файл (лістинг 4.17.).

```
script.cmd
@ECHO OFF
CLS
FOR /R %%f IN (*.txt) DO PRINT %%f
```

Лістинг 4.17. — Приклад використання циклу FOR ... IN ... DO для роздрукування всіх текстових файлів

Якщо замість набору вказана тільки крапка (.), то команда перевіряє всі підкаталоги поточного каталогу. Наприклад, якщо ми знаходимося в каталозі D:\TEXT з двома підкаталогами BOOKS і ARTICLES, то приклад виконання циклу FOR ... IN ... DO з ключем /R буде мати вигляд, представлений у лістингу 4.18.

```
script.cmd
@ECHO OFF
CLS
FOR /R %%f IN (.) DO ECHO %%f
```

```
D:\TEXT>d:\script.cmd
D:\TEXT\.
D:\TEXT\ARTICLES\.
D:\TEXT\BOOKS\.
```

Лістинг 4.18. — Приклад виконання циклу FOR ... IN ... DO з ключем /R

Ключ /L дозволяє реалізувати за допомогою команди FOR арифметичний цикл, в цьому випадку синтаксис має наступний вигляд:

FOR /L %%змінна IN (початок,крок,кінець) DO команда [параметри]

У цій конструкції задана після ключового слова IN трійка *(початок,крок,кінець)* розкривається в послідовність чисел із заданими початком, кінцем і кроком збільшення. Так, набір (1,1,5) розкривається в (1 2 3 4 5), а набір (5, -1,1) замінюється на (5 4 3 2 1). Приклад наведено у лістингу 4.19.

```
script.cmd
@ECHO OFF

CLS

FOR /L %%f IN (1,1,5) DO ECHO %%f
```

```
D:\>script.cmd
1
2
3
4
5
```

Лістинг 4.19. — Приклад організації арифметичного циклу за допомогою конструкції FOR ... IN ... DO

Числа, одержувані в результаті виконання циклу FOR /L, можна використовувати в арифметичних обчисленнях.

Розглянемо приклад (лістинг 4.20.).

```
script.cmd
@ECHO OFF
CLS

FOR /L %%f IN (1,1,5) DO CALL :2 %%f
GOTO :EOF
:2
SET /A M=10*%1
ECHO 10*%1=%M%
```

Лістинг 4.20. — Приклад організації арифметичного циклу за допомогою конструкції FOR ... IN ... DO

```
D:\>script.cmd
10*1=10
10*2=20
10*3=30
10*4=40
10*5=50
```

Лістинг 4.20. (закінчення) — Приклад організації арифметичного циклу за допомогою конструкції FOR ... IN ... DO

В даному командному файлі ми використали команду CALL не для виклику іншого командного файлу, а для переходу на певну мітку командного файлу з передачею параметрів переходу.

Так, CALL допускає використання міток в якості адресата виклику. Застосовується наступний синтаксис:

CALL :мітка параметри командного рядка

При виклику створюється новий контекст поточного пакетного файлу з заданими аргументами, і управління передається на інструкцію, розташовану відразу після мітки.

Завдання. Видалити всі файли в поточному каталозі. Ім'я кожного файлу, що видаляється, записати в файл-звіт, який повинен знаходитися в кореневому каталозі диска D:\.

Найпотужніші можливості (і одночасно найбільш заплутаний синтаксис) має команда FOR з ключем /F:

FOR /F [ключі] %%змінна IN (набір) DO команда [параметри]

Тут параметр *набір* містить імена одного або декількох файлів, які по черзі відкриваються, читаються і обробляються. Обробка полягає в читанні файлу, розбитті його на окремі рядки тексту і виділенні з кожного рядка заданого числа підрядків. Потім знайдений підрядок використовується в якості значення змінної при виконанні основного тіла циклу (заданої команди).

За замовчуванням ключ /F виділяє з кожного рядка файлу перше слово, очищене від оточуючих його пробілів. Порожні рядки в файлі пропускаються.

