



AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE

## Tworzenie i korzystanie z modułów

Piotr Wydrych

<http://www.kt.agh.edu.pl/~wydrych/>

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki  
Katedra Telekomunikacji

30 kwietnia 2009

1/20



## Przestrzenie nazw i pakiety

- Przestrzeń nazw — organizacja zmiennych i funkcji.
- Pierwszy krok w kierunku programowania obiektowego.
- W Perlu implementowane przez pakiety (packages).
- W Perlu nie ma zmiennych globalnych.
- Skrypt rozpoczyna działanie w pakiecie `main`.

### Zmiana bieżącego pakietu na **NAMESPACE**

`package NAMESPACE`

5/20



## AGH package NAMESPACE

- Obowiązuje do końca bloku lub do następnej deklaracji `package`.
- Do zmiennej `$var` w pakiecie `Pkg` odwołujemy się z dowolnego pakietu `$Pkg::var`.
- Do identyfikatorów w pakiecie `main` można odwołać się używając pustej nazwy pakietu.
- Wszystkie następne identyfikatory (poza deklarowanymi przez `my`) odnoszą się do obecnej przestrzeni nazw.
  - ! `$Pkg::var` nie jest identyfikatorem. Jest nazwą zmiennej.
- Kilka plików może korzystać z jednego pakietu.
- Jeden plik może korzystać z kilku pakietów.

6/20



## AGH Zagnieżdżanie pakietów

- Pakiet może zawierać się w innym pakiecie:  
`$Pkg1::Pkg2::var`.
- Odwołania muszą zawsze być pełne.
  - `$Pkg2::var` zawsze odwołuje się do tej samej zmiennej, nawet jeżeli wywołane zostanie z wewnątrz pakietu `Pkg1`.
- Każdy pakiet zawiera się w `main`.
  - `$::var` to to samo co `$main::main::var`
  - `$Pkg1::var` to to samo co `$main::Pkg1::var`

7/20



### Przykład

```
our $x = 'main';

package P1::P2;
our $x = 2;

package P1;
our $x = 1;

print $x;                # 1
print $main::x;         # main
print $::x;              # main
print $P1::x;           # 1
print $main::P1::x;     # 1
print $P2::x;           #
print $P1::P2::x;      # 2
```

8/20



Pakiet umieszczony w pliku o takiej nazwie jak pakiet (po zamianie :: na /) z rozszerzeniem pm.

- Moduł podczas dołączania może wyeksportować swoje symboli do bieżącego pakietu.
- Moduł może domyślnie eksportować określone symboli.
- Moduł zezwala na eksportowania jedynie wybranych symboli (w szczególności — żadnych).
- Najczęściej eksport wspomagany jest przez moduł `Exporter`.

### Import domyślnie eksportowanych symboli

```
use Modul;
```

### Import wybranych symboli

```
use Modul qw(funkcja1 $skalar %hash);  
use Modul ('funkcja1', '$skalar', '%hash');
```

### Wyłączenie importu

```
use Modul ();
```



### Modul.pm

```
package Modul;  
  
require Exporter;  
  
our @ISA = qw(Exporter);  
our @EXPORT = qw(funkcja1 funkcja2);  
our @EXPORT_OK = qw(funkcja3 @tablica);  
  
# inicjowanie zmiennych i deklarowanie funkcji  
our @tablica = ();  
sub funkcja1 { ... };  
  
# moduł musi zwrócić na koniec wartość inną niż fałsz  
1;
```



- Zazwyczaj nie ma możliwości wywołania nie zadeklarowanej funkcji.
- Jeżeli pakiet, do którego należałaby wywoływana nieistniejąca funkcja, posiada funkcję `AUTOLOAD`, to jest ona wykonywana.



### Przykład

```
package Autoladowanie;

use strict;
use warnings;




require Exporter;
use subs qw(pierwsza trzecia);

our @ISA = qw(Exporter);
our @EXPORT = qw(pierwsza druga);
our @EXPORT_OK = qw(trzecia czwarta);

sub AUTOLOAD {
    print $Autoladowanie::AUTOLOAD, '(', join(',_', @_), ")\n";
}
```

16/20



-  Autor nieznany  
perlmod - Perl modules (packages and symbol tables).  
<http://perldoc.perl.org/perlmod.html>.
-  Autor nieznany  
perlmodlib - constructing new Perl modules and finding existing ones.  
<http://perldoc.perl.org/perlmodlib.html>.
-  K. Robert  
perlmodstyle - Perl module style guide.  
<http://perldoc.perl.org/perlmodstyle.html>.

17/20