II. Programowanie wsadowe DOS/Windows

Instrukcję przygotowano na podstawie materiałów udostępnionych przez kolegów z Zakładu Sterowania Wydziału Elektrycznego (Obecnie oryginał instrukcji znajduje się na stronie http://www.ee.pw.edu.pl/~graniszw), plików pomocy do systemu Windows oraz materiałów własnych.

1. Wstęp

Program wsadowy jest to ciąg poleceń trybu linii komend lub wywołań programów zapisany w pliku tekstowym o rozszerzeniu `.bat` Celem programowania wsadowego jest:

- przyspieszenie pracy
- personalizacja środowiska systemu operacyjnego

W języku programowania wsadowego można stosować wszystkie komendy dostępne w trybie linii komend. Do sterowania działaniem programu wsadowego służą komendy sterujące:

`call` - uruchamia programu wsadowego z poziomu innego programu wsadowego, a następnie powrót do programu źródłowego (odpowiednik procedury w językach wyższego rzędu)

`echo` - wyświetla komunikaty na ekranie, oraz może służyć do włączania lub wyłączania wyświetlania komunikatów

`for` - powtarza komendę dla zestawu zmiennych

`goto` - przechodzi do innej części programu

`If` - wykonuje komendy w zależności od spełnienia warunku

`pause` - zatrzymuje wykonanie programu do czasu wciśnięcia dowolnego klawisza

`rem` - komentarz

`shift` - przesuwa parametry programu (%9→%8,%8→%7 itd.)

W dalszej części instrukcji zostaną szczegółowo omówione powyższe instrukcje.

Informacje o sposobie wywołania komendy uzyskuje się poprzez wpisanie nazwy komendy z parametrem `/?` np.: `if /?`. Nazwy programów wsadowych podlegają tym samym ograniczeniom, jak nazwy plików w systemie - dopuszczalne są długie nazwy. Programy wsadowe uruchamia się podając w trybie interaktywnym nazwę pliku wsadowego (najlepiej łącznie z rozszerzeniem). Plik wsadowy można także uruchomić w trybie okienkowym wybierając go poprzez podwójne (lub jednokrotne - zależnie od ustawień) kliknięcie klawiszem myszki - tak jak każdy inny program. Zatrzymać wykonanie programu wsadowego można poprzez naciśnięcie kombinacji klawiszy `Ctrl+S` lub `Pause`. Wykonanie programu jest kontynuowane po naciśnięciu dowolnego klawisza. Przerwać działanie programu wsadowego można poprzez naciśnięcie kombinacji klawiszy `Ctrl+C` lub `Pause`.

2. Używanie Notatnika do tworzenia skryptów.

Skrypty są plikami tekstowymi, więc do ich tworzenia można używać programu `Notatnik`. Należy jednak uważać na jedną cechę tego programu: podczas zapisywania dodaje on rozszerzenie `txt` do nazwy programu, np. zapisując pod nazwę `skrypt.bat` program zapisze `skrypt.bat.txt`, czyli NIE jest to skrypt. Aby `Notatnik` nie dodawał rozszerzenia, to przy pierwszym zapisie (tylko wtedy podajemy nazwę) oprócz wpisania nazwy i rozszerzenia, np. `skrypt.bat` należy na oknie

dialogowym wybrać opcję Wszystkie pliki w okienku Zapisz jako typ. Uchroni to przed dodaniem rozszerzenia txt.

Przy przeglądaniu plików za pomocą standardowego narzędzia „Mój komputer" są ukryte znane przez system operacyjny rozszerzenia plików, co utrudnia podejrzenie prawdziwego rozszerzenia pliku. Aby wyłączyć tą opcję, należy wybrać z menu Narzędzia→Opcje folderów... a następnie na zakładce Widok odznaczyć opcję Ukryj rozszerzenia plików znanych typów.

3. Wyświetlanie komunikatów

Komunikaty w programach wsadowych można wyświetlać korzystając z instrukcji echo. Składnia tego polecenia:

```
echo <Treść komunikatu>
```

Aby wyświetlić pustą linię należy użyć polecenia

```
echo.
```

(z kropką na końcu, bez spacji).