Необов'язковий параметр *ключі* служить для перевизначення заданих за замовчуванням правил обробки рядків. Ключі являють собою укладений в лапки рядок, що містить наведені в табл. 4.2. ключові слова.

Таблиця 4.2. — Ключі циклу FOR з ключем /F

Ключ	**Опис**
EOL=C	Визначення символу коментарів на початку рядка (допускається завдання тільки одного символу)
SKIP=N	Число рядків, що пропускаються при обробці на початку файлу
DELIMS=XXX	Визначення набору роздільників для заміни заданих за замовчуванням пробілу і знака табуляції
TOKENS=X,Y,M-N	Визначення номерів підрядків, що виділяються з кожного рядка файлу і передаються для виконання в тіло циклу

При використанні ключа TOKENS=X,Y,M-N створюються додаткові змінні. Формат M-N являє собою діапазон підстрок з номерами від M до N. Якщо останній символ в рядку TOKENS= є зірочкою, то створюється додаткова змінна, значенням якої буде весь текст, що залишився в рядку після обробки останнього підрядка.

Розберемо застосування цієї команди на прикладі пакетного файлу, який виконує розбір файлу myfile.txt (лістинг 4.21.).

script.cmd
```
@ECHO OFF
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN (
myfile.txt) DO ECHO %%i %%j %%k
```

Лістинг 4.21. — Приклад використання циклу для обробки текстового файлу

```
D:\>more myfile.txt
AAA BBBB CCCC, DDDDD EEEE
FFFFF,GGGG HHHH
;IIII JJJJJ KKKKK
```

```
D:\>script.cmd
BBBB CCCC DDDDD EEEE
GGGG HHHH
```

Лістинг 4.21. (закінчення) — Приклад використання циклу для обробки текстового файлу

Зауваження. Ключ TOKENS дозволяє витягти з одного рядка файлу до 26 підрядків, тому заборонено використовувати імена змінних, що не є буквами англійського алфавіту (a…z). Слід пам'ятати, що імена змінних FOR є глобальними, тому одночасно не може бути активно понад 26 змінних.

Команда FOR /F також дозволяє обробляти окремий рядок. Для цього слід ввести потрібний рядок в лапках замість набору імен файлів в дужках. Рядок буде оброблений так, як ніби він був взятий з файлу (лістинг 4.22.).

```
script.cmd
@ECHO OFF
SET M=AAA BBBB CCCC,DDDDD EEEE
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN (
"%M%") DO ECHO %%i %%j %%k
```

```
D:\>script.cmd
BBBB CCCC DDDDD EEEE
```

Лістинг 4.22. — Використання циклу FOR для обробки окремого рядка

Нарешті, команда FOR /F дозволяє обробляти рядок виводу іншої команди. Для цього слід замість набору імен файлів в дужках ввести рядок виклику команди в апострофах (не в лапках!). Рядок передається для виконання інтерпретатора команд cmd.exe, а виведення цієї команди записується в пам'ять і обробляється так, як ніби рядок виведення взятий з файлу. Наприклад, наступний командний файл

виведе перелік імен всіх змінних середовища, визначених в даний час в системі (лістинг 4.23.).

```
script.cmd
@ECHO OFF
CLS
ECHO Environment variable names
ECHO.
FOR /F "DELIMS==" %%i IN ('SET') DO ECHO %%i
```

```
D:\>script.cmd
Environment variable names

ALLUSERSPROFILE
APPDATA
CommonProgramFiles
CommonProgramFiles(x86)
CommonProgramW6432
COMPUTERNAME
```

Лістинг 4.23. — Використання циклу FOR для обробки виведення команди

Домашнє завдання до Лекції 4.

Написати командний файл, який буде створювати в директорії, яка задана в якості першого параметра командного рядка, N (вводиться в якості другого параметра командного рядка) порожніх файлів, з іменами, що відповідають їх порядковому номеру.