Temat: Struktura języka PHP.

Zadanie:

Odszukaj w serwisie internetowym Wikipedii informacje na temat języka programowania PHP.

PHP (ang. Hypertext Preprocessor) jest obiektowym językiem programowania wykonywanym po stronie serwera (back-end) przeznaczonym do tworzenia i generowania aplikacji internetowych. Jako język skryptowy stosuje prostą składnię. Kod umieszcza się w plikach o domyślnym rozszerzeniu `*.php`, a następnie umieszcza na serwerze posiadającym interpreter PHP. Strony www zawierające skrypty PHP nazywamy aplikacjami www. Jeśli zwykłemu plikowi `html` nadamy rozszerzenie `php` to zostanie on wyświetlony prawidłowo, mimo że nie jest skryptem PHP. Dzieje się tak dlatego, że parser (program analizujący składnię skryptu) PHP ma dwa tryby pracy:

- html – bez przetwarzania wyświetla całą treść,
- php – traktuje treść jako skrypt do przetworzenia.

Wyróżnia się trzy główne obszary zastosowań PHP:

- skrypty po stronie serwera,
- skrypty wywoływane z wiersza poleceń,
- aplikacje po stronie klienta – zaawansowane funkcje PHP umożliwiające tworzenie aplikacji desktopowych.

Skrypt wykonywany po stronie serwera nie jest widoczny dla użytkownika. Skrypty PHP umieszczane w dokumentach HTML są reprezentowane przez:

- znaczniki kanoniczne,
- znaczniki typu SGML,
- znaczniki typu ASP,
- znaczniki HTML.

Znaczniki kanoniczne są standardowymi znacznikami PHP. Skrypty PHP są plikami tekstowymi. Do ich tworzenia można wykorzystać dowolny edytor tekstu. Umieszczając kod PHP w kodzie html, należy użyć następujących znaczników:

```
<?php       #znacznik otwierający sekcję PHP
  instrukcje skryptu
?>          #znacznik zamykający
```

Znaczniki SGML są zapisywane w postaci skróconej i wymagają włączenia opcji `short_open_tag`, a kod ma postać według poniższego schematu:

```
<?
  instrukcje skryptu
?>
```

Znaