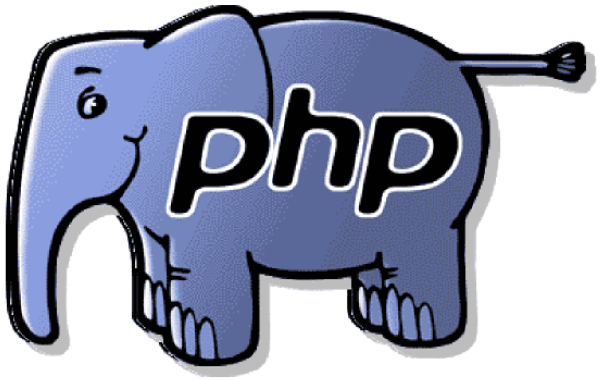
Podstawy php

G. Przęczek

STAŁE

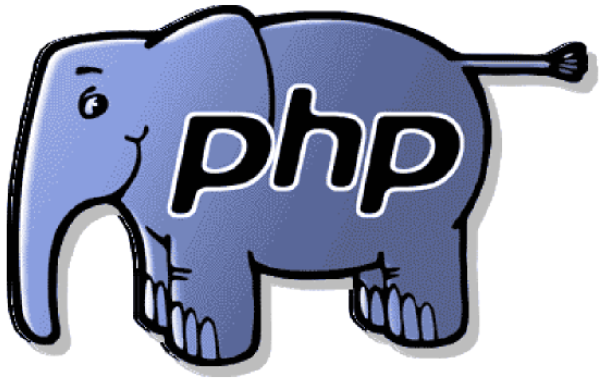
W języku php stałe definiujemy wywołując funkcje **Define**
Ma ona 3 parametry:
$name
$value
$case_insensitive
Dwa pierwsze są wymagane trzeci opcjonalny
Parametr$ name jest napisem. Ustala on nazwę definiowanej stałej
Parametr $value może być dowolnego typu skalarnego



W przypadku funkcji Define może być:

Liczbą całkowitą
define(‘Liczb_stron’,4);

Napisem
defnie(‘Tytuł,’’Jaś i Malgosia’);



Wartością logiczną
define (‘Czy drukować tytuł’,true);

Warto pamiętać

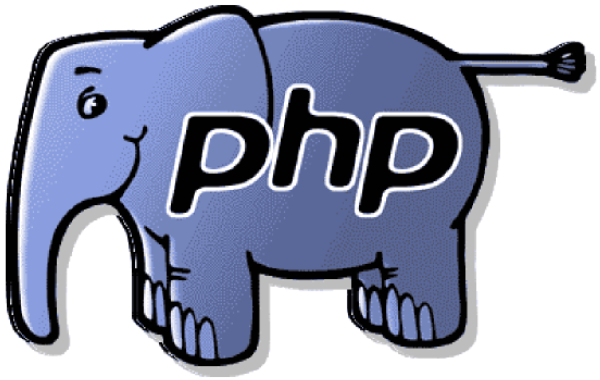
Wielkość liter w nazwach stałych jest odrózniana
Funkcja define zwraca wynik logiczny: true w przypadku powodzenia false w przypadku błędu błędem zakończy się np. próba dwukrotnego zdefiniowania stałej o tej samej nazwie

Po zdefiniowaniu stałej

define(‘Liczb_stron’,4);

jej wartość jest dostępna globalnie w całym skrypcie.Wartość stałej możemy wydrukować

echo(‘Liczb_stron’);

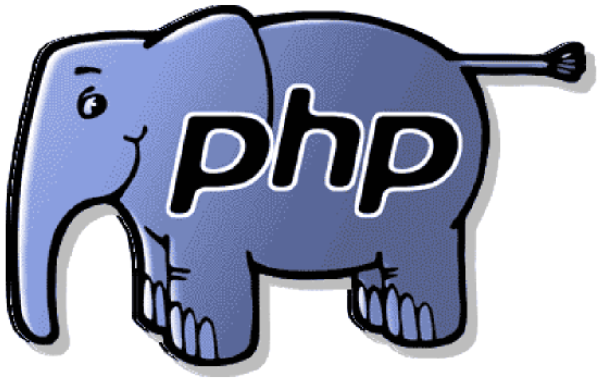


Zmienne

Zmienne w Php nie są deklarowane
Przed nazwą zmiennej występuje znak $
Zasięg zmiennej jest lokalny

Zmienną możemy wprowadzać w dowolnej części skryptu. Operator = przypisuje zmiennej określoną wartość
Wartość zmiennej możemy wydrukować za pomocą instrukcji echo

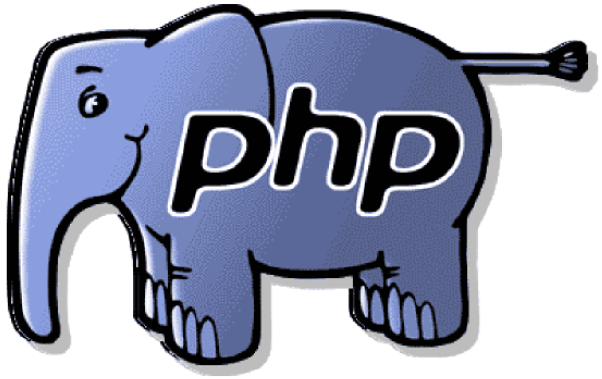
```
$a = 1;
echo ($a);
```



Wartość i typ zmiennej możemy sprawdzić za pomocą funkcji **var_dump()**

W wyniku wykonania skryptu

```
<?php
$dlugosc = 1234;
var_dump($dlugosc);
?>
```



Otrzymamy

```
int 1234
```

Oznacza on, że zmienna o nazwie długość przechowuje liczbę całkowitą(napis int) o wartości 1234

Wyrażeniem jest każda instrukcja, która zwraca wartość

Np. instrukcja przypisania

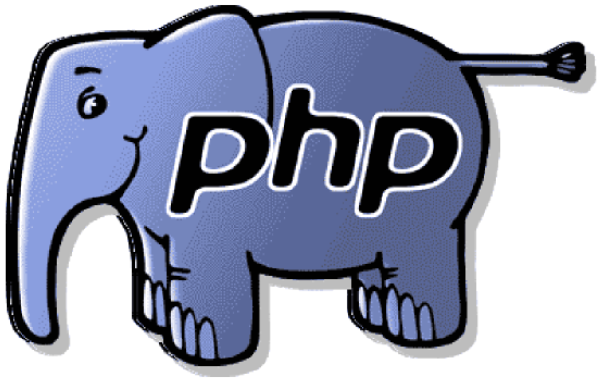
$a = 5

Zwraca wartość 5 W ten sposób instrukcja przypisania może wystąpić jako fragment większego wyrażenia

$b = ($a = 5) +7;

Powyższa instrukcja spowoduje najpierw przypisanie wartości 5 do zmiennej $a. wartość 5 zwrócona przez pierwsze przypisanie zostanie powiększona o 7 i wstawiona do zmiennej b. ostatecznie zmienna b przyjmuje wartość 12. Dokładnie taki sam efekt uzyskamy stosując 2 instrukcje

```
$a = 5;
$b = $a + 7;
echo ($b);
```

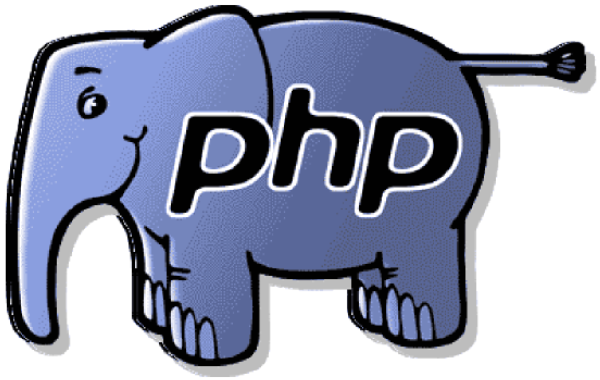


Proste operacje na zmiennych - operatory

OPERATORY:

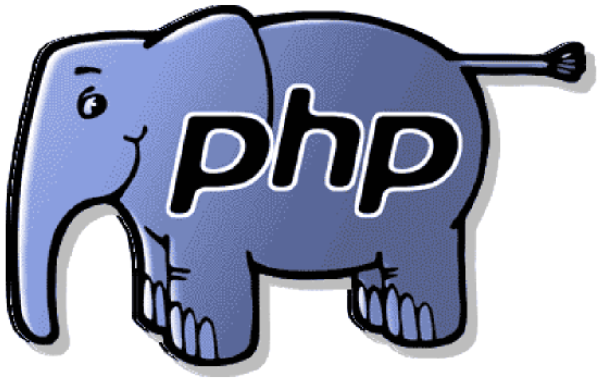
//operator przypisania – wykorzystywany do przypisania określonych wartości

```
$licznik = 0;
$Licznik = "5";
$ilosc = 7;
$informacja = "Stan licznika wynosi ";
$wlaczony = true;

$licznik = $ilosc;
```



// operatory arytmetyczne

```
echo $ilosc + 15;

$ilosc = $ilosc + 15;
$wynik1 = $ilosc - $licznik;
$wynik2 = $ilosc * $Licznik;
$wynik3 = 150 / $ilosc;
$ilosc = 2;
$reszta = 15 % $ilosc;

echo 'Reszta z dzielenia 15 / ' . $ilosc . ' wynosi '. $reszta;
```

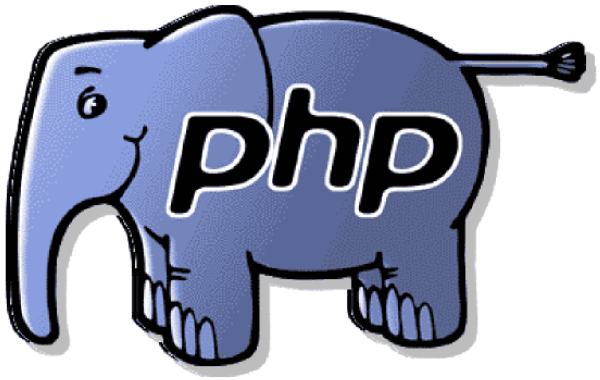
// inne operatory

// nawiasy

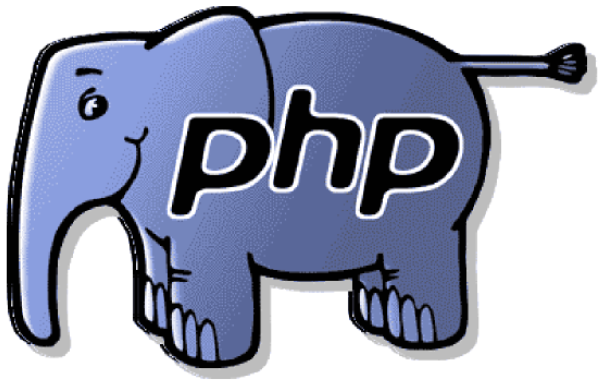
```
$wynik = ($wynik2 + 10) * 100;
```

// wytłumianie błędów / ostrzeżeń

```
@ $wynik = $ilosc / 0;
```



```
// inkrementacja i dekrementacja

$a = 1;

//++$a; //preinkrementacja
//$a++; //postinkrementacja

echo $a++; echo "<br>";
echo $a;
```



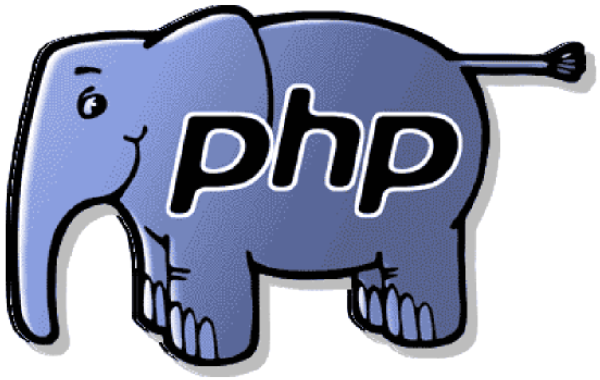
Instrukcje warunkowe if else

Instrukcje warunkowe są bardzo istotne gdyż dzięki nim możemy sprawdzając stan różnych zmiennych zdecydować czy wykonujemy takie czy inne operacje.