Przykład

```
echo Program przesuwa pliki z katalogu KAT1 do KAT2
move KAT1 KAT2
dir KAT2
```

Aby dany tekst nie wyświetlać na ekranie, ale aby zapisać do pliku, należy posłużyć się symbolami przekierowania > oraz >> omówionymi w instrukcji do pierwszego ćwiczenia.

W celu wyświetlania dłuższych tekstów, korzystniej jest zapisać treść w pliku tekstowym i wyświetlić go używając instrukcji type - przyspiesza to pracę, szczególnie w sieciach.

Komenda echo może być użyta do ograniczenia wyświetlania wykonywanych instrukcji programu. I tak, aby wyłączyć wyświetlanie wykonywanych instrukcji należy użyć instrukcji:

```
echo off
```

Aby włączyć wyświetlanie wykonywanych instrukcji:

```
echo on
```

Efekt podobny do użycia echo off dla pojedynczej instrukcji można osiągnąć poprzedzając instrukcję znakiem @. Warto wspomnieć, że wywołując polecenie echo off pojawi się na ekranie ta instrukcja (bo „echo" nie jest jeszcze wyłączone), dlatego najczęściej jest wywoływane polecenie @echo off. Zazwyczaj nie jest pożądane wyświetlanie poszczególnych poleceń ze skryptu, dlatego też większość skryptów będzie się zaczynać od tego polecenia.

Przykład

```
@echo off
echo Program kopiuje pliki tekstowe z katalogu KAT1 do
KAT2
copy KAT1 KAT2
dir KAT2
```

4. Komenda `pause`

Zatrzymanie programu można wymusić korzystając z komendy `pause`. Instrukcja `pause` wyświetla komunikat

```
Naciśnij dowolny klawisz aby kontynuować...
```

a następnie oczekuje na wciśnięcie dowolnego klawisza.

Przykład

```
@echo off
echo Program kasuje pliki tekstowe w KAT1
echo Zaczynam kasować...
pause
del KAT1\*.txt
cls
dir KAT1
```

W celu pominięcia wyświetlania komunikatu, wyjście komendy `pause` można przekierować na wyjście puste: `nul`, tak jak na poniższym przykładzie:

```
echo Naciśnij dowolny klawisz
pause > nul
```

Ćwiczenia do wykonania

1. W katalogu, do którego masz prawo do zapisu, utwórz skrypt o nazwie `a.bat`
2. Skrypt ma utworzyć następującą strukturę katalogów:

```
.
     kat1
     kat2
          podkatalog jeden
```

   (proszę zachować spację w nazwie podkatalogu)
3. W każdym podkatalogu proszę utworzyć plik `nazwa.txt`, który będzie zawierał ścieżkę do tego katalogu (należy wykorzystać polecenie `cd` i przekierowanie strumienia)

5. Etykieta i skok do etykiety `goto`

Etykieta jest to nazwa danego wiersza w skrypcie. Etykiety definiuje się pisząc nazwę etykiety poprzedzoną dwukropkiem np.:

```
:etykieta
```

Poprzez etykiety i instrukcje skoku `goto` można zmienić kolejność wykonywania instrukcji programu. Składnia polecenia skoku wygląda następująco:

```
goto etykieta
```

Przykład:

```
echo Zaraz zostanie wywołana instrukcja skoku
```

```
goto et1
echo To polecenie zostanie pominięte
:et1
```

6. Programy z parametrami i polecenie `shift`

Programy wsadowe mogą być wywoływane z parametrami tak jak zwykłe programy/polecenia, np. dla polecenia copy `plik1 plik2` pierwszym argumentem jest `plik1`, a drugim `plik2`. Dostęp do parametrów z poziomu programu wsadowego odbywa się przez użycie symboli %0 do %9. Symbol %0 oznacza nazwę wykonywanego programu wsadowego. Symbole od %1 do %9 odpowiadają kolejnym parametrom. W sposób prosty można się odwołać do dziewięciu parametrów.

