<https://www.perltutorial.org/perl-regular-expression-character-classes/>

<https://perlmaven.com/manipulating-perl-arrays>

Table of Contents

[Print 1](#_Toc76236767)

[EOF – do przechowywania długich stringów 2](#_Toc76236768)

[Wielka litera: 3](#_Toc76236769)

[Wielka litera dla całego tekstu: 3](#_Toc76236770)

[Wielka litera dla fragmentu tekstu – od \U do \E: 4](#_Toc76236771)

[Znak \ przed każdym znakiem niealfanumerycznym (czyli np. spacją): 4](#_Toc76236772)

[Mała litera dla całego tekstu: 5](#_Toc76236773)

[Typy danych – skalar. Może być liczbą albo tekstem: 5](#_Toc76236774)

[Sprawdzanie długości tekstu: 6](#_Toc76236775)

[Wyszukiwanie tekstu w tekście: 7](#_Toc76236776)

[Zamiana fragmentu tekstu: 7](#_Toc76236777)

[Konkatenacja: 8](#_Toc76236778)

[Typ danych – array czyli macierz (chyba): 8](#_Toc76236779)

[W taki sposób odwołujemy się do poszczególnych elementów macierzy: 9](#_Toc76236780)

[Taka lista która ma w sobie zarówno stringi jak i numery nazywa się complex list 9](#_Toc76236781)

[Wstawianie wartości do zmiennych w macierzy: 9](#_Toc76236782)

[Sekwencyjne tworzenie macierzy: 10](#_Toc76236783)

[Sprawdzanie rozmiaru macierzy: 10](#_Toc76236784)

[Dodawanie elementów na końcu macierzy: 11](#_Toc76236785)

[Dodawanie elementu na początku macierzy: 11](#_Toc76236786)

[Usuwanie ostatniego elementu listy: 11](#_Toc76236787)

[Usuwanie pierwszego elementu listy: 12](#_Toc76236788)

[Sortowanie elementów w macierzy: 12](#_Toc76236789)

[Zastępowanie wybranych elementów macierzy: 12](#_Toc76236790)

[Zmiana stringa w macierz: 12](#_Toc76236791)

[Zmiana macierzy w string: 13](#_Toc76236792)

[Macierze są numerowane od 0 ale można zmienić to na inną wartość: 13](#_Toc76236793)

[Łączenie macierzy: 13](#_Toc76236794)

[Typ danych klucz-wartość: 13](#_Toc76236795)

[Deklaracja i odwołanie do wartości: 13](#_Toc76236796)

[Zmiana wartości: 14](#_Toc76236797)

[Odwołanie do kv jak do macierzy wywyołując klucz lub wartość: 14](#_Toc76236798)

[Sprawdzanie czy podany klucz istnieje (exists) i funkcja if: 15](#_Toc76236799)

[Sprawdzane rozmiaru kv: 15](#_Toc76236800)

[Dodawanie kolejnej wartości do kv: 16](#_Toc76236801)

[Usuwanie wartości z kv: 16](#_Toc76236802)

[Przelatywanie po wartościach w kv: 16](#_Toc76236803)

[If than else: 17](#_Toc76236804)

[Jest jeszcze unless, który działa podobnie: 17](#_Toc76236805)

[Switch: 18](#_Toc76236806)

[Alternatywą do switch jest given: 19](#_Toc76236807)

[Pętla while: 20](#_Toc76236808)

[Do while: 20](#_Toc76236809)

[Until: 21](#_Toc76236810)

[Do until: 22](#_Toc76236811)

[Next czyli omijanie linijek kodu w przypadku dgy spełniony jest określony warunek: 23](#_Toc76236812)

[Pętla for: 23](#_Toc76236813)

[Wykorzystujemy tutaj przypisanie do specjalnej zmiennej $\_ ale możemy też sobie stworzyć sztuczny iterator: 24](#_Toc76236814)

[Pętla foreach: 24](#_Toc76236815)

[Data: 24](#_Toc76236816)

[Czas unixowy: 25](#_Toc76236817)

[Maski daty: 26](#_Toc76236818)

[Subroutines 28](#_Toc76236819)

[Coś jak funkcja: 28](#_Toc76236820)

[Funkcja z parametrem: 29](#_Toc76236821)

[Funkcja z parametrem, zliczanie liczb podanych w argumencie: 29](#_Toc76236822)

[Jako argument można podawać listę: 32](#_Toc76236823)

[Funkcja może zwracać arraya: 34](#_Toc76236824)

[Zmienne prywatne i globalne: 35](#_Toc76236825)

[Odczyt z pliku txt 35](#_Toc76236826)

[Wprowadzanie wartości z terminala do zmiennych: 36](#_Toc76236827)

[Funkcja chomp(): 37](#_Toc76236828)

[Regular expressions: 37](#_Toc76236829)

[Szukamy w którym stringu są 4 cyfry pod rząd: 41](#_Toc76236830)

[Wyszukaj i zamień tekst (wszystkie wystąpienia): 43](#_Toc76236831)

[Zlicz zmiany, które zostały wykonane: 45](#_Toc76236832)

[Zamiana znak po znaku: 45](#_Toc76236833)

[Wskaźniki: 47](#_Toc76236834)

[Moduł 48](#_Toc76236835)

[Klasy 49](#_Toc76236836)

[Baza danych 53](#_Toc76236837)

[Insert 56](#_Toc76236838)

[Select 59](#_Toc76236839)

[Select z wpisywanym warunkiem na target: 60](#_Toc76236840)

[UPDATE 61](#_Toc76236841)

[DELETE 62](#_Toc76236842)

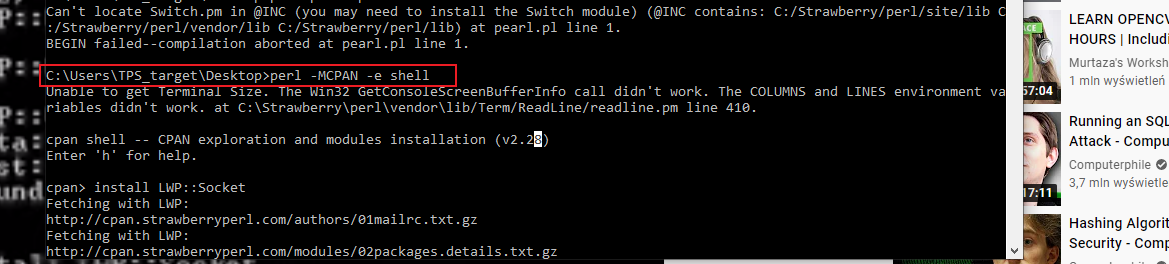
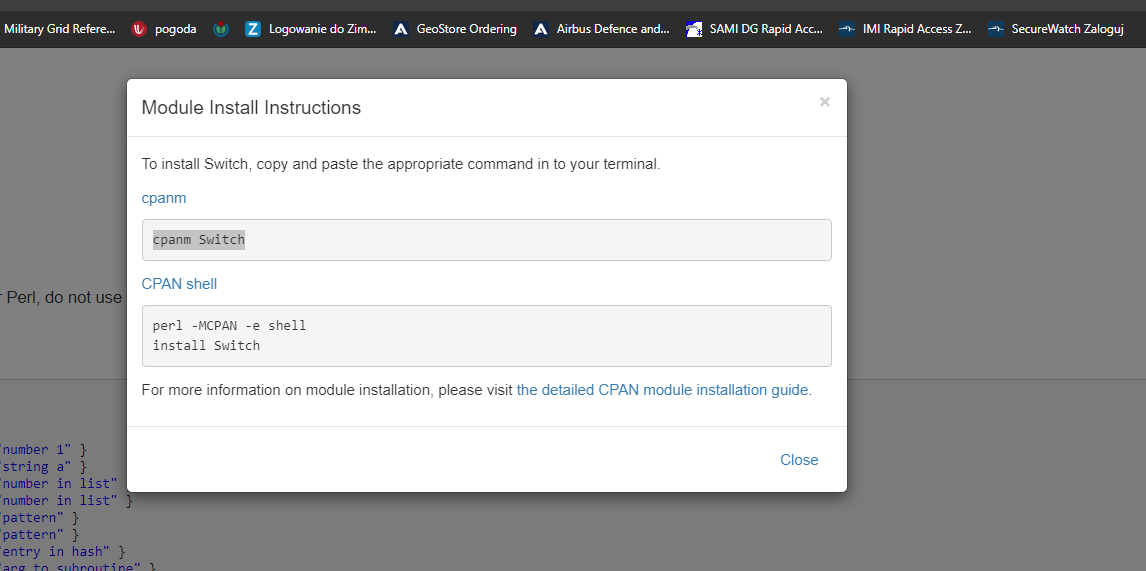
[Sortowanie: 64](#_Toc76236843)

[Sortowanie znaków: 64](#_Toc76236844)

[Sortowanie cyfr: 64](#_Toc76236845)

[Printf – formatowanie: 66](#_Toc76236846)

[Sortowanie w arrayu: 70](#_Toc76236847)



https://www.tutorialspoint.com/perl/perl\_syntax.htm

*print ”Hello world\n”;*

n jest po to żeby było przejście do kolejnej linijki

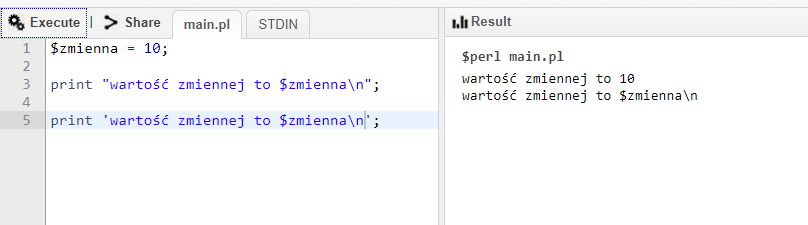
średnik jest na końcu każdego polecenia

PERL – practical extraction and report language

Larry Wall

# Print

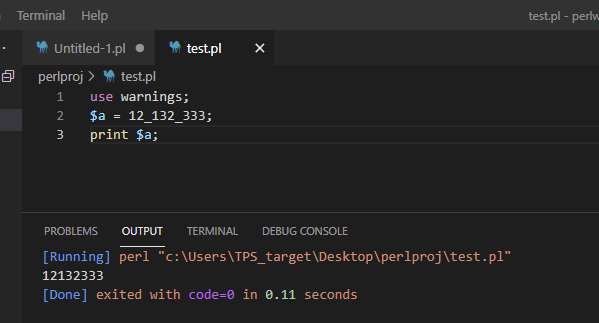
Jeśli w print użyjemy pojedynczego nawiasu zamiast podwójnego to zmienna będzie traktowana jako zwykły tekst:



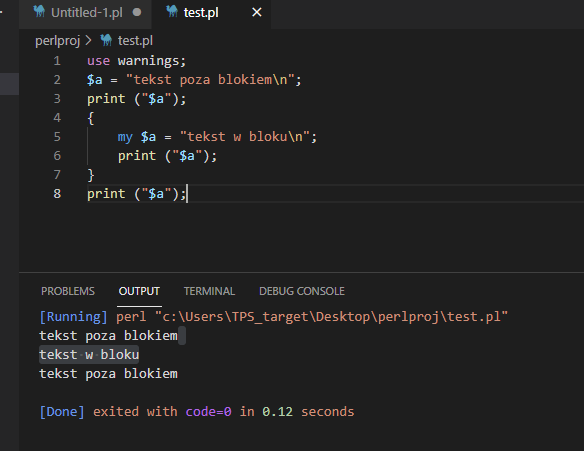
Używanie zmiennej w tekście (tak jak w pierwszym przykładzie) nazywa się interpolacją.

Interpolację można robić dla zmiennych, macierzy (arrays) ale nie dla klucz-warość(hashes).

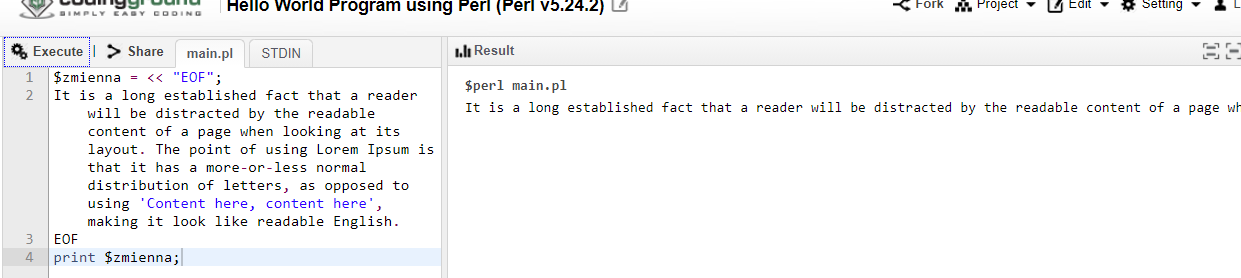
W celu zwiększenia czytelności zapisu duzych liczb możemy wstawiać znak \_ pomiędzy miejscami dziesiętnymi:



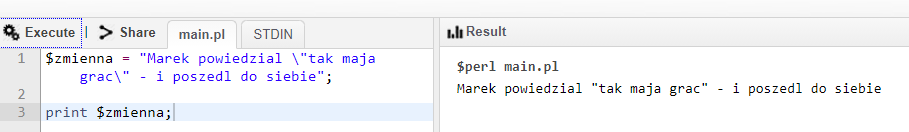
Zmienne mają swój zasięg – obejmują blok w którym się znajdują:



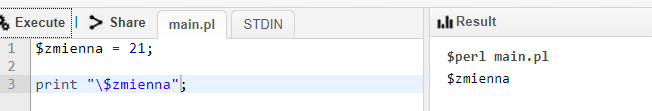
# EOF – do przechowywania długich stringów



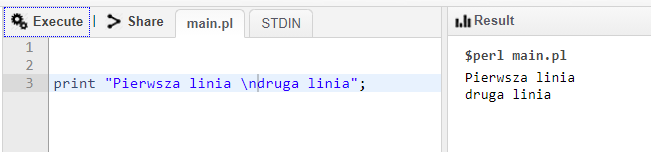
Żeby uwzględnić cudzysłów w stringu należy go poprzedzić znakiem \



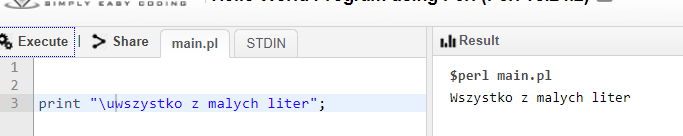
Jeśli użyjemy znaku \ przed $ to print potraktuje to jako tekst a nie jako zmienną:



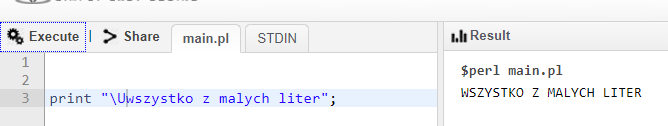
przejście do nowej linii:

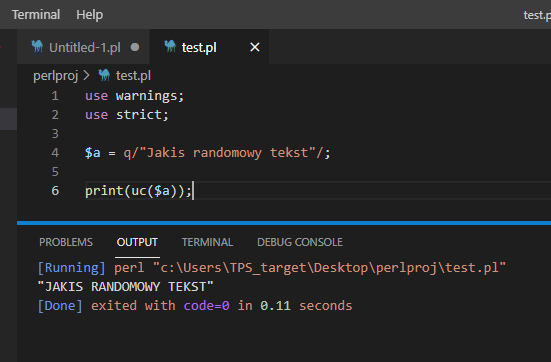


# Wielka litera:

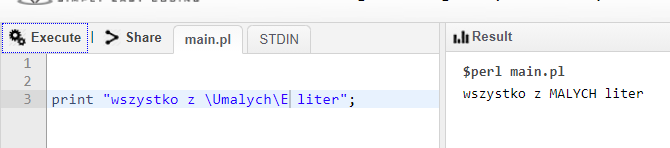


# Wielka litera dla całego tekstu:

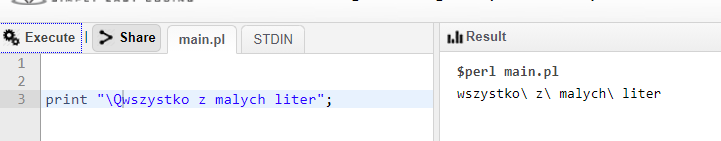




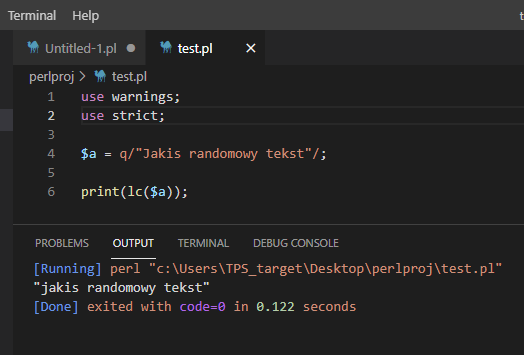
# Wielka litera dla fragmentu tekstu – od \U do \E:



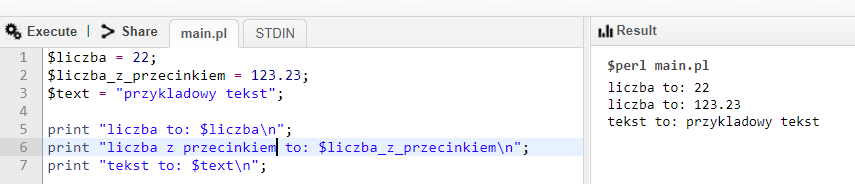
# Znak \ przed każdym znakiem niealfanumerycznym (czyli np. spacją):

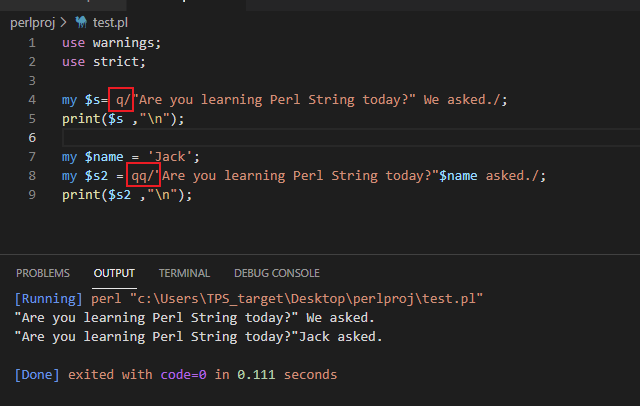


# Mała litera dla całego tekstu:

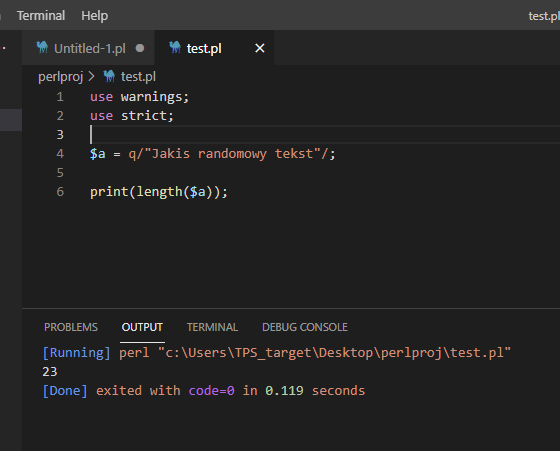


# Typy danych – skalar. Może być liczbą albo tekstem:

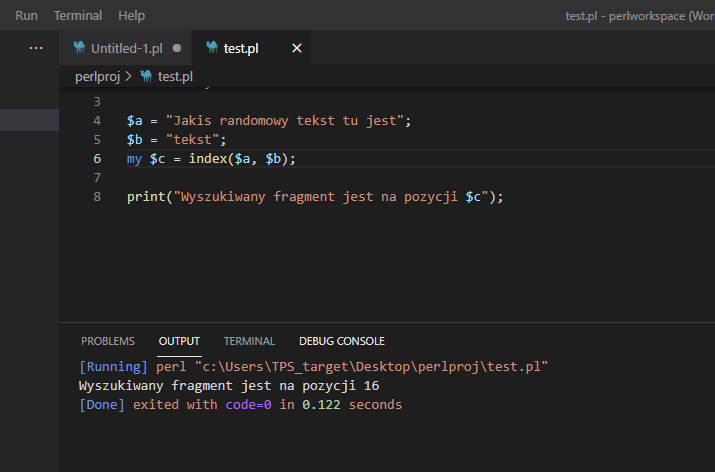


Tekst w zmiennej można zapisywać w inny sposób. W zależności czy będziemy robili interpolację czy nie możemy zapisać: 

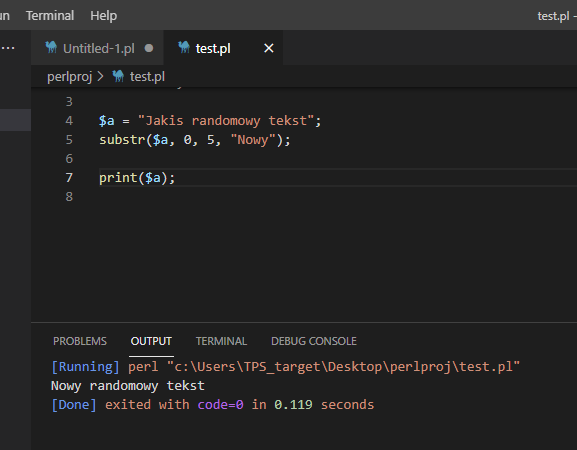
## Sprawdzanie długości tekstu:



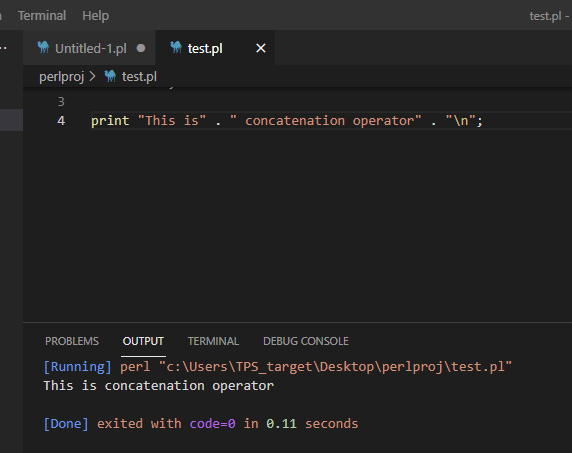
## Wyszukiwanie tekstu w tekście:



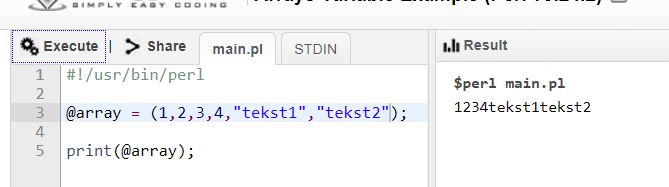
## Zamiana fragmentu tekstu:



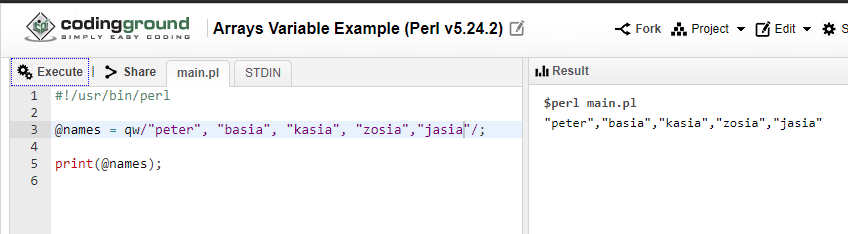
## Konkatenacja:



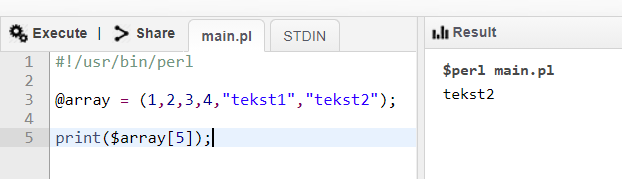
# Typ danych – array czyli macierz (chyba):



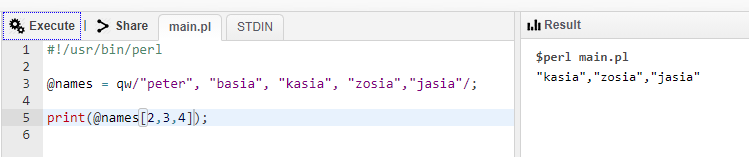
Inny sposób na deklarację macierzy:

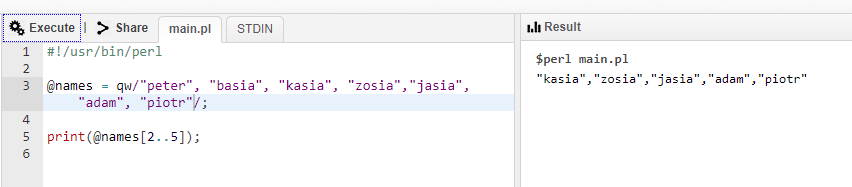


## W taki sposób odwołujemy się do poszczególnych elementów macierzy:

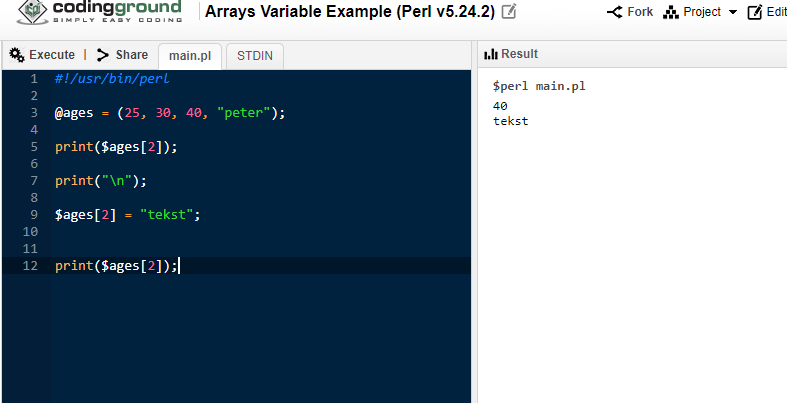


## Taka lista która ma w sobie zarówno stringi jak i numery nazywa się complex list

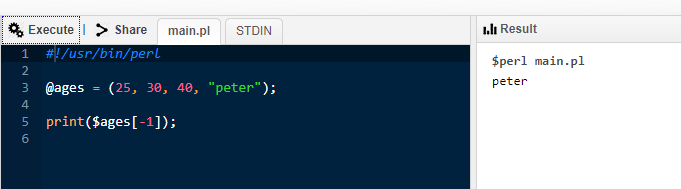




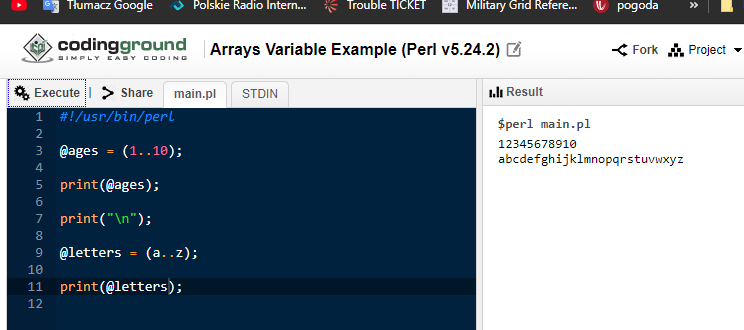
## Wstawianie wartości do zmiennych w macierzy:



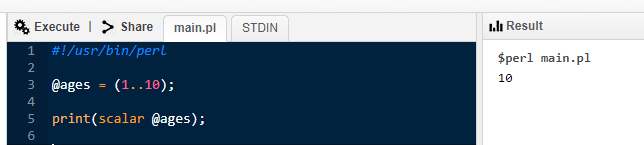
Można odwoływać się do elementów macierzy od tyłu:

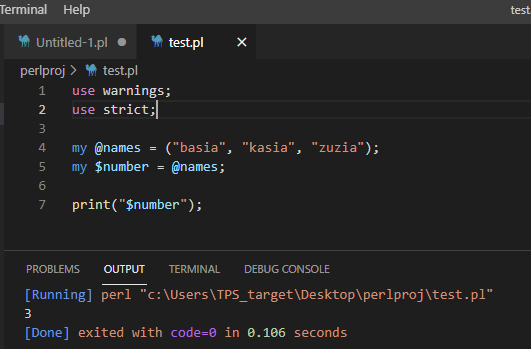


## Sekwencyjne tworzenie macierzy:

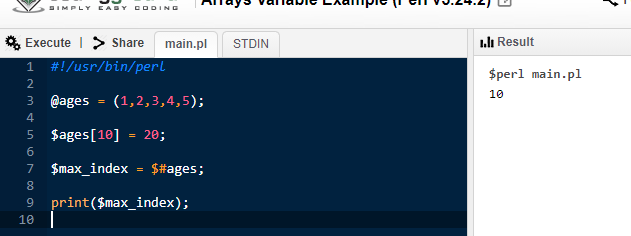


## Sprawdzanie rozmiaru macierzy:

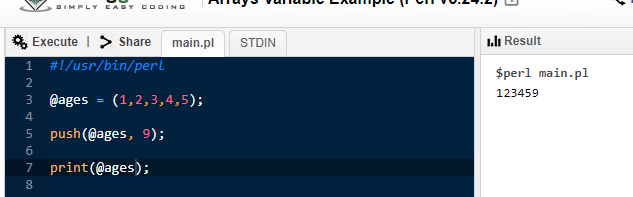




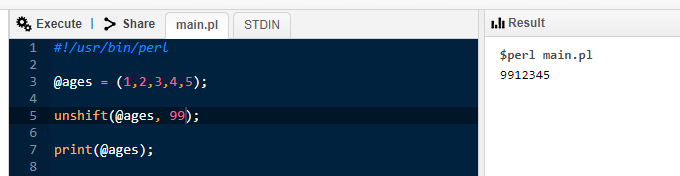
Macierz nie musi być „ciągła”. Możemy sprawdzić maksymalny indeks macierzy:



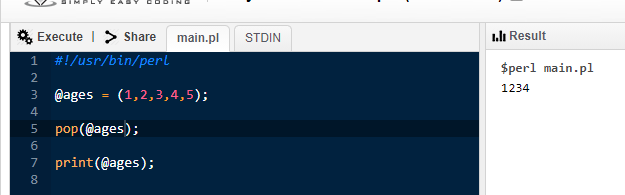
## Dodawanie elementów na końcu macierzy:



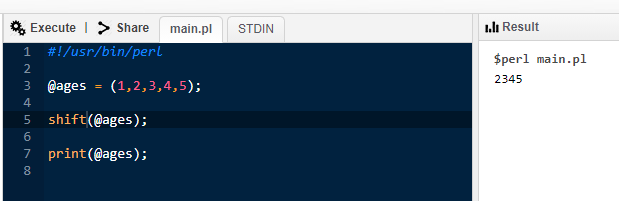
## Dodawanie elementu na początku macierzy:



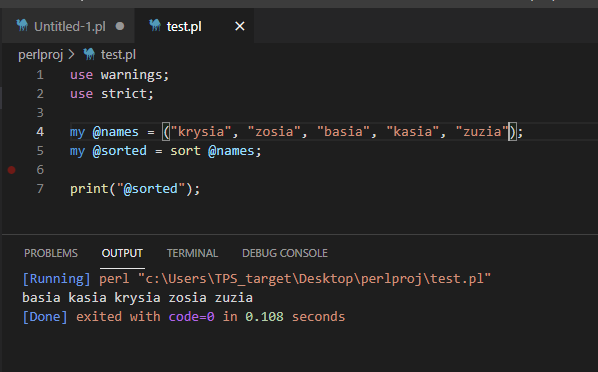
## Usuwanie ostatniego elementu listy:



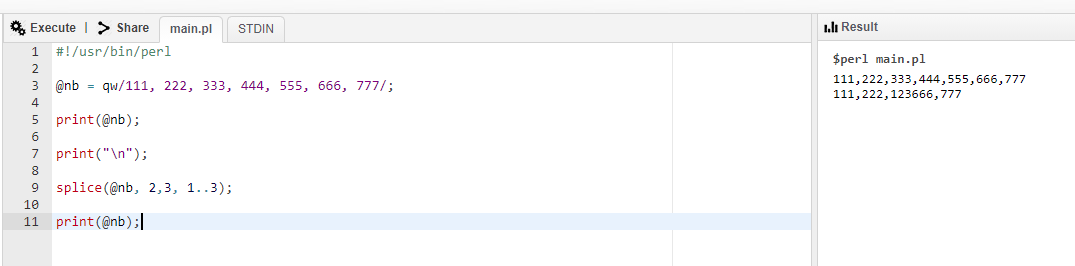
## Usuwanie pierwszego elementu listy:



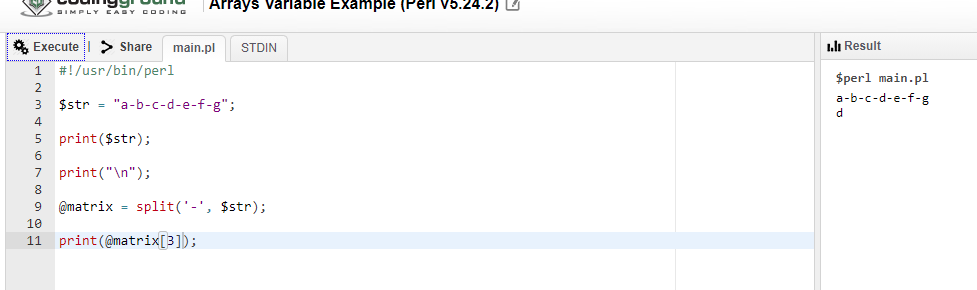
## Sortowanie elementów w macierzy:



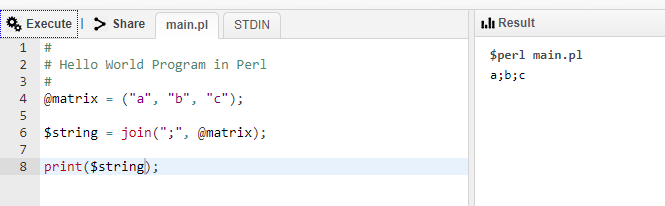
## Zastępowanie wybranych elementów macierzy:



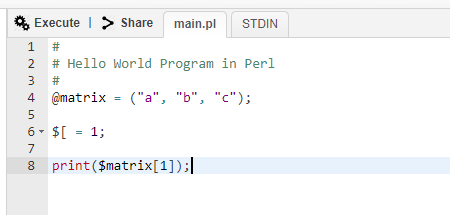
## Zmiana stringa w macierz:



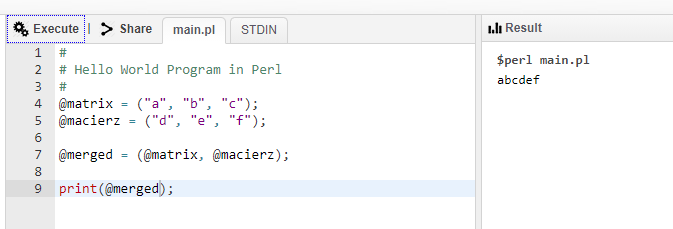
## Zmiana macierzy w string:



## Macierze są numerowane od 0 ale można zmienić to na inną wartość:

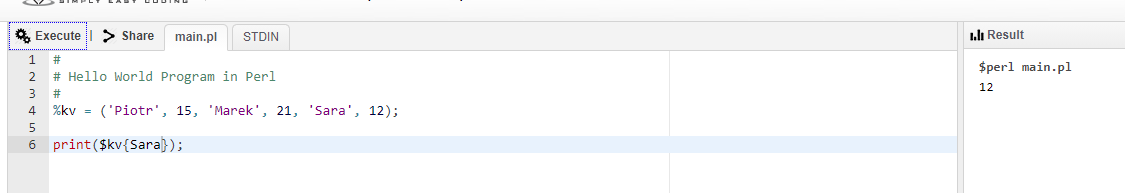


## Łączenie macierzy:

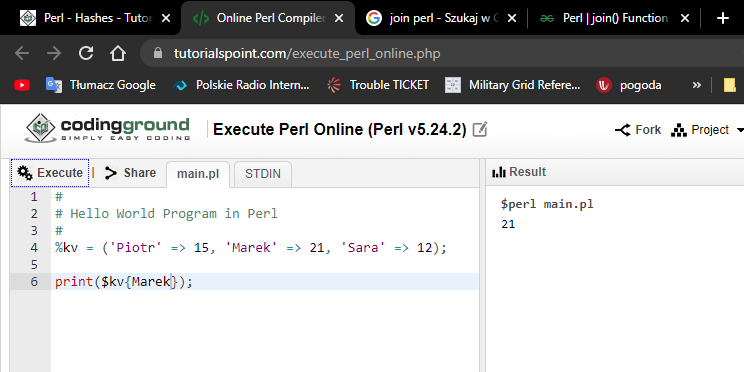


# Typ danych klucz-wartość:

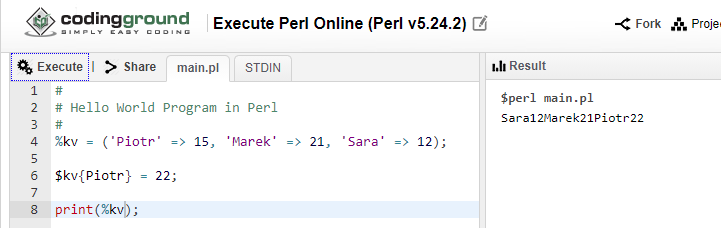
## Deklaracja i odwołanie do wartości:



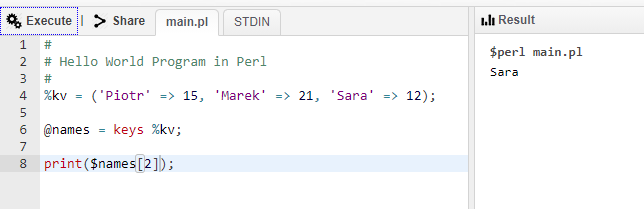
Można też deklarować w taki sposób żeby nie było wątpliwości co jest kluczem a co wartością:

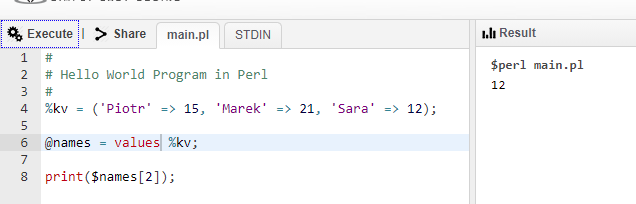


## Zmiana wartości:

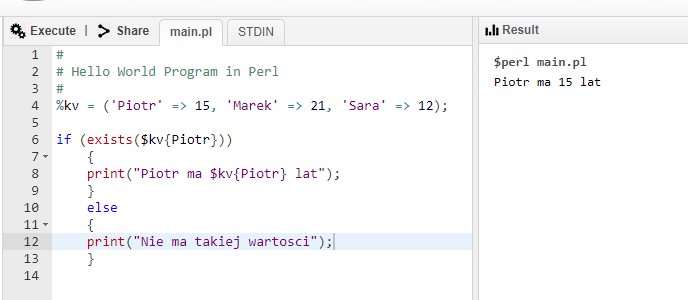


## Odwołanie do kv jak do macierzy wywyołując klucz lub wartość:

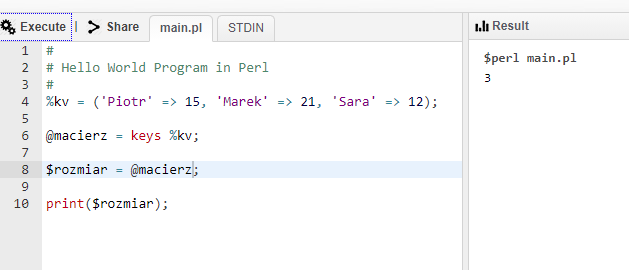




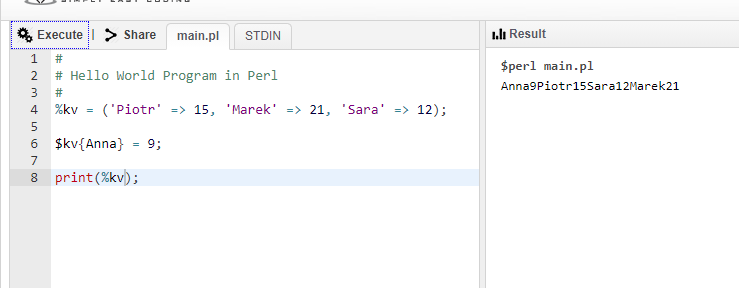
## Sprawdzanie czy podany klucz istnieje (exists) i funkcja if:



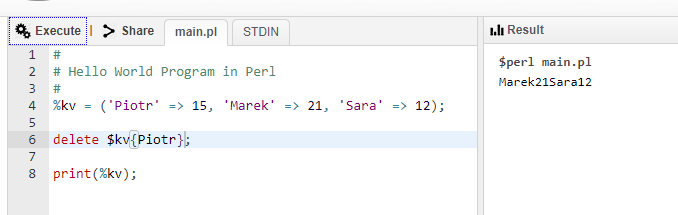
## Sprawdzane rozmiaru kv:



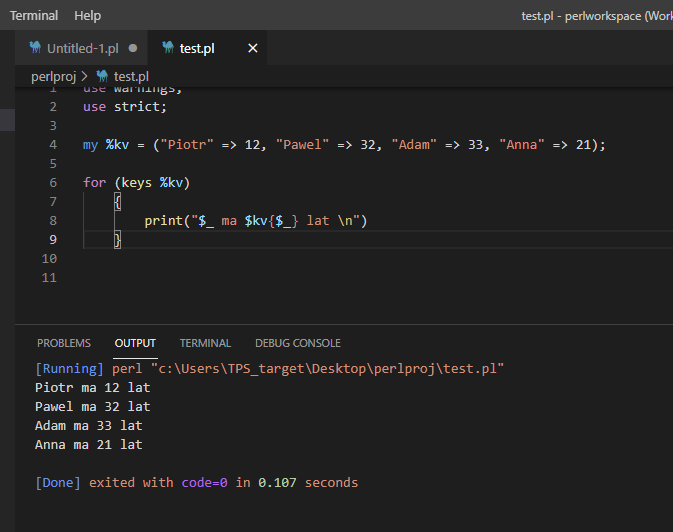
## Dodawanie kolejnej wartości do kv:



## Usuwanie wartości z kv:

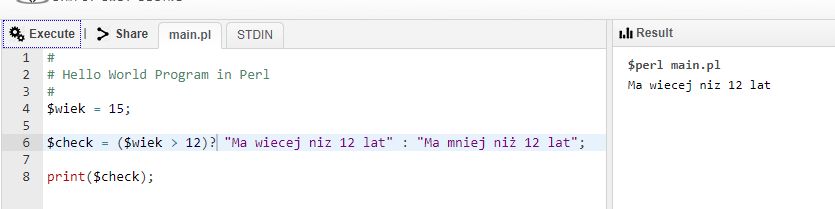


## Przelatywanie po wartościach w kv:

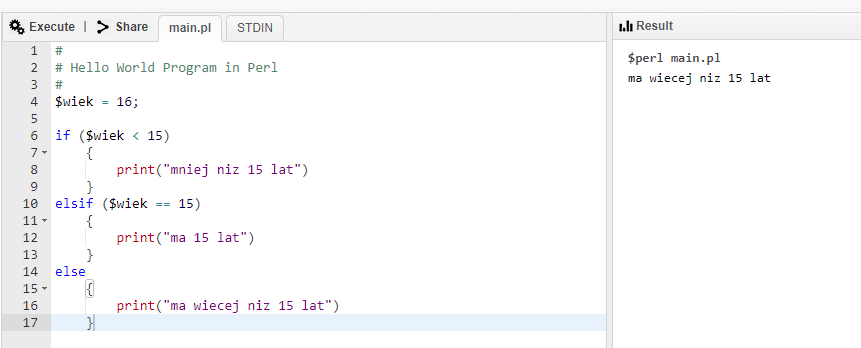


# If than else:

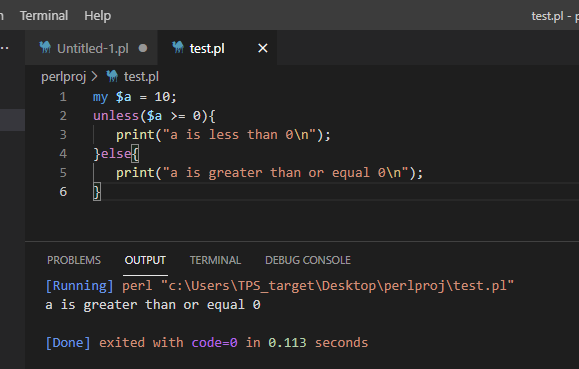
Najprostrza forma:



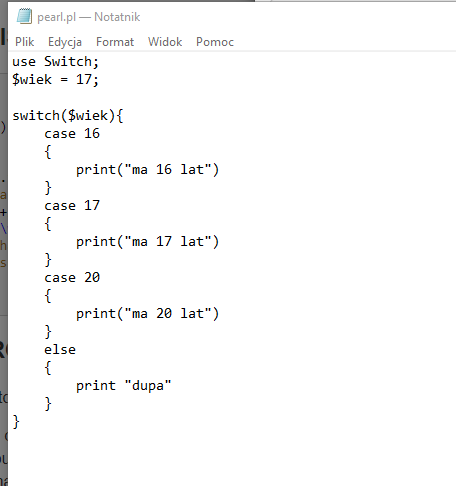
Zwykła forma:

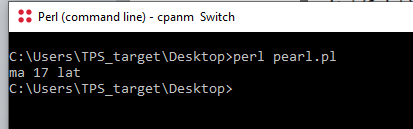


## Jest jeszcze unless, który działa podobnie:

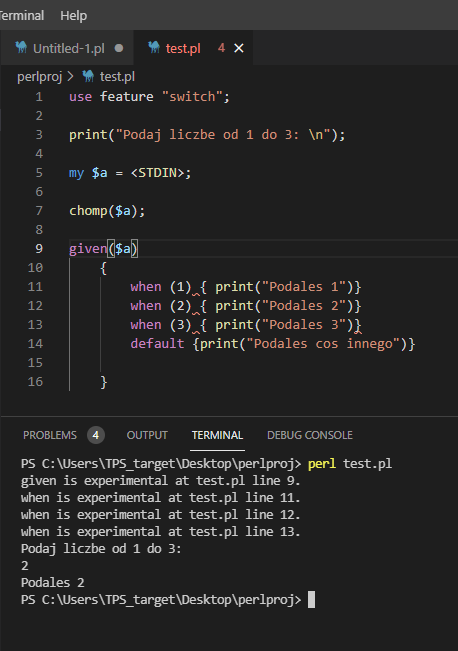


# Switch:

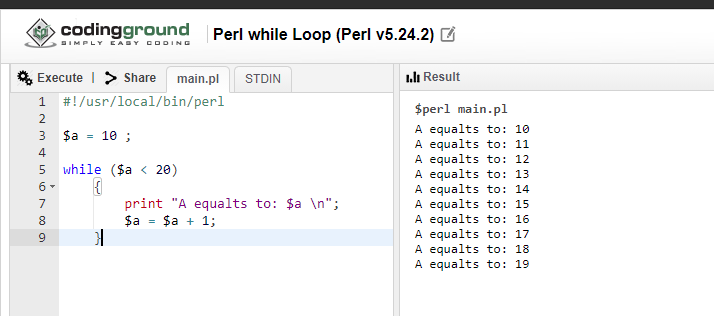




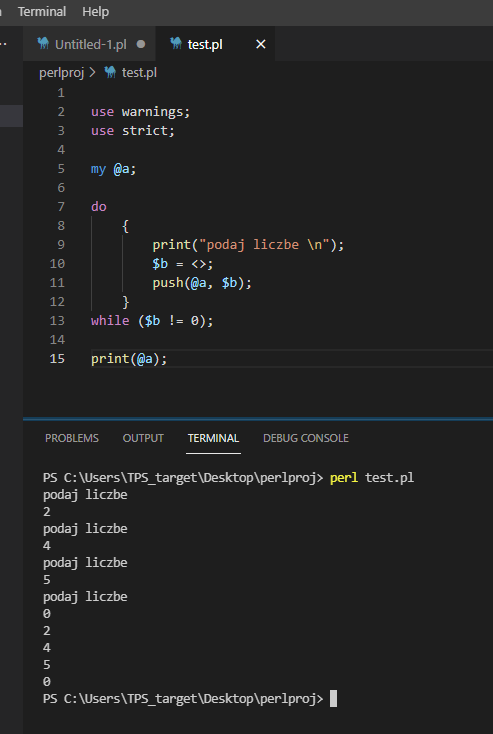
## Alternatywą do switch jest given:



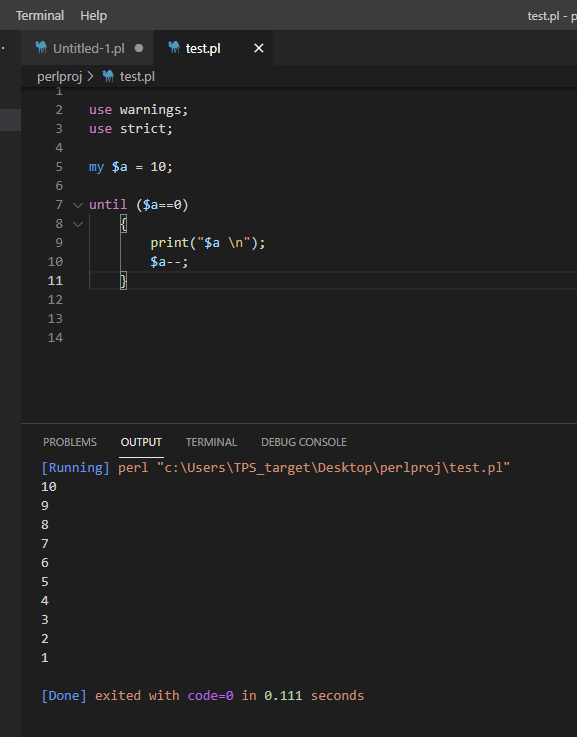
# Pętla while:



## Do while:



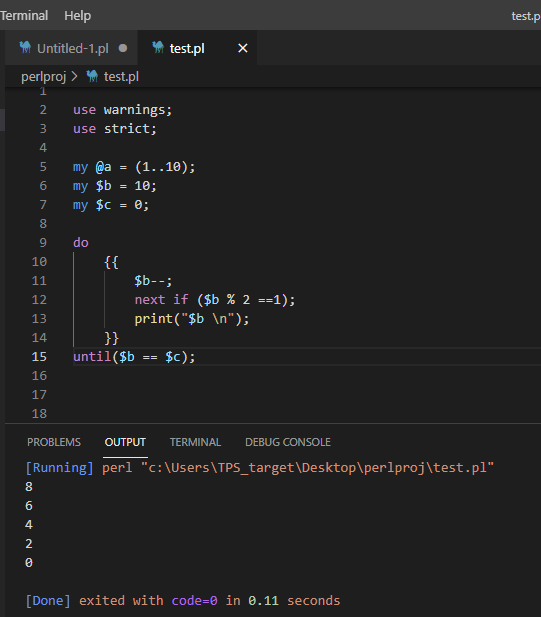
## Until:



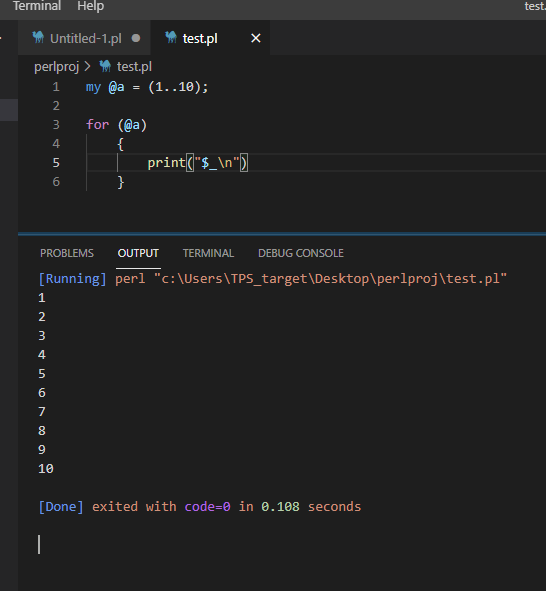
## Do until:



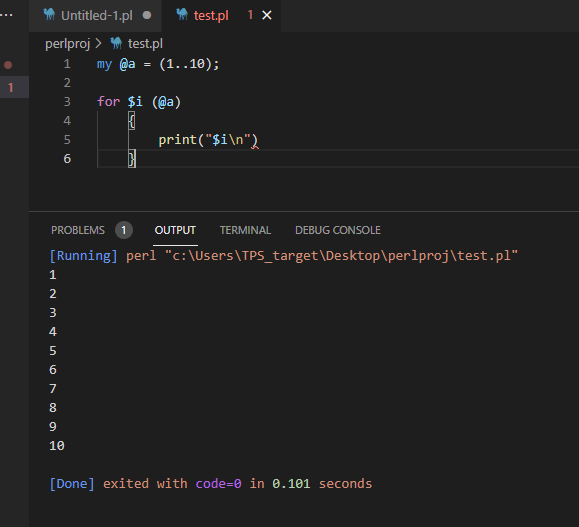
# Next czyli omijanie linijek kodu w przypadku dgy spełniony jest określony warunek:

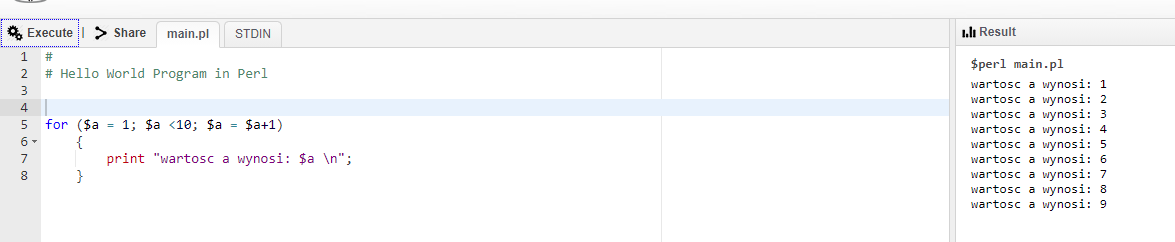


# Pętla for:

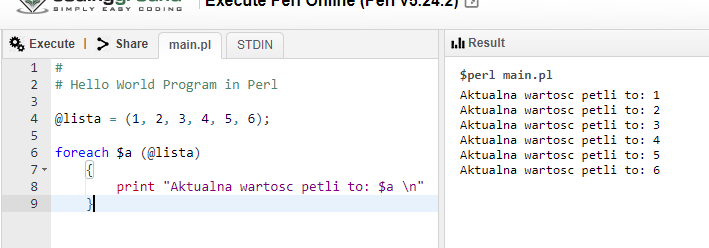


## Wykorzystujemy tutaj przypisanie do specjalnej zmiennej $\_ ale możemy też sobie stworzyć sztuczny iterator:

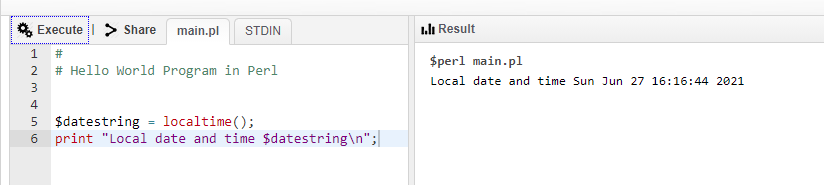


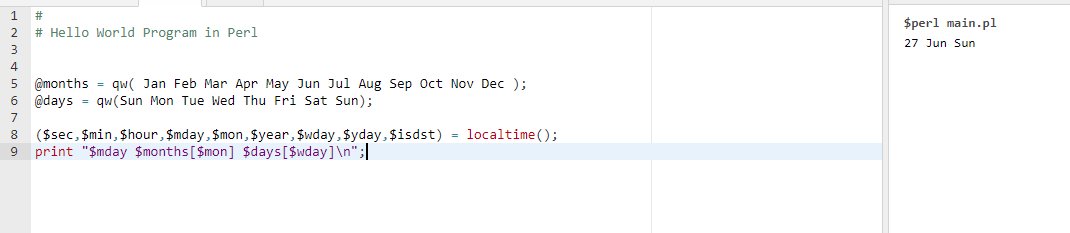


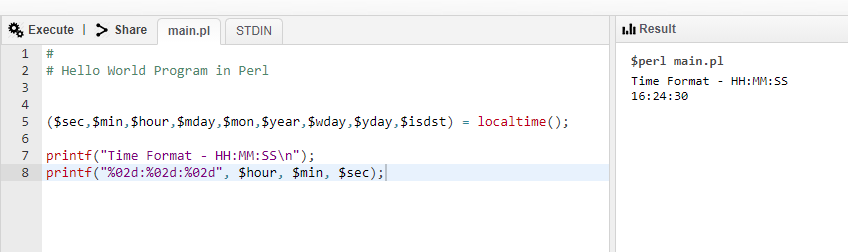
## Pętla foreach:



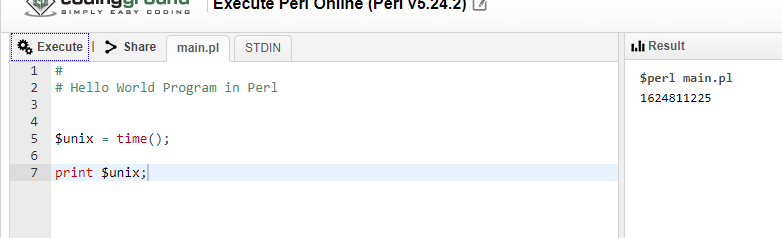
# Data:



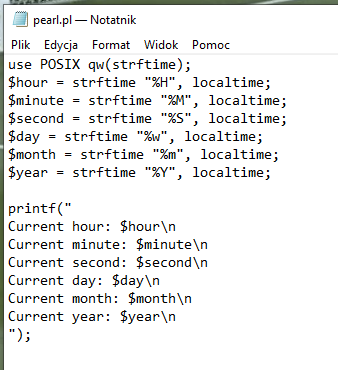


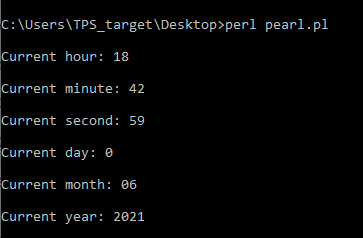


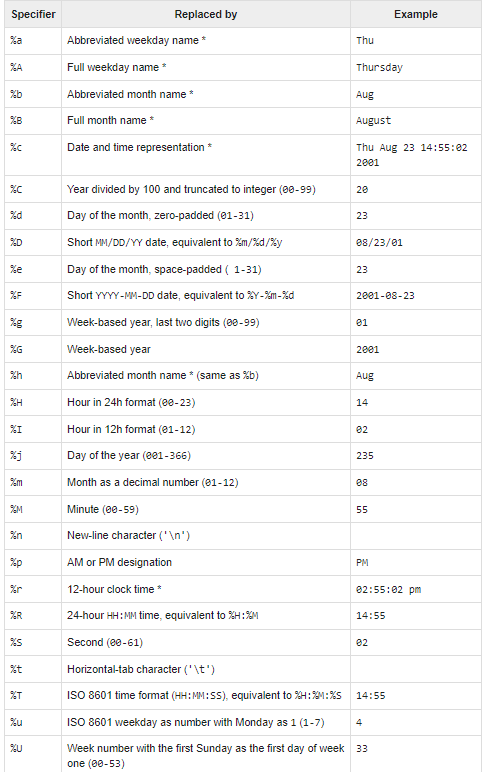
## Czas unixowy:

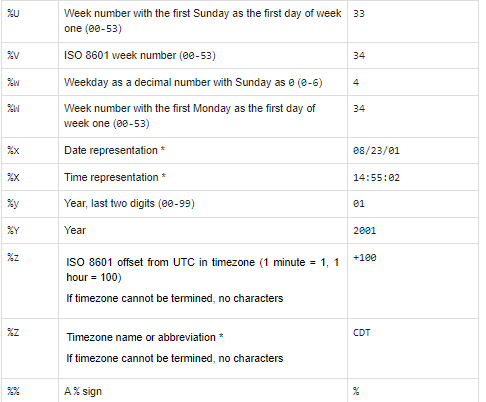


## Maski daty:



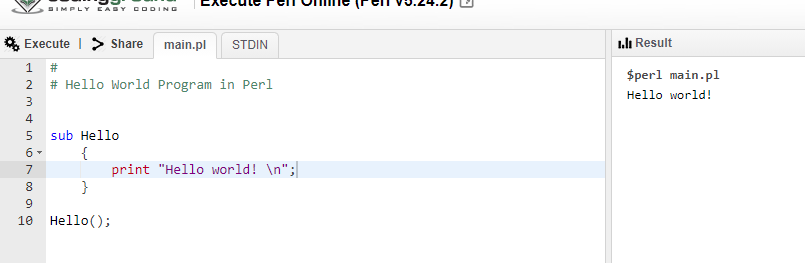




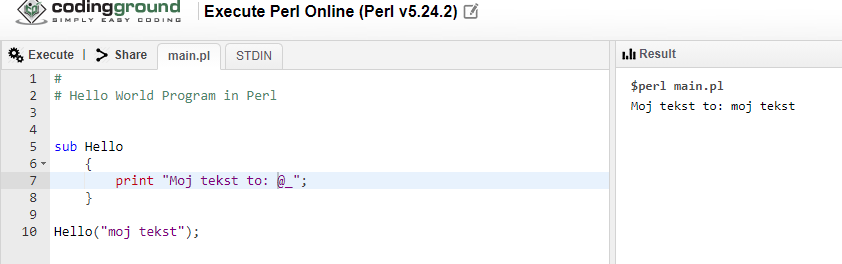


# Subroutines

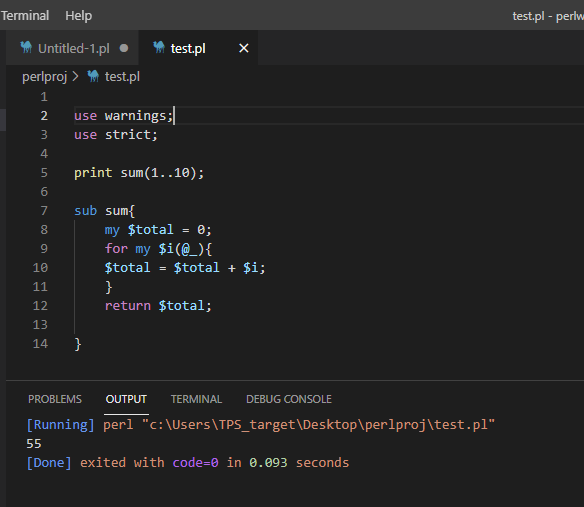
# Coś jak funkcja:



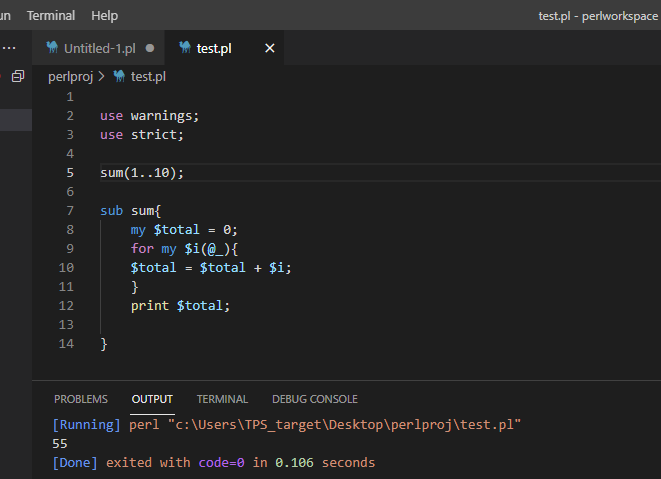
## Funkcja z parametrem:



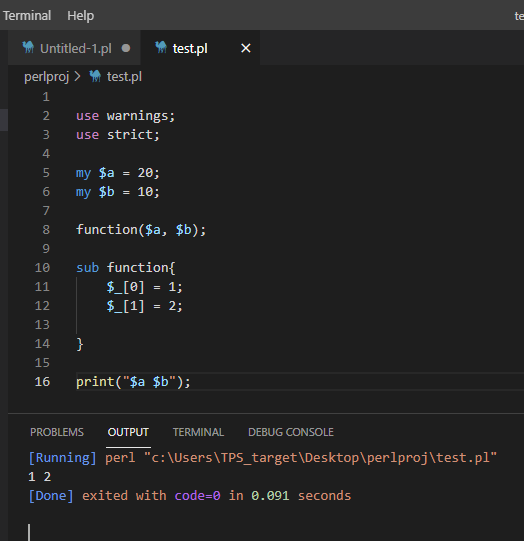
## Funkcja z parametrem, zliczanie liczb podanych w argumencie:



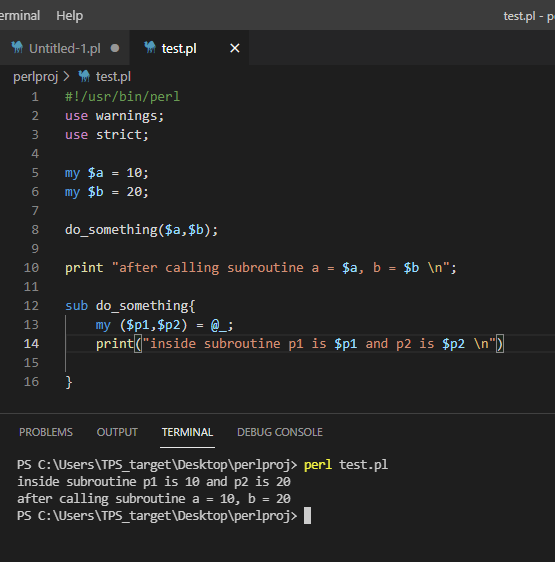
Alternatywnie:



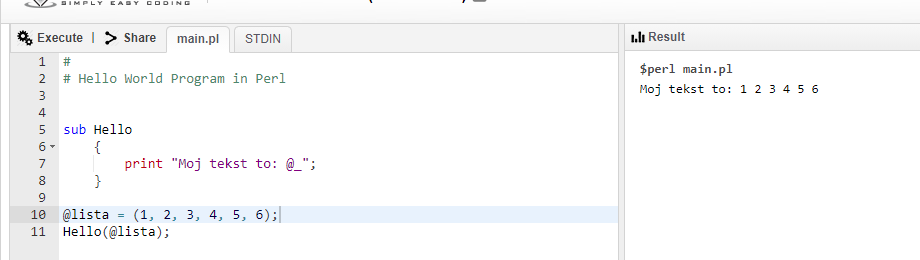
Wiele argumentów:



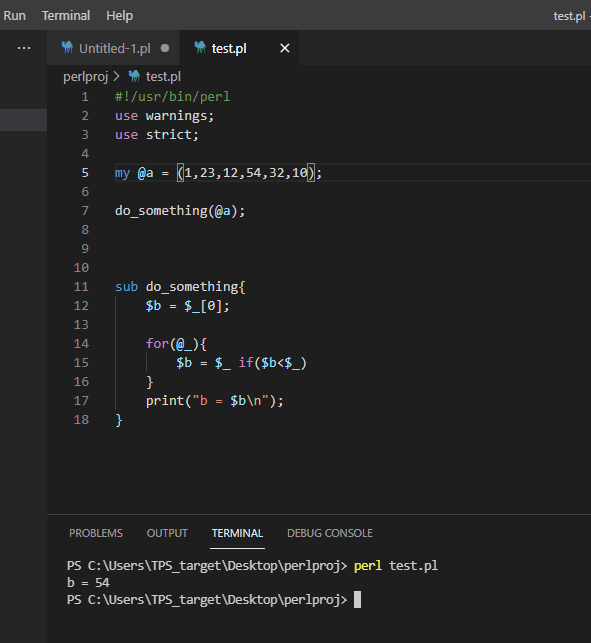
Podawanie wielu arguentów do funkcji:



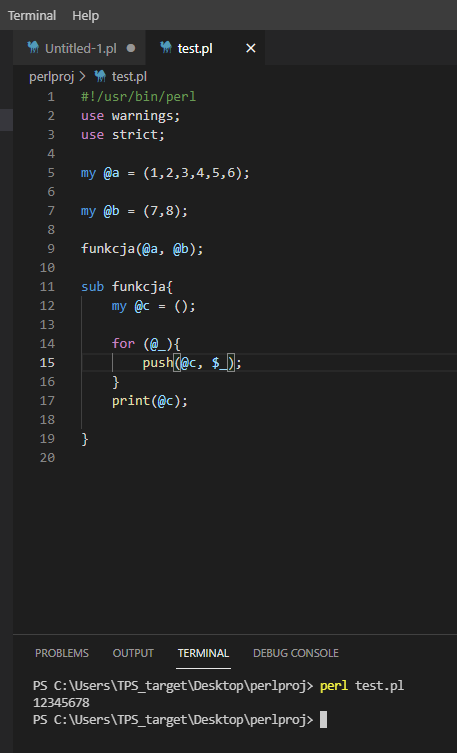
## Jako argument można podawać listę:



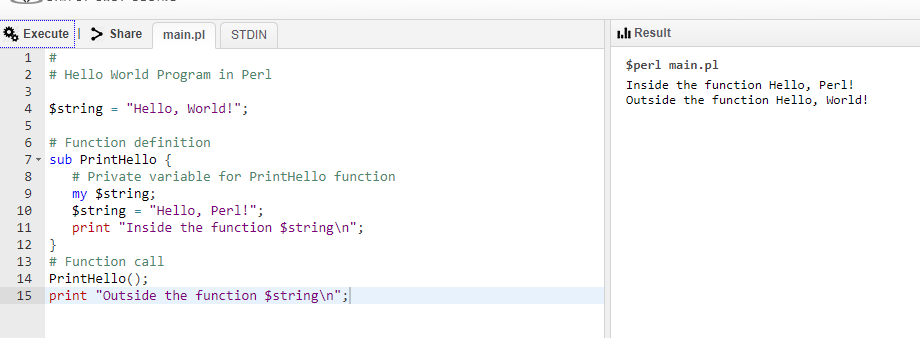
Przykład – znajdź największą liczbę spośród podanych:



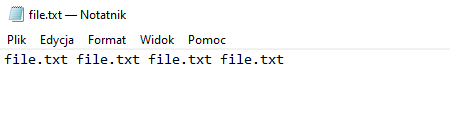
## Funkcja może zwracać arraya:

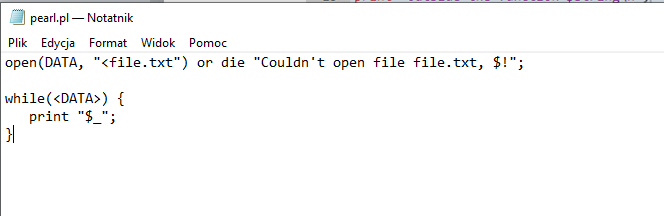


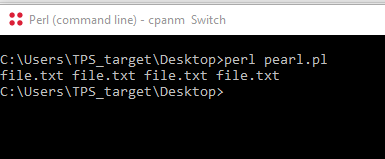
# Zmienne prywatne i globalne:



Odczyt z pliku txt umiezczonego w tej samej lokalizacji co odpalany:



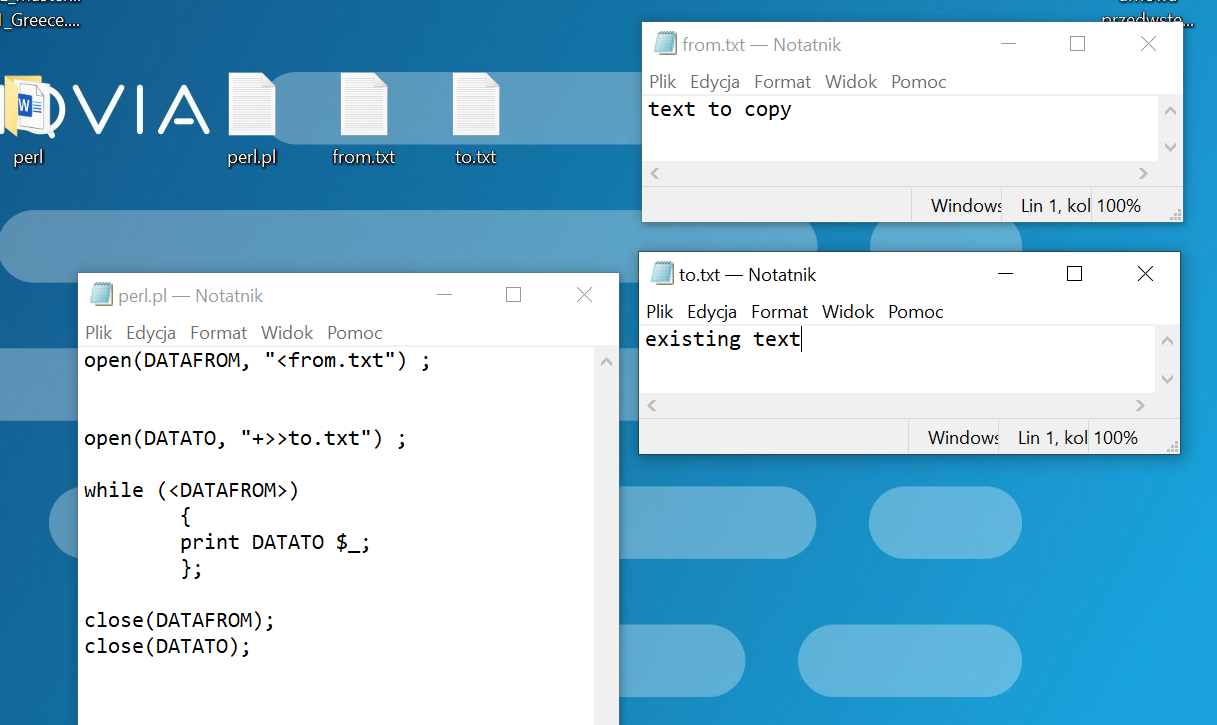




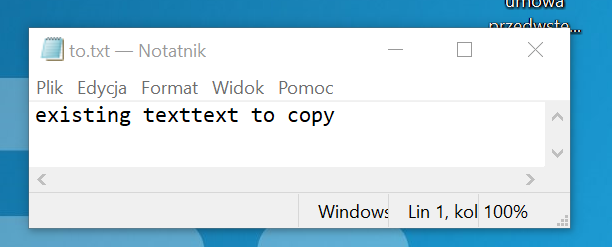
open(DATA, "<file.txt")

Zaznaczony fragment odpowiada za to, że odpalamy plik w trybie read only.

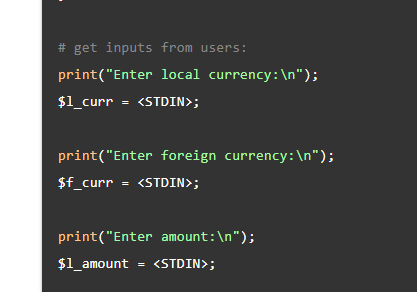
Kopiowanie z pliku do pliku:



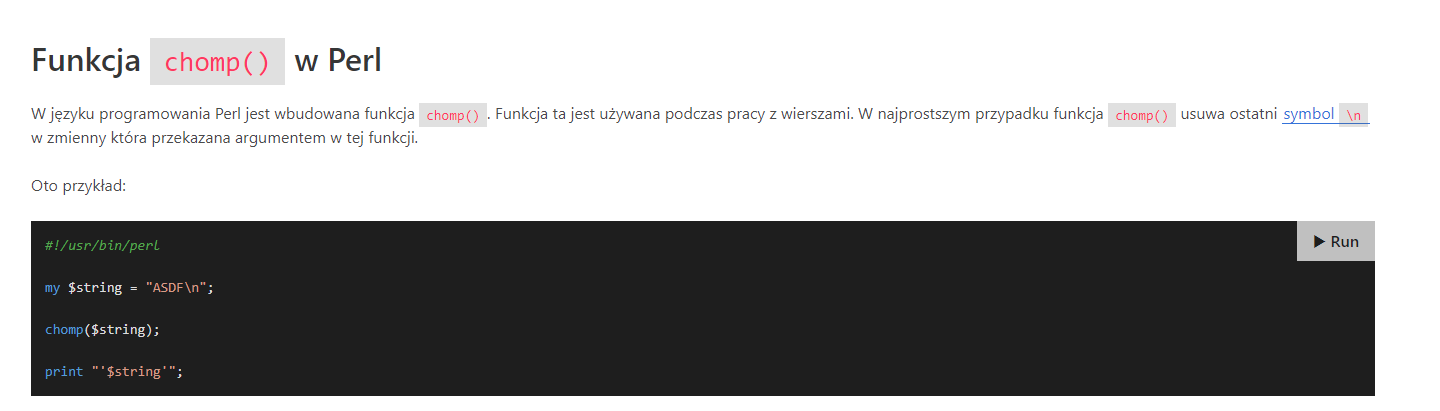
Zawartość pliku po odpaleniu skryptu:



# Wprowadzanie wartości z terminala do zmiennych:



# Funkcja chomp():

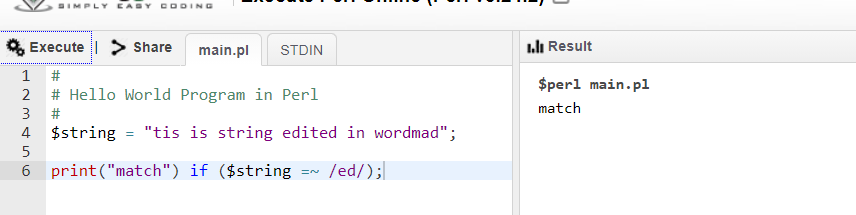


Sprawdzanie czy w podanym tekście są cyfry lub litery:

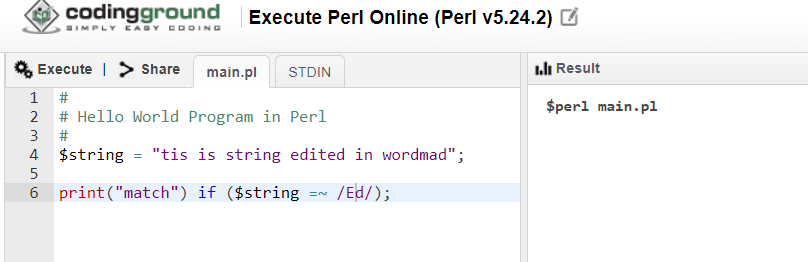


# Regular expressions:

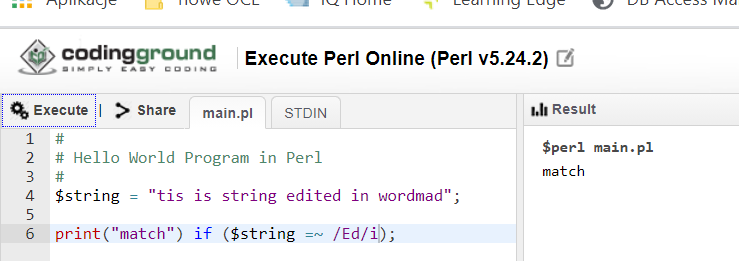
Sprawdzamy czy w podanym tekście jest string ‘ed’:



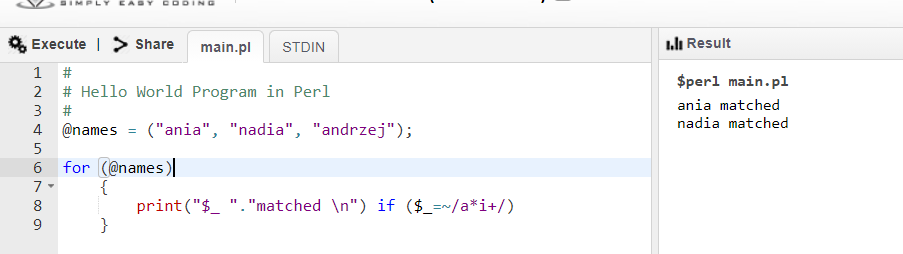
Wielkość znaków ma znaczenie. W takim przypadku „Ed” nie zostanie wyszukane:

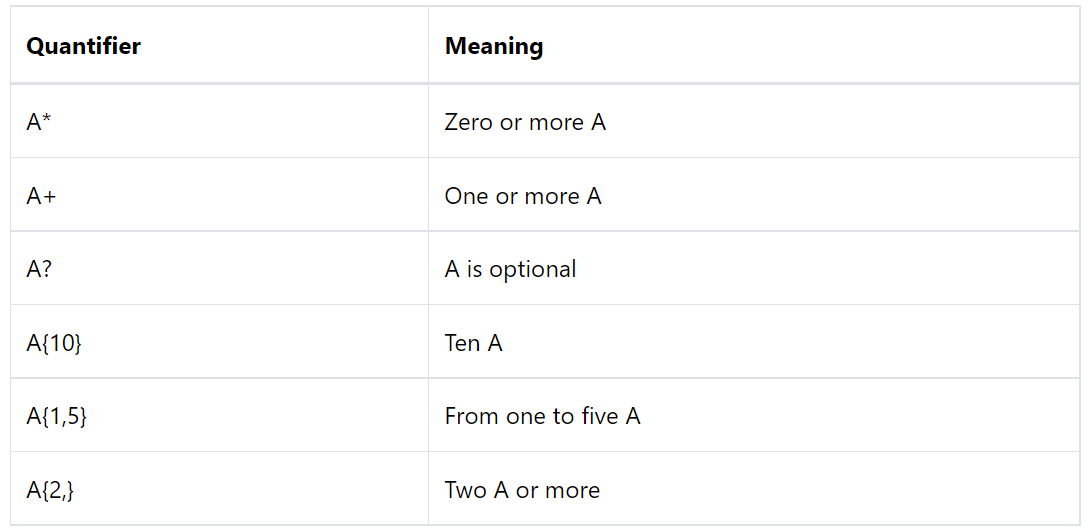


Ale jeśli zrobimy w ten sposób to wielkość znaków przestanie mieć znaczenie:

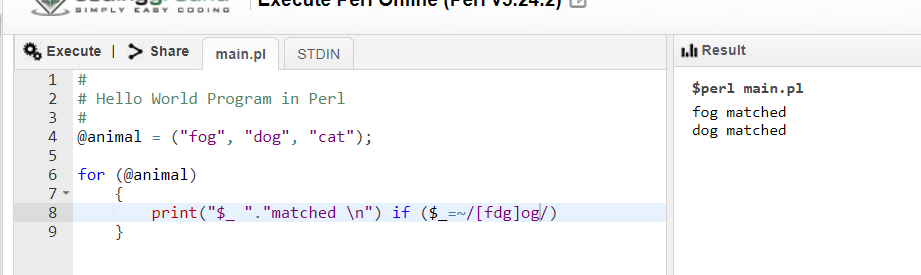


Teraz wyszukujemy w ciągu wszystkich jego elementów w którym występuje zero lub więcej liter „a”, po których jest co najmniej jedna litera „i”:





W nawiasach klamrowych umieszczamy zbiór możliwych liter:

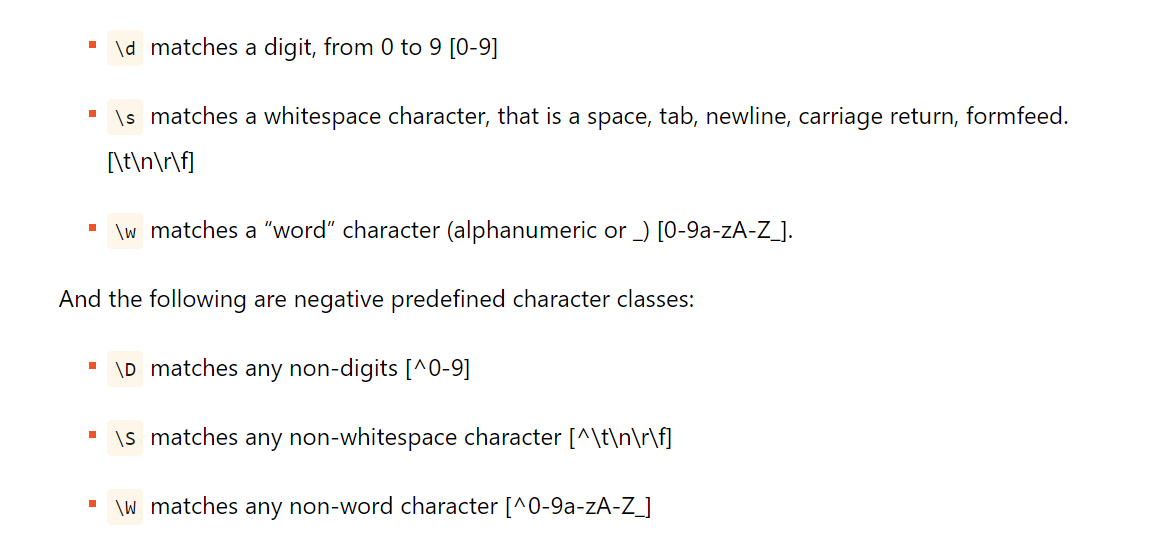


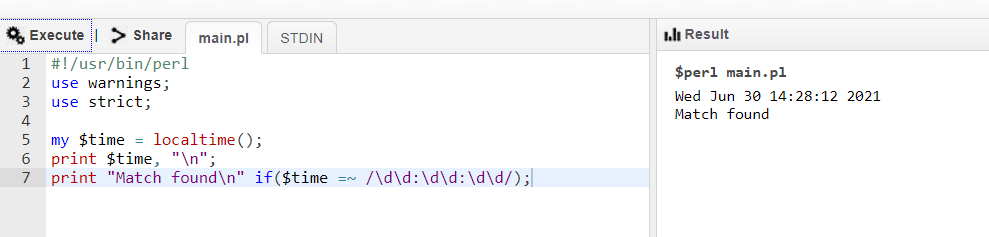
Czyli w tym przypadku wyszukane mogą być wyrazy: „fog”, „dog”, „gog”.

/elem[012345]/; # matches 'elem0', 'elem1'... or 'elem5'

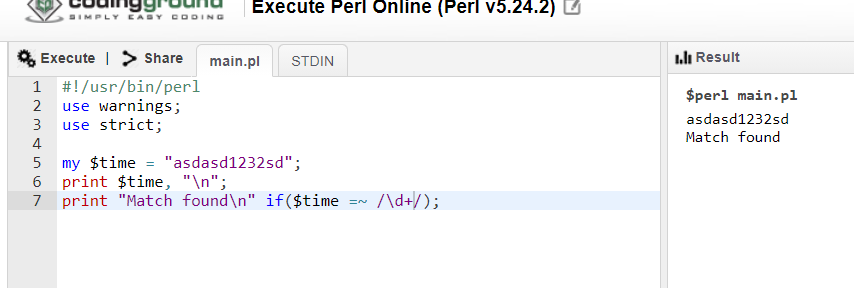
/[elem[0-1000]]/;

/[a-z]1/; matches a1,b1,... to z1

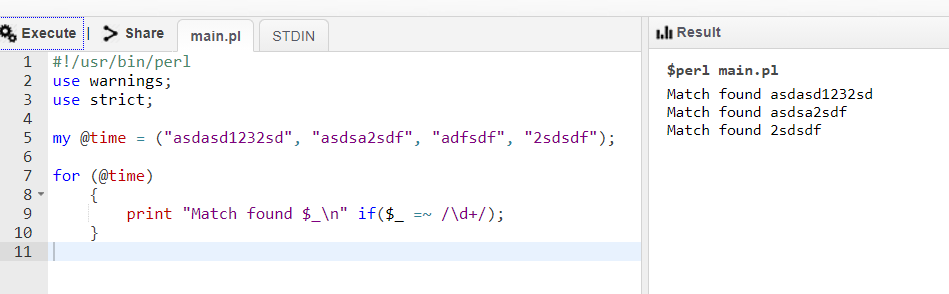


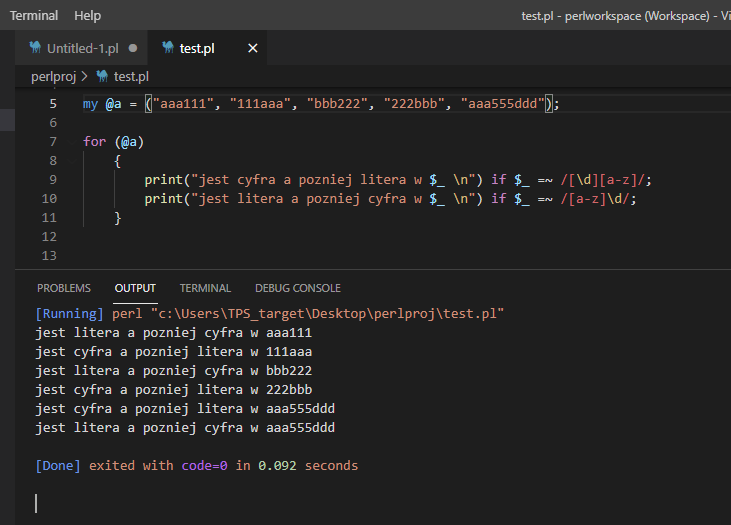


Sprawdzamy czy w stringu są cyfry (\d+):

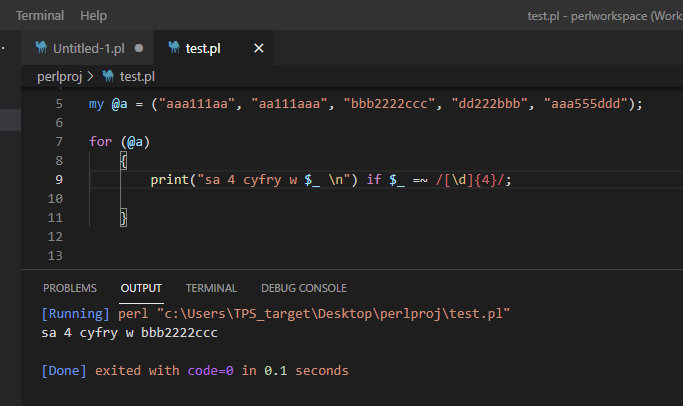


To samo tylko z macierzą:

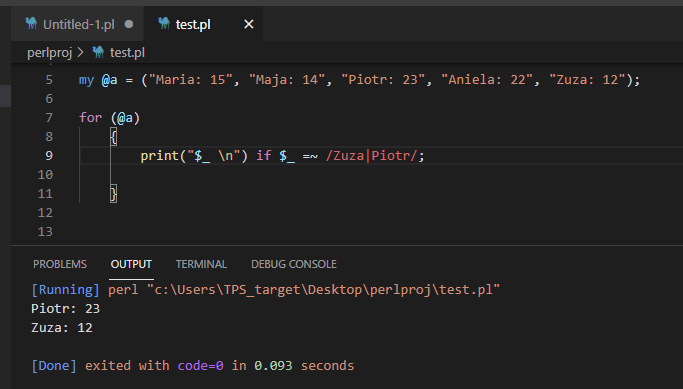




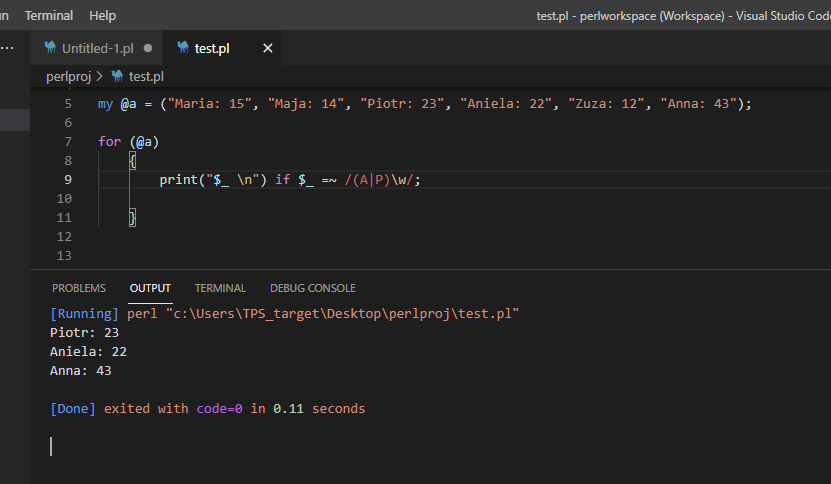
## Szukamy w którym stringu są 4 cyfry pod rząd:



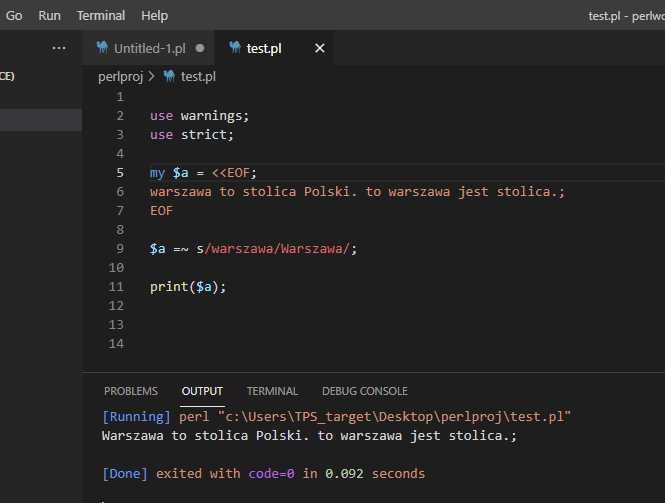
Wyszukiwanie ze znaczkiem „|”:



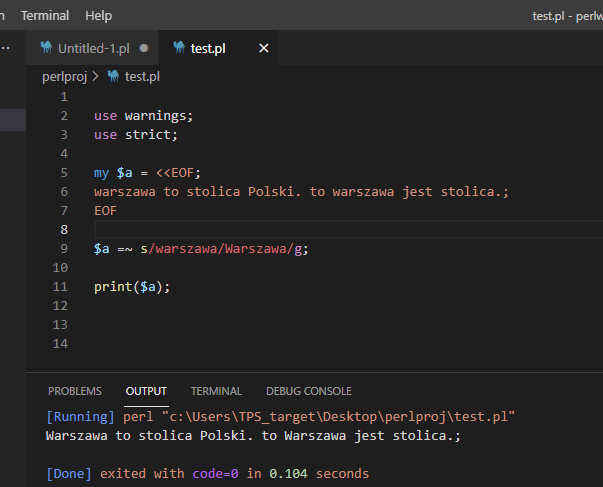
Wyszukiwanie tylko imion, które zaczynają się na „A” lub „P”:



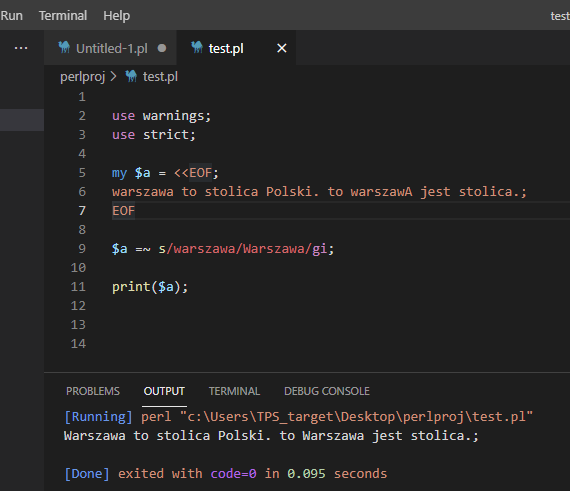
Wyszukaj i zamień tekst (pierwsze wystąpienie):



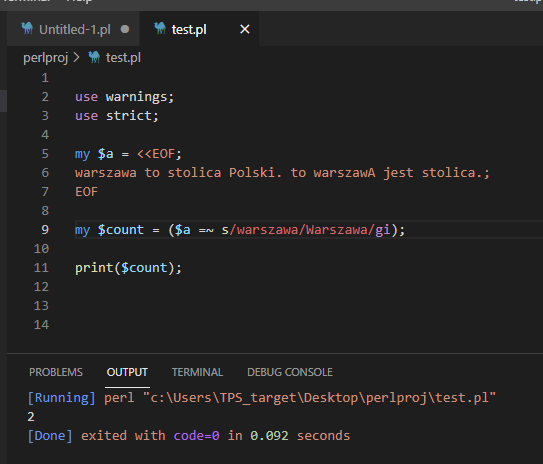
## Wyszukaj i zamień tekst (wszystkie wystąpienia):



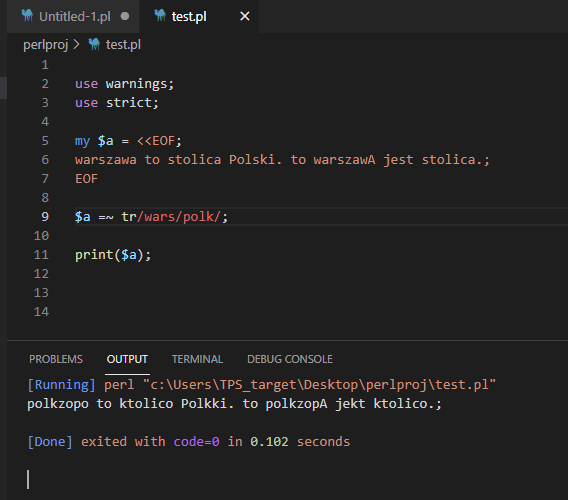
Wyszukaj i zamień tekst bez względu na wielkość znaków:



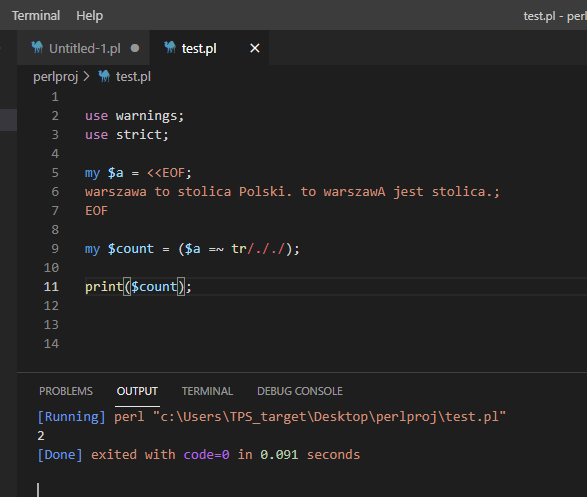
## Zlicz zmiany, które zostały wykonane:



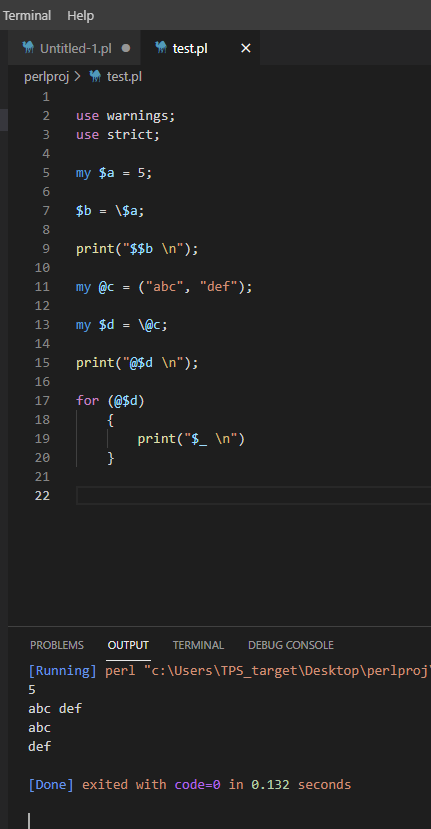
## Zamiana znak po znaku:



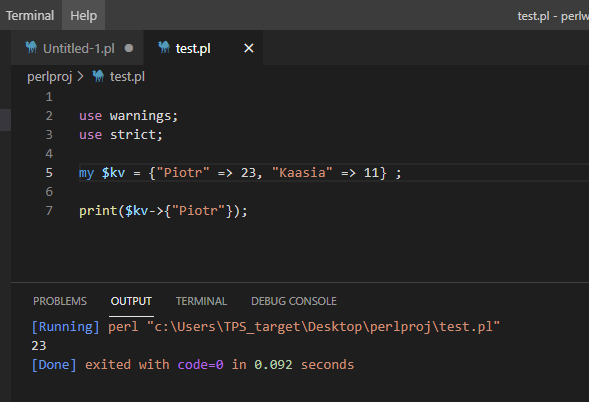
Może być to użyteczne gdy chcemy zliczyć kropki w stringu:



# Wskaźniki:



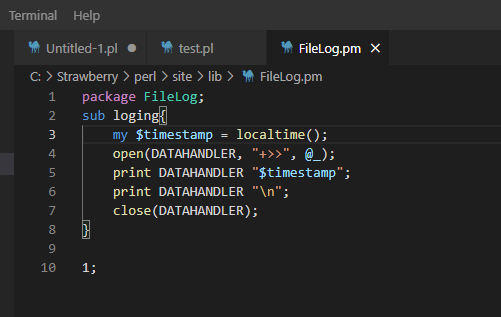
Klucz wartość można definiować jako zmienną static i odwoływać się w pewien specyficzny sposób. Nazywa się to anonymous hash references:



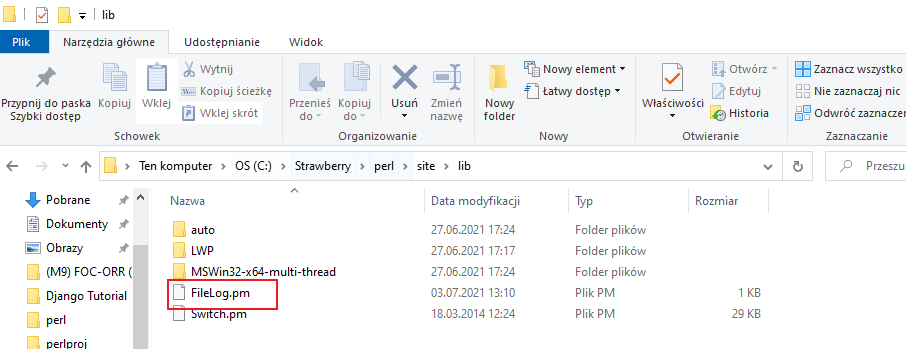
# Moduł

Modułu zawiera kod, który sobie sami wymyślimy w tym funkcje. Tego samego modułu można używać w różnych programach i to jest jego zaleta.

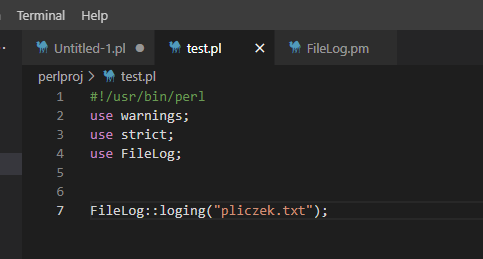
Tworzymy moduł który będzie zapisywał logi w pliku płaskim. Log to będzie zwykły timestamp. Tworzymy mofuł o takim ciele:



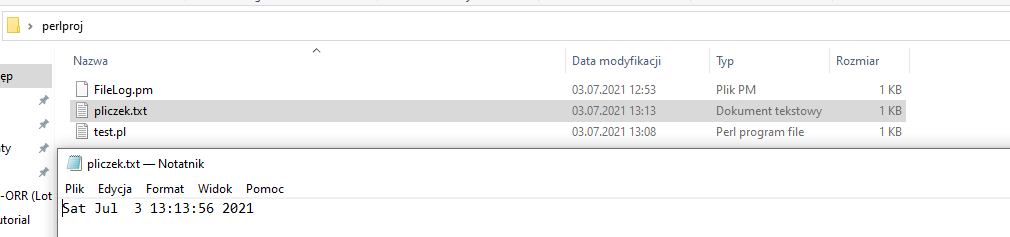
I umieszczamy go w takiej lokalizacji:



Tworzymy aplikację, która używa modułu o takim ciele:

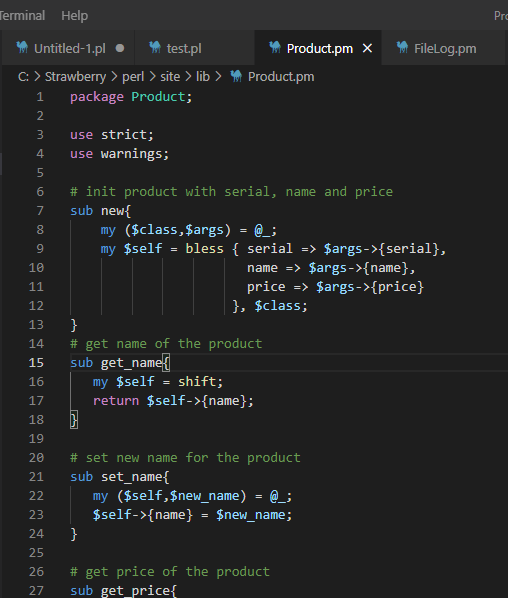


I w lokalizacji gdzie mamy naszą aplikację (aplikacja nazywa się test.pl) tworzy nam się plik o takiej zawartości:



Klasy:

Tworzę plik Product.pm o takiej zawartości i lokalizacji:



package Product;

use strict;

use warnings;

# init product with serial, name and price

sub new{

my ($class,$args) = @\_;

my $self = bless { serial => $args->{serial},

name => $args->{name},

price => $args->{price}

}, $class;

}

# get name of the product

sub get\_name{

my $self = shift;

return $self->{name};

}

# set new name for the product

sub set\_name{

my ($self,$new\_name) = @\_;

$self->{name} = $new\_name;

}

# get price of the product

sub get\_price{

my $self = shift;

return $self->{price};

}

# set price for the product

sub set\_price{

my ($self,$new\_price) = @\_;

$self->{price} = $new\_price;

}

# get serial

sub get\_serial{

my $self = shift;

return $self->{serial};

}

# set serial

sub set\_serial{

my ($self,$new\_price) = @\_;

$self->{price} = $new\_price;

}

# return formatted string of the product

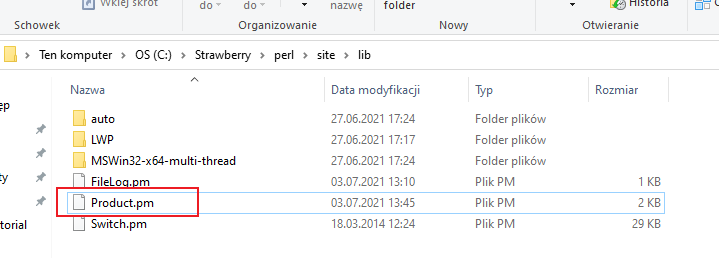
sub to\_string{

my $self = shift;

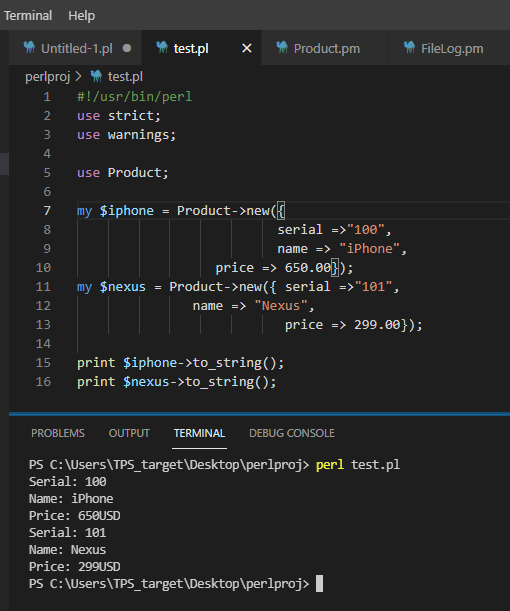
return "Serial: $self->{serial}\nName: $self->{name}\nPrice: $self->{price}USD\n";

}

1;



Tak tworzymy obiekty danej klasy i używamy ich metod:



#!/usr/bin/perl

use strict;

use warnings;

use Product;

my $iphone = Product->new({

                          serial =>"100",

                          name => "iPhone",

                  price => 650.00});

my $nexus = Product->new({ serial =>"101",

               name => "Nexus",

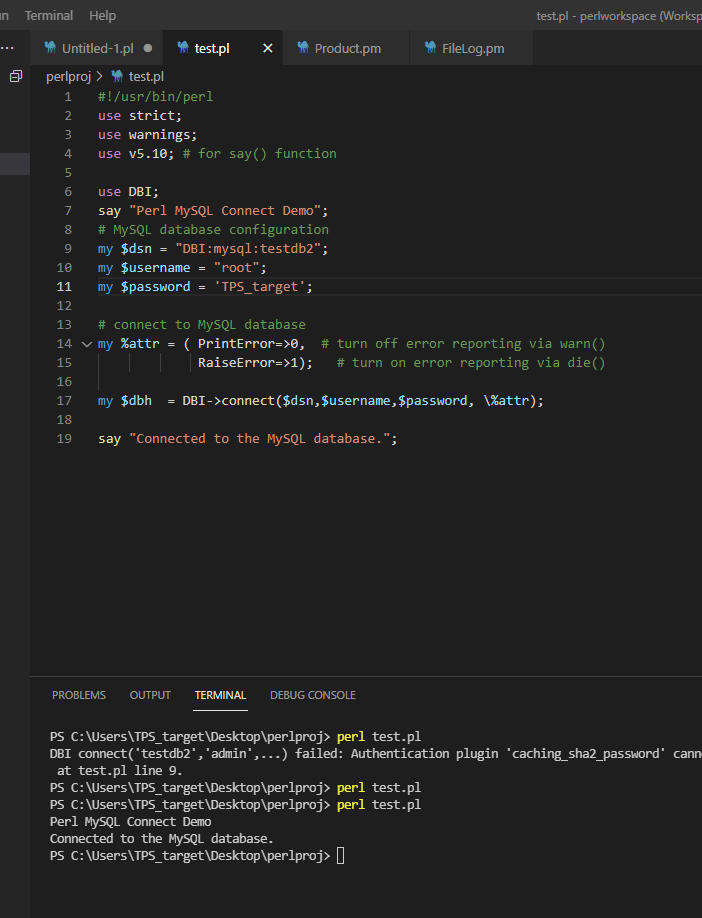
                           price => 299.00});

print $iphone->to\_string();

print $nexus->to\_string();

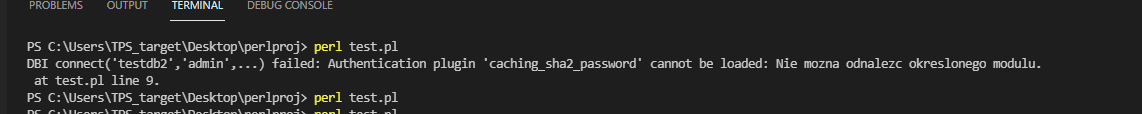
# Baza danych

Połączenie z bazą:



Testdb2 to nazwa bazy.

Często wyskakiwał mi taki błąd:



To dlatego, że złego usera i hasło wpisywałem.

Create table w perl:

#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

say "Perl MySQL Connect Demo";

# MySQL database configuration

my $dsn = "DBI:mysql:testdb2";

my $username = "root";

my $password = 'TPS\_target';

# connect to MySQL database

my %attr = ( PrintError=>0,  # turn off error reporting via warn()

             RaiseError=>1);   # turn on error reporting via die()

my $dbh  = DBI->connect($dsn,$username,$password, \%attr);

my @ddl =     (

        # create tags table

        "CREATE TABLE tags (

            tag\_id int(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

            tag varchar(255) NOT NULL

            ) ENGINE=InnoDB;",

           # create links table

           "CREATE TABLE links (

          link\_id int(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

          title varchar(255) NOT NULL,

          url varchar(255) NOT NULL,

          target varchar(45) NOT NULL

        ) ENGINE=InnoDB;",

        # create link\_tags table

        "CREATE TABLE link\_tags (

          link\_id int(11) NOT NULL,

          tag\_id int(11) NOT NULL,

          PRIMARY KEY (link\_id,tag\_id),

          KEY fk\_link\_idx (link\_id),

          KEY fk\_tag\_idx (tag\_id),

          CONSTRAINT fk\_tag FOREIGN KEY (tag\_id)

             REFERENCES tags (tag\_id),

          CONSTRAINT fk\_link FOREIGN KEY (link\_id)

             REFERENCES links (link\_id)

        ) ENGINE=InnoDB"

           );

# execute all create table statements

for my $sql(@ddl){

  $dbh->do($sql);

}

say "All tables created successfully!";

# disconnect from the MySQL database

$dbh->disconnect();

## Insert

Program pozwala na wprowadzanie danych do bazy z użyciem linii komend:

#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

say "Perl MySQL Connect Demo";

# MySQL database configuration

my $dsn = "DBI:mysql:testdb2";

my $username = "root";

my $password = 'TPS\_target';

# connect to MySQL database

my %attr = ( PrintError=>0,  # turn off error reporting via warn()

             RaiseError=>1);   # turn on error reporting via die()

my $dbh  = DBI->connect($dsn,$username,$password, \%attr);

my @links = get\_links();

my $sql = "INSERT INTO links(title,url,target)

       VALUES(?,?,?)";

my $stmt = $dbh->prepare($sql);

# execute the query

foreach my $link(@links){

  if($stmt->execute($link->{title}, $link->{url}, $link->{target})){

    say "link $link->{url} inserted successfully";

  }

}

$stmt->finish();

# disconnect from the MySQL database

$dbh->disconnect();

sub get\_links{

   my $cmd = '';

   my @links;

   # get links from the command line

   my($title,$url,$target);

   # repeatedly ask for link data from command line

   do{

     say "title:";

     chomp($title = <STDIN>);

     say "url:";

     chomp($url = <STDIN>);

     say "target:";

     chomp($target = <STDIN>);

     #

     my %link = (title=> $title, url=> $url, target=> $target);

     push(@links,\%link);

     print("\nDo you want to insert another link? (Y/N)?");

     chomp($cmd = <STDIN>);

     $cmd = uc($cmd);

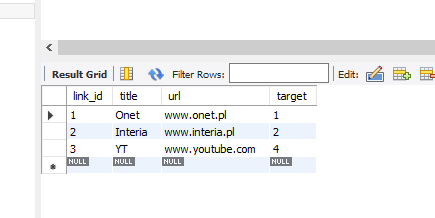
   }until($cmd eq 'N');

   return @links;

}

## Select

Mam taka tabelę:



#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

say "Perl MySQL Connect Demo";

# MySQL database configuration

my $dsn = "DBI:mysql:testdb2";

my $username = "root";

my $password = 'TPS\_target';

# connect to MySQL database

my %attr = ( PrintError=>0,  # turn off error reporting via warn()

             RaiseError=>1);   # turn on error reporting via die()

my $dbh  = DBI->connect($dsn,$username,$password, \%attr);

query\_links($dbh);

# disconnect from the MySQL database

$dbh->disconnect();

sub query\_links{

  # query from the links table

  my ($dbh) = @\_;

  my $sql = "SELECT title,

                    url

             FROM links";

  my $sth = $dbh->prepare($sql);

  # execute the query

  $sth->execute();

  while(my @row = $sth->fetchrow\_array()){

     printf("%s\t%s\n",$row[0],$row[1]);

  }

  $sth->finish();

}

### Select z wpisywanym warunkiem na target:

#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

say "Perl MySQL Connect Demo";

# MySQL database configuration

my $dsn = "DBI:mysql:testdb2";

my $username = "root";

my $password = 'TPS\_target';

# connect to MySQL database

my %attr = ( PrintError=>0,  # turn off error reporting via warn()

             RaiseError=>1);   # turn on error reporting via die()

my $dbh  = DBI->connect($dsn,$username,$password, \%attr);

say "target:";

chomp(my $target = <STDIN>);

query\_links\_by\_target($dbh, $target);

# disconnect from the MySQL database

$dbh->disconnect();

sub query\_links\_by\_target{

  # query from the links table by target

  my ($dbh,$target) = @\_;

  my $sql = "SELECT title,url,target

             FROM links

             WHERE target = ?";

  my $sth = $dbh->prepare($sql);

  # execute the query

  $sth->execute($target);

  while(my @row = $sth->fetchrow\_array()){

     print("$row[0] $row[1] $row[2]");

  }

  $sth->finish();

}

## UPDATE

Asdsa

#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

# MySQL database configurations

my $dsn = "DBI:mysql:perlmysqldb";

my $username = "root";

my $password = '';

say "Perl MySQL Update Data Demo";

# connect to MySQL database

my %attr = ( PrintError=>0, RaiseError=>1);

my $dbh = DBI->connect($dsn,$username,$password, \%attr);

# update statement

my $sql = "UPDATE links

SET title = ?,

url = ?,

target = ?WHERE id = ?";

my $sth = $dbh->prepare($sql);

my $id = 2;

my $title = "Perl MySQL Update Data Tutorial";

my $url = "https://www.mysqltutorial.org/perl-mysql/perl-mysql-update/";

my $target = "\_self";

# bind the corresponding parameter

$sth->bind\_param(1,$title);

$sth->bind\_param(2,$url);

$sth->bind\_param(3,$target);

$sth->bind\_param(4,$id);

# execute the query

$sth->execute();

say "The record has been updated successfully!";

$sth->finish();

# disconnect from the MySQL database

$dbh->disconnect();

## DELETE

#!/usr/bin/perl

use strict;

use warnings;

use v5.10; # for say() function

use DBI;

say "Perl MySQL Delete Data Demo";

# MySQL database configurations

my $dsn = "DBI:mysql:perlmysqldb";

my $username = "root";

my $password = '';

# connect to MySQL database

my %attr = (PrintError=>0, RaiseError=>1);

my $dbh = DBI->connect($dsn,$username,$password, \%attr);

# delete 1 row with id 1

# say "Link with id 1 deleted successfully!" if delete\_one\_row($dbh,1);

# delete all rows in the clinks table

# say "All links deleted successfully!" if delete\_all\_rows($dbh);

# disconnect from the MySQL database

$dbh->disconnect();

sub delete\_one\_row {

# delete one row from table

# $dbh: database handle

# $link\_id: id of the link that need to delete

my($dbh,$link\_id) = @\_;

my $sql = "DELETE FROM clinks WHERE link\_id = ?";

my $sth = $dbh->prepare($sql);

return $sth->execute($link\_id);

}

sub delete\_all\_rows {

# delete all rows in the clinks table

my($dbh) = @\_;

my $sql = "TRUNCATE TABLE clinks";

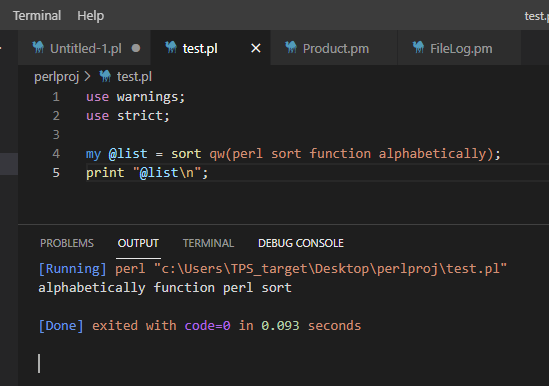
my $sth = $dbh->prepare($sql);

return $sth->execute();

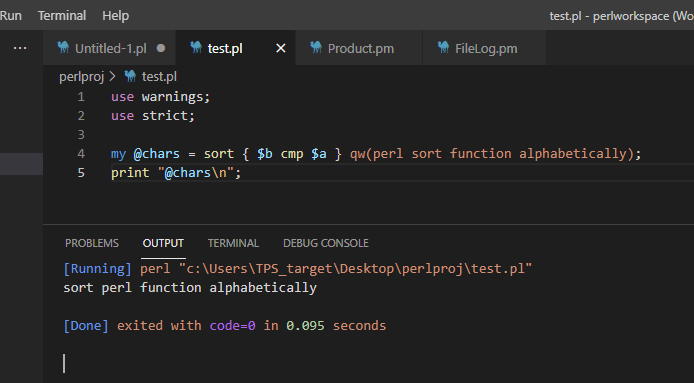
}

# Sortowanie:

## Sortowanie znaków:

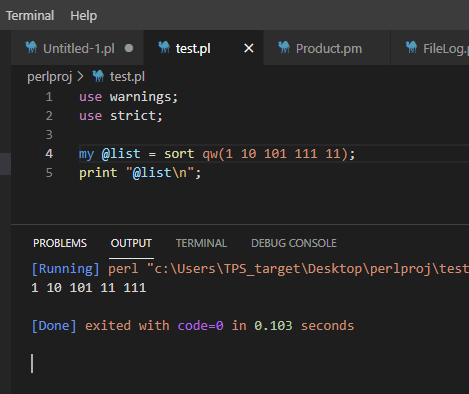


Sortowanie odwrotne znaków:

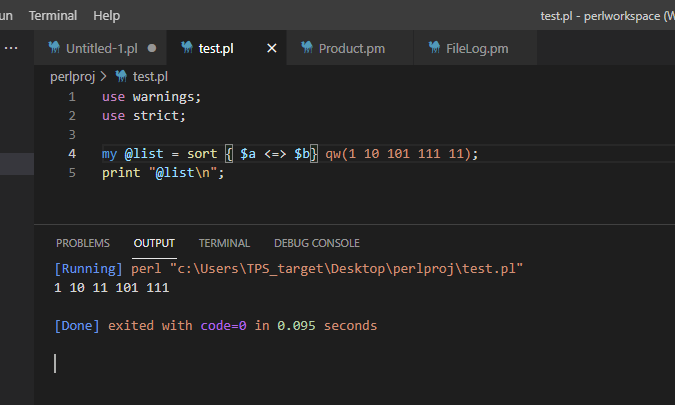


## Sortowanie cyfr:

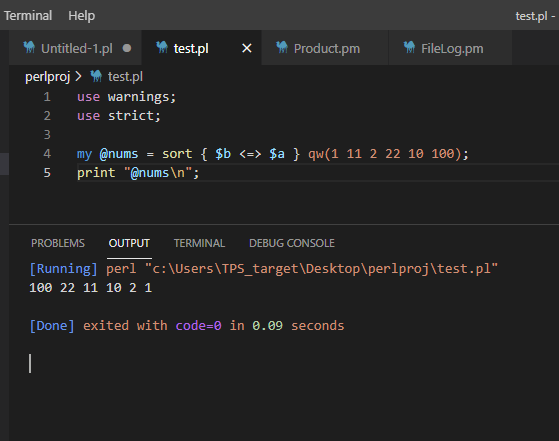
Jeśli spróbujemy tak z cyframi to będzie tak:



Żeby cyfry poprawnie posortować to trzeba tak:

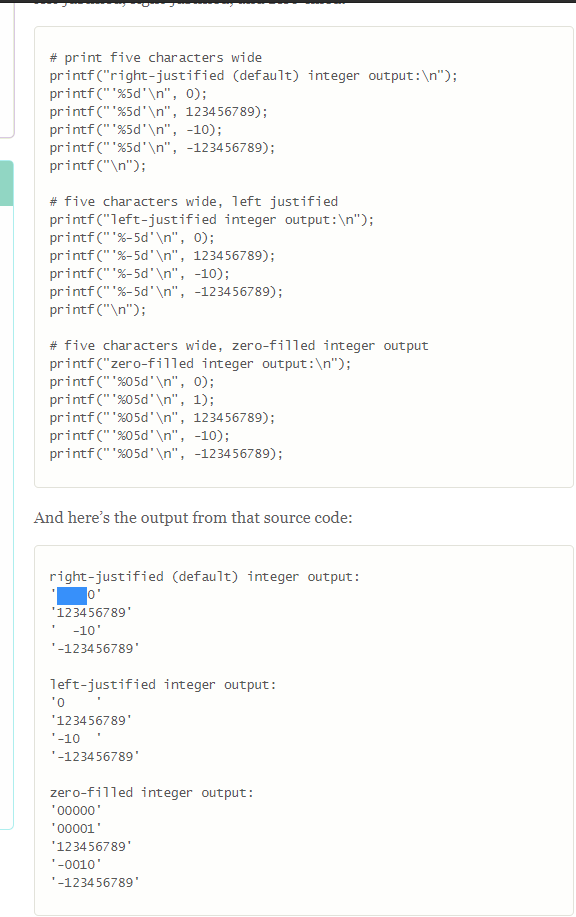


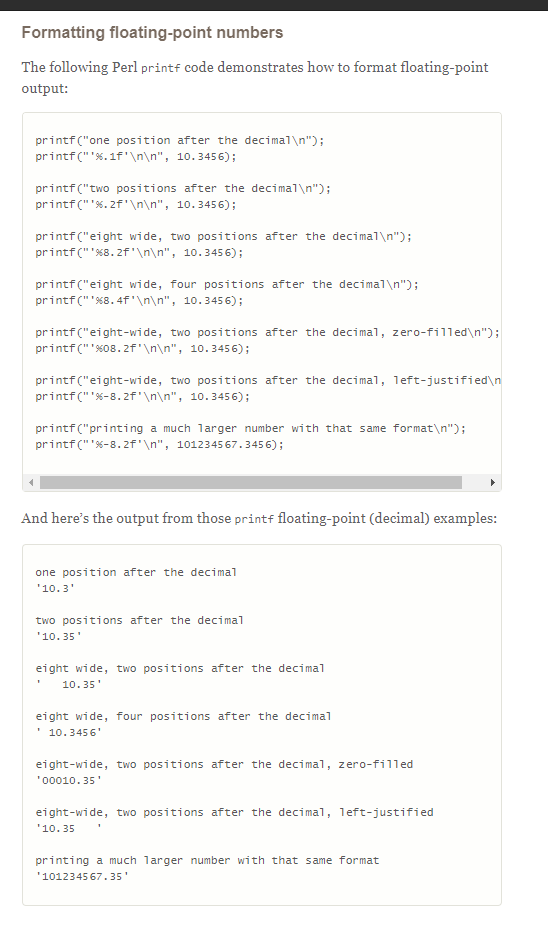
Sortowanie odwrotne cyfr:



# Printf – formatowanie:

Integers







## Sortowanie w arrayu:

