# Ćwiczenia 2

## Słowniczek

- wscript.echo
- wscript.quit
- cint
- clng
- cdbl
- inputBox
- do while ... loop
- if then ... else ... end if

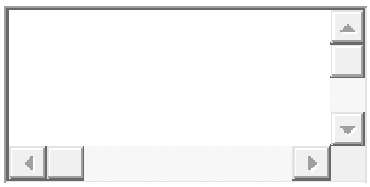
Tak w skrócie prezentują się wszystkie komendy prezentowane w poprzednich ćwiczeniach. Uważna osoba mogła by stwierdzić, że z różnych komend korzysta się w różny sposób. Tak jest w istocie – ponieważ są trzy podstawowe grupy poleceń w tym języku programowania:

- Procedury Systemowe
- Funkcje
- Wyrażenia

## Procedury Systemowe

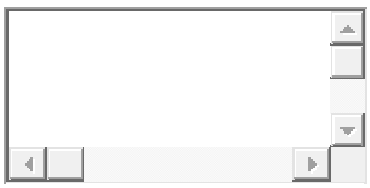
Procedury Systemowe służą do wykonywania czynności systemowych, nie oczekując od nich żadnej informacji zwrotnej – np. ‘wypisz na ekran’, ‘zakończ działanie programu’. Dzięki nim można również odwoływać się do plików w systemie, tworzyć obiekty systemowe etc (ale to już zaawansowane funkcje, o których tu nie będzie mowy). Ich cechą charakterystyczną jest słówko ‘wscript.’ na początku. Świadczy ono o tym, że polecenie jest wykonywane bezpośrednio przez mechanizm WSH.

Ogólny kontekst użycia:



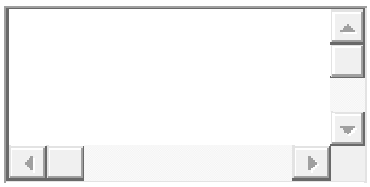
1 WScript.polecenie "opcjonalnie ciąg znaków"

np.:



1 WScript.echo "zakończ program"

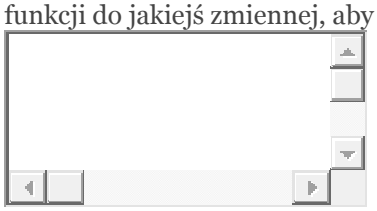
lub:



1 WScript.quit

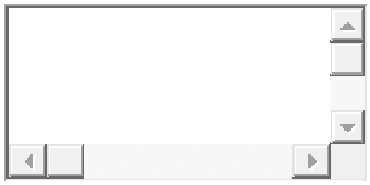
# Funkcje

Funkcje są najpowrzechniej używanymi poleceniami w językach programowania. To dzięki nim można przetwarzać dane, robić obliczenia, prowadzić dialog z użytkownikiem etc. O ile **zmienne**i **stałe** można potraktować jak **rzeczowniki**, o tyle **Funkcje** są odpowiednikami **czasowników**, czyli służą do wykonywania pewnych czynności. Cechą charakterystyczną funkcji jest to, że**zwraca ona jakąś wartość**. Dokonuje ona jakiejś czynności, w wyniku której otrzymamy wartość zwrotną – np. funkcją jest dodawanie kilku liczb. W wyniku działania tej funkcji otrzymamy pojedynczą wartość liczbową. Dla tego też ogólnym, najczęstrzym kontekstem użycia będzie przypisanie wyniku działania funkcji do jakiejś zmiennej, aby przechować wynik do dalszych obliczeń:



```
1 zmienna=jakasFunkcja(parametr1,parametr2)
```

np:



```
1 imie=inputBox("podaj imie")
```

gdzie “podaj imie” zostanie wyświetlone jako komunikat, użytkownik będzie mógł wpisać ciąg znaków, który ‘zostanie zwrócony przez funkcję’ – czyli to co wpisze użytkownik, zostanie przekazane, zapisane w zmiennej o nazwie ‘imie’.

```
1 liczba=cint("2")
```