Przykład

```
@echo off
echo Program kopiuje pliki tekstowe
pause
copy %1\*.txt %2
cls
dir %2 /p
```

Aby móc odczytać kolejne parametry NIE można posłużyć się symbolem `%10`, ponieważ przez system zostanie to zrozumiane jako parametr pierwszy, do którego zostanie doklejone 0, czyli dla wywołania:

```
Skrypt.bat Ala Ola 4 5 6 7 8 9 10 11
```

parametry będą interpretowane w następujący sposób

```
%1 = Ala
%10 = Ala0
```

Aby móc korzystać z kolejnych parametrów należy użyć polecenia `shift`, które przesuwa kolejność argumentów o jeden, tzn. parametr `%1` staje się parametrem `%0`, parametr `%2` staje się `%1` itd., natomiast niedostępny wcześniej parametr dziesiąty staje się parametrem `%9` (poprzednia wartość parametru `%0` zostanie „skasowana").

Przykład

```
@echo off
echo %0
shift
echo %0
shift
echo %0
```

Po wywołaniu skryptu poleceniem

```
Skrypt.bat Ala Ola Ela
```

wyświetli się następujący wynik:

```
Skrypt.bat
Ala
```

```
Ola
```

7. Instrukcja warunkowa `if`

W programach wsadowych można używać instrukcji warunkowej `if`, której ogólna zasada sprowadza się do sprawdzenia określonego warunku logicznego i przy jego spełnieniu wykonania dowolnej instrukcji.

`if [not] errorlevel numer instrukcja`

(Nawiasy [] oznaczają część opcjonalną.)
Ta postać instrukcji `if` wykona instrukcję `instrukcja` jeżeli kod wyjściowy z poprzedniego programu (`errorlevel`) [nie] jest mniejszy niż numer.

Przykład

```
copy nie_ma_tego_pliku.txt plik2.txt
if errorlevel 1 echo Nie udało się skopiować
```

`if [not] ciąg1==ciąg2 instrukcja`

Wykonanie instrukcji `instrukcja`, jeżeli ciąg znaków `ciąg1` [nie] jest równy ciągowi `ciąg2`. Aby ustrzec się przed błędami najlepiej jest porównywać ciągi znaków wzięte w cudzysłów. Umożliwi to porównywanie pustego ciągu znaków (dwa cudzysłowy obok siebie) oraz ciągów znaków ze spacjami. Jeżeli jeden z ciągów znaków jest parametrem wejściowym, to może (ale nie musi) być wzięty w cudzysłów, wówczas ciąg znaków będzie w "podwójnym cudzysłowie". Aby się przed tym ustrzec, należy odwoływać się do parametru poprzez `%~1` (znak tylda ~) – więcej na ten temat w dalszej części instrukcji.

Przykład

```
if "%~1"=="kopiuj" goto FCOPY
if "%~1"=="przesun" goto FMOVE
goto EXIT
:FCOPY
copy KAT1 KAT2
goto EXIT
:FMOVE
move KAT1 KAT2
:EXIT
```

`if [not] exist nazwa_pliku instrukcja`

Wykonanie instrukcji `instrukcja` jeżeli [nie] istnieje plik: `nazwa_pliku`.

Ćwiczenia do wykonania

1. W katalogu, w którym utworzyłeś skrypt `a.bat`, utwórz skrypt o nazwie `b.bat`
2. Skrypt musi być wywoływany z jednym parametrem. Jeżeli uruchomiono skrypt bez podania parametru, wówczas skrypt ma wyświetlić ostrzeżenie i zakończyć działanie.

3. W zależności od wartości tego argumentu skrypt ma wykonać następujące działania:
   dla a – wypisać na ekranie zawartość bieżącego katalogu
   dla b – otworzyć w domyślnej przeglądarce stronę wydziału elektrycznego
   dla c – zapisać do pliku o nazwie 000.txt</I> bieżącą datę i godzinę
4. W przypadku, gdy argument nie jest równy żadnemu z powyższych, skrypt ma wyświetlić odpowiedni komunikat o błędzie i po naciśnięciu dowolnego klawisza skrypt ma zakończyć działanie.

8. Instrukcja for

FOR [opcja] [%% | %]x IN (zbiór) DO polecenie

Instrukcja `for` powtarza komendę dla każdej pozycji w zadanym zbiorze, przypisując danej zmiennej `x` kolejne wartości z tego zbioru. W ten sposób umożliwia ona np. uruchomienie określonego polecenia dla każdego pliku znajdującego się w zbiorze.
Zmienna `x` (może to być oczywiście inna nazwa) jest zmienną sterującą. W przypadku, gdy komenda `for` jest wpisywana w linii poleceń, to nazwa zmiennej musi być poprzedzona jednym znakiem %. W przypadku, gdy komenda ta znajduje się w pliku wsadowym, wymagane jest poprzedzenie jej %%.