Instrukcja if nazwana jest także instrukcją selekcji, służy do warunkowego wykonania skryptu w zależności od wartości zmiennej

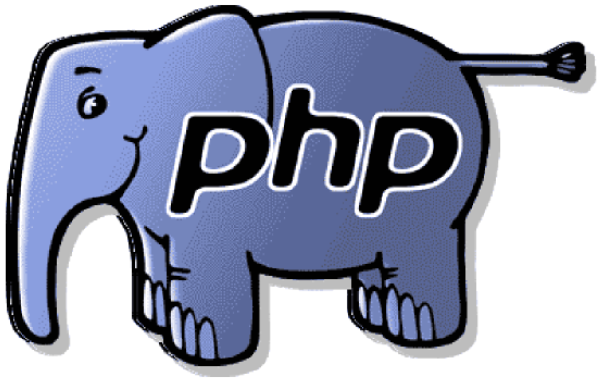
Przykład użycia

Najprostszym przykładem użycia instrukcji jest sprawdzenie czy użytkownik podał prawidłowe hasło, jeżeli tak to zezwalamy mu na pewne czynności jeżeli nie to każemy mu wpisać hasło raz jeszcze



Budowa instrukcji if else

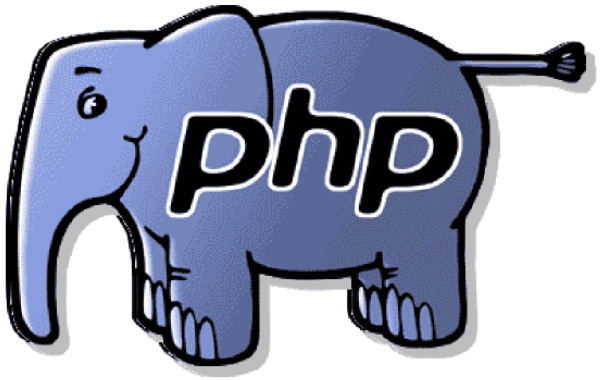
```
$ilosc = 15;

if  warunek) {

Jeżeli prawda wykona się blok kodu
            instrukcja 1
            $sprawdzenie = true;
            // kolejne instrukcje ...
}

Instrukcja else oznacza w przeciwnym wypadku
else {
            instrukcja2
            $sprawdzenie = false;
            // kolejne instrukcje ...
}
```



Operatory porównania:

Operator Zwraca TRUE, jeżeli:

==	Równość (takie same wartości)
!=	Nierówność
<>	Nierówność, znaczy to samo co !=
<	Mniejszy
>	Większy
<=	Mniejszy lub równy
>=	Większy lub równy



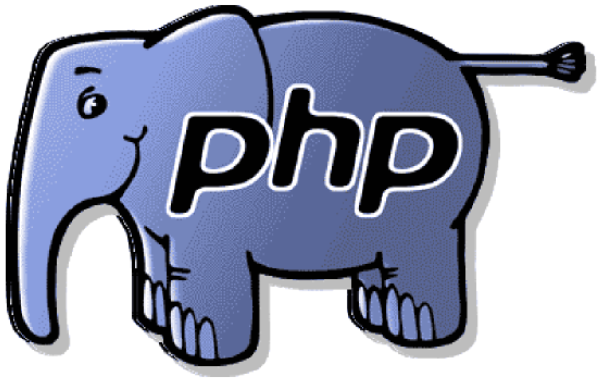
Pętla for

Pętla for w języku PHP przyjmuje postać taką jak w języku C++

for (zmienna l jako licznik -wartość początkowa; warunek dla licznika , zwiększenie licznika) {

Instrukcje

}

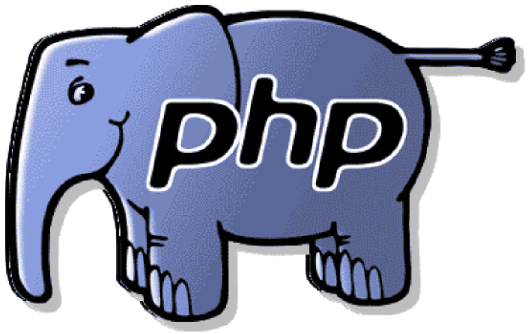


Przebieg pętli

- Zmienna działająca jako licznik to zmienna początkowa
- Warunek dla licznika to określenie ile razy ta pętla się nam ma wykonać jeśli np. 16 to $i > \<16
- Najpierw odczytywany jest licznik
- Następnie sprawdzany jest warunek jeśli przyjmuje on wartość false to wykonanie całej instrukcji zostanie zakończone
- Jeśli warunek będzie prawdziwy to następuje wykonanie instrukcji a po niej - zwiększenie licznika
- Po wykonaniu instrukcji a oraz zwiększenie licznika sprawdzany jest znów warunek z
- Jeśli warunek zakończenia przyjmuje wartość true to następuje ponowne wykonanie instrukcji i inst. oraz i zwiększenie licznika a po nim kolejny test warunku
- Jeśli warunek zakończenia przyjmuje wartość

  false to wykonanie całej instrukcji for

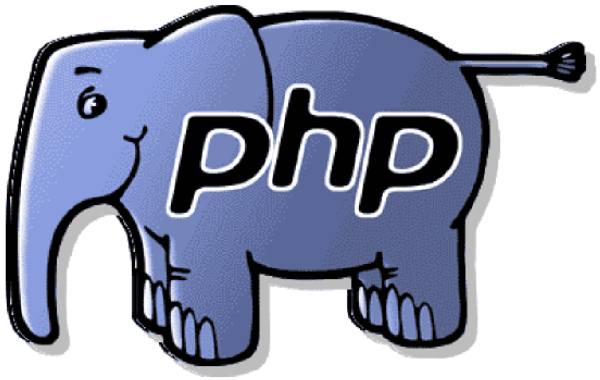
  zostanie zakończenia



Przykład

```
for ($i=1; $a<=16; i++) {

echo ("$i");
}
```

Co robi ten skrypt?



Operatory Inkrementacji i Dekrementacji czyli zwiększania bądź zmniejszania wartości

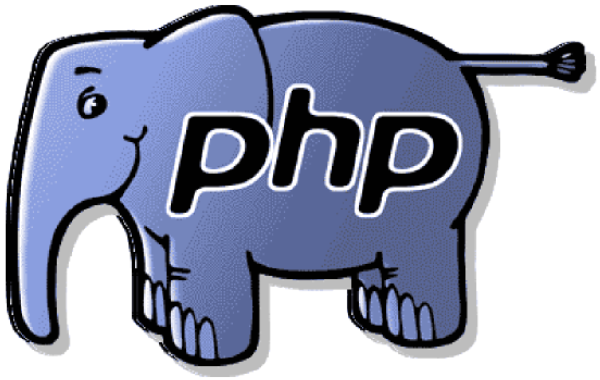
$a++ **Inkrementacja** zwiększanie wartości

$a = 1; //mamy tutaj zdefiniowana zmienna ma ona wartość 1

++$a; /* następnie za pomocą takiego operatora możemy ją zwiększyć , w tym przypadku jest

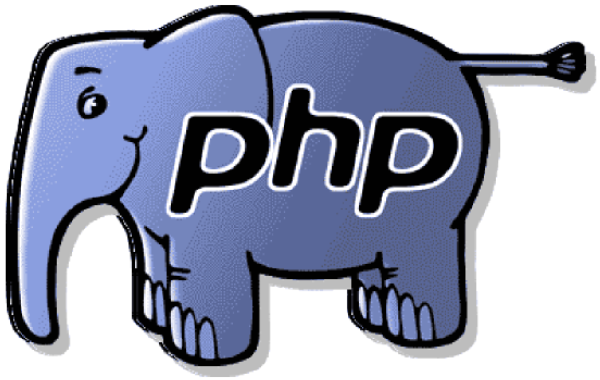
to tzw **preinkremetacja** oznacza to ze zmienna najpierw jest zwiększaną a potem zwracana

echo ($a);



Nieco inaczej działa **postinkrementacja** i nieco inaczej ją zapisujemy $a++

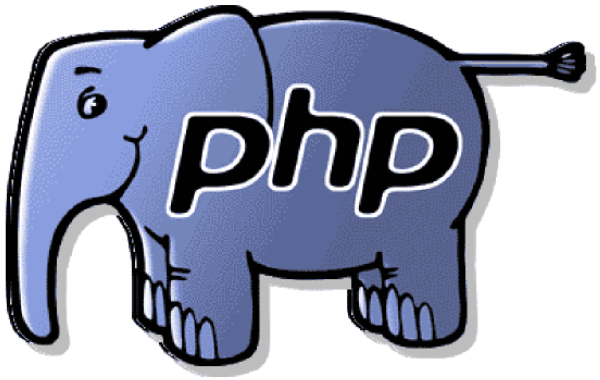
- Działa ona tak że najpierw jest zwracana zmienna a potem jest zwiększana o 1
- W przypadku **dekrementacji** mamy doczynienia z zmniejszaniem wartości



SWITCH CASE *instrukcja wyboru*

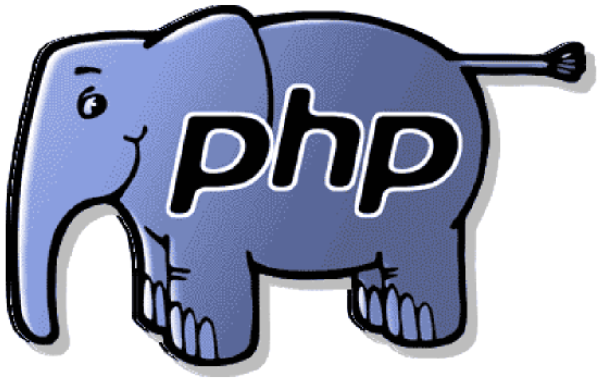
- Instrukcja ta pozwala na sterowanie programem, ale działa nieco inaczej od instrukcji if.

- Pozwala na wybranie jednego z przypadków



Konstrukcja

```
switch(zmienna, która jest sprawdzana) {
case1:
instr1;
break; - pozwala na przerwanie kodu
…

default:
case2:
nstr2;
break;
}
```



Jak to działa ? „opis łopatologiczny”

- Po słowie kluczowym switch wpisujemy nazwę zmiennej która jest sprawdzana

```
$a=2;
switch ($a) {
```

Następnie definiujemy Blok kodu musimy zdefiniować instrukcje aby zdefiniować każdy taki przypadek musimy użyć słowa

```
case ”2”;
    echo ”to jest liczba 2”;
```

Tutaj określamy co ma się wykonać jeśli spełniony będzie warunek czyli dla liczby 2 ma wypisać

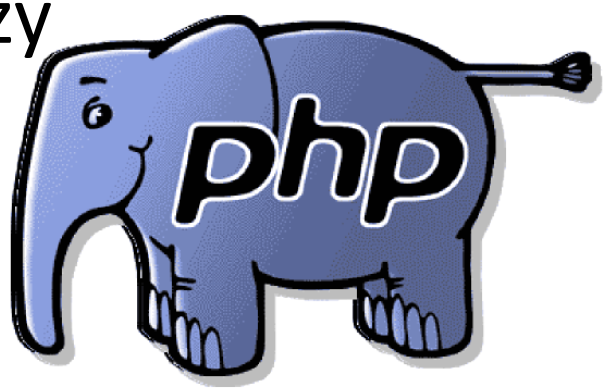
break; - używamy aby zakończyć instrukcję

WHILE

Instrukcja while w języku php ma postać

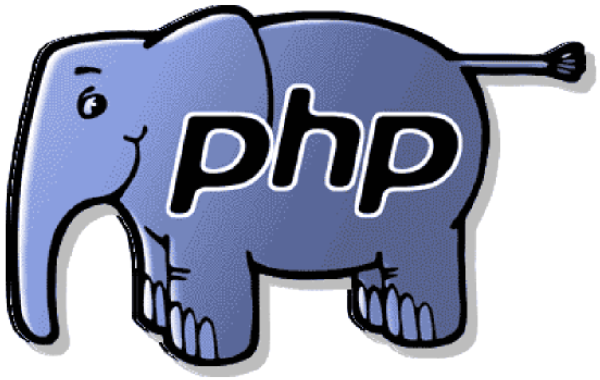
```
while (warunek) {
instrukcja;
}
```

- Służy do wielokrotnego wykonywania instrukcji w zależności od tego czy podany warunek jest spełniony



Wykonanie instrukcji przebiega następująco:

- Najpierw sprawdzany jest warunek
- Jeśli warunek spełnia wartość logiczną true to wykonywana jest instrukcja po jej wykonaniu ponownie sprawdzany jest warunek
- Jeśli warunek przyjmie wartość false to instrukcja while jest zakończona



DO WHILE

- Pętla while sprawdza warunek przed wykonaniem instrukcji pętla do while po wykonaniu

```
do {
Instrukcja
}
while (warunek);
```

- nie zależnie czy warunek jest spełniony to i tak wykonana się chociaż raz