Ten przykład wymaga dodatkowego wyjaśnienia. Dla komputera istnieje bardzo poważna różnica pomiędzy **“2”** a **2**. Pierwszy zapis oznacza **znak ’2′** – identycznie jak znak ‘a’ czy znak spacji. Dla człowieka ’2′ oznacza prawie zawsze ‘**wartość dwa**‘. Różnica staje się oczywista, jeśli spróbuje się zsumować liczby (wartości) lub znaki:

“2″ + “2″ = “22″ tak jak “a” + “b” = “ab”

w tym przypadku znak ‘+’ oznacza konkatenację, czyli łączenie ciągów znaków

2 + 2 = 4

w tym przypadku znak ‘+’ oznacza operator matematyczny ‘suma’.
Jeśli więc w trakcie pisania programu nie jest się pewnym jak zostanie potraktowana liczba – jako znak czy jako wartość – warto komputerowi to podpowiedzieć. Funkcja ‘**cint**‘ (skrót od Convert to INTeger) jako parametr przyjmuje znak, a zwraca wartość liczbową. Czyli w efekcie tego działania, otrzymamy zmienną ‘liczba’, która będzie przetrzymywać wartość dwa, będąc pewnym, że chodzi właśnie o wartość a nie o znak ’2′.

Język oferuje wiele funkcji, dzięki którym użytkownik może pisać własne programy. Podczas pisania programu, można pisać własne funkcje, rozszeżając tym samym możliwości języka, na czas działania programu.

# Wyrażenia

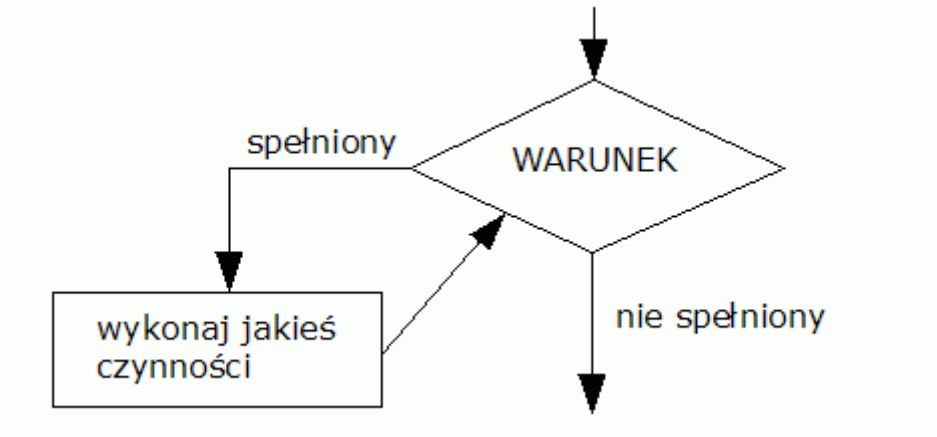
Wyrażenia służą do wykonywania specjalnych instrukcji, zmieniających sposób wykonywania programu. Są to instrunkcje pętli (wykonanie pewnego bloku programu wielokrotnie), wyrażenia warunkowe (jeśli ‘coś’ to zrób to, a jeśli ‘nie coś’ to zrób coś innego) i inne.

## Wyrażnie warunkowe a pętla

W poprzednich ćwiczeniach pokazany był przykład użycia pętli – ‘póty póki’ czyli ‘do while’:

```
1 <b>do while</b> WARUNEK
2   BLOK INSTRUKCJI
3 <b>loop</b>
```

Ogólny schemat blokowy takiej pętli wygląda tak:

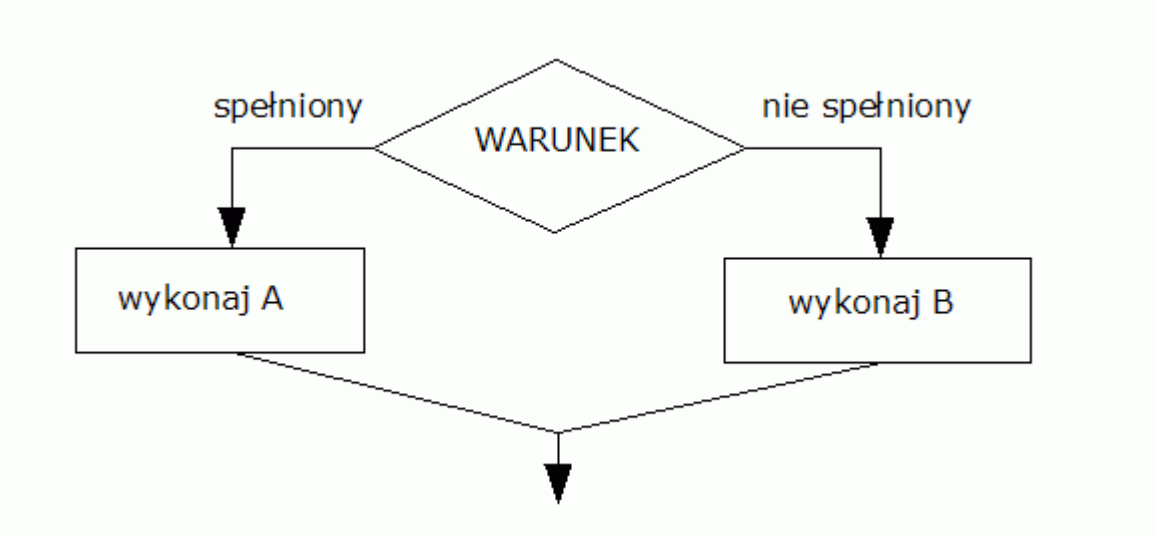



- sprawdzany jest warunek (np. czy i < 10? ) wpisany po ‘do while’
- jeśli warunek jest spełniony, to wykonywany jest blok instrukcji aż do polecenia ‘loop’
- program wraca do wystąpienia ‘do while’ [1] aby znów sprawdzić warunek
- jeśli warunek jest niespełniony, program omija wszystko aż do linii ‘loop’ i zaczyna dalej wykonywać instrunkcje.

Jak widać instrukcję taką stosuje się do wielokrotnego wykonania jakiejś czynności.

Wyrażenie warunkowe ‘if’ stosuje się, jeśli chce się jednorazowo wykonać pewną czynność ‘pod warunkiem, że’:

```
1   <b>if</b> WARUNEK then
2     BLOK INSTRUKCJI 1
3   <b>else</b>
4     BLOK INSTRUKCJI 2
5   <b>end if</b>
```




- sprawdzany jest warunek (np. czy i < 10? )
- jeśli warunek jest spełniony, wykona się COŚ i program przejdzie do dalszego wykonywania instrukcji po słówku ‘end if’
- jeśli warunek jest niespełniony, wykona się czynność zdefniowana po słówku ‘else’. Następnie program zacznie wykonywać instrunkcje po słówku ‘end if’.

Korzystanie z pętli ‘do while’ może być niebezpieczne – jeśli z jakiegoś powodu warunek zawsze będzie spełniony, program będzie w nieskończoność wykonywał blok instruncji w pętli. Np:

```
1 i=1
2 do while i&lt;&gt;10
3   i=i+2
4 loop
```

warunek ‘i różne od 10’ będzie zawsze spełniony, ponieważ i będzie zawsze liczbą nieparzystą. spowoduje to wykonywanie się pętli w nieskończoność. Jeśli chce się wykonać blok instrukcji określoną ilość razy, lepiej skorzystać z pętli ‘for’:

```
1 for i=0 to 10 do
2   BLOK INSTRUKCJI
3 next
```

co oznacza ‘dla i równego 0 aż do 10 wykonuj:’. Kiedy zatem używać ‘do while’ a kiedy ‘for’?

## zadanie

Sprawdź jak działa instrukcja ‘for’ wykonując poniższy kod programu:

```
1 for i=0 to 10
2   wscript.echo i
3 next
```

Korzystając z instrukcji ‘inputBox’ oraz ‘cint’ prezentowanych w poprzednich ćwiczeniach oraz instrukcji ‘for’, napisz program, który będzie wypisywał liczby od zera aż do podanej przez użytkownika.

Następnie, korzystając z instrukcji ‘do while’ oraz ‘inputBox’ napisz program, który będzie się pytał o użytkownika o hasło tak długo, aż nie poda prawidłowego.