FOR %I IN (C:*.*) DO @echo %I

To polecenie wyświetli listę plików (tylko plików) o dowolnej nazwie z katalogu głównego `c:\`. Proszę zauważyć znak @ przed poleceniem `echo`: bez tego znaczka za każdym razem pojawiałoby się wykonywane polecenie, a następnie wynik tego poleceni. Znak po prostu @ eliminuje wyświetlanie polecenia.
Skrypt zawierający tę instrukcję mógłby wyglądać tak:

```
@echo of
FOR %%I IN (C:\*.*) DO echo %%I
echo on
```

W tym przypadku należy zwrócić uwagę na fakt poprzedzenia zmiennej `I` dwoma znakami procent. Ponadto przed poleceniem `echo` nie ma znaku @, ponieważ wyświetlanie polecenia zostało wyłączone wcześniej poleceniem `echo off`.

FOR /D %I IN (C:*.*) DO @echo %I

Przełącznik /D powoduje, że zamiast plików wyszukiwane są katalogi.

FOR /R "%USERPROFILE%\Menu Start" %I IN (*.*) DO @echo %I

Przełącznik `/R` powoduje, że wyszukiwanie plików (brak przełącznika `/D`) jest rozpoczynane od katalogu `"%USERPROFILE%\Menu Start"` i kontynuowane we wszystkich podkatalogach. Zmienna środowiskowa `%USERPROFILE%` określa ścieżkę dostępu do katalogu z profilem użytkownika, może być rozwinięta do `C:\Documents and Settings\user`. Dodanie `\Menu Start` spowodowało, że ścieżka startowa wygląda następująco `C:\Documents and Settings\user\Menu start`. Ponieważ w nazwie katalogu `Menu start` jest spacja, dlatego konieczne jest wzięcie ścieżki w cudzysłów.

Instrukcja FOR umożliwia ponadto traktowanie zbioru jako zbioru liczb całkowitych (przełącznik /L) oraz przetwarzanie danych z pliku (przełącznik /F). Zainteresowanych odsyłam do pliku pomocy dla instrukcji FOR.

9. Zaawansowane odwoływanie się do parametrów wywoływania skryptu.

Jeżeli parametrem wywołania skryptu %1 jest nazwa pliku, a program na tej podstawie musi określić nazwę katalogu, "czystą" nazwę pliku i oddzielnie rozszerzeniem pliku, wówczas należy posłużyć się następującą składnią:

- %~1 - rozwija %I usuwając wszystkie obejmujące cudzysłowy (")
- %~f1 - rozwija %I do pełnej nazwy ścieżki
- %~d1 - rozwija %I tylko do litery dysku
- %~p1 - rozwija %I tylko do ścieżki
- %~n1 - rozwija %I tylko do nazwy pliku
- %~x1 - rozwija %I tylko do rozszerzenia pliku

Charakterystyczny jest tutaj znak ~ (tylda), który usuwa cudzysłowy.

Przykład

```
@echo off
if not exist "%~1" goto end
echo copy %1 "%~d1%~p1%~n1.bak"
:end
echo on
```

Ćwiczenia do wykonania

1. W katalogu, w którym utworzyłeś skrypt a.bat, utwórz skrypt o nazwie c.bat
2. Skrypt ma utworzyć w katalogu bieżącym plik o nazwie wynik.log, do którego zapisze bieżącą godzinę
3. Skrypt ma wyszukać wszystkie pliki o rozszerzeniu txt znajdujące się w katalogu bieżącym i jego podkatalogach.
4. Dla każdego znalezionego pliku skrypt ma wykonać następujące trzy rzeczy:
    a) wypisać nazwę pliku wraz ze ścieżką dostępu na ekranie
    b) dopisać nazwę pliku wraz ze ścieżką dostępu do utworzonego wcześniej pliku wynik.log
    c) skopiować plik do tego samego katalogu pod tą samą nazwę, ale z innym rozszerzeniem: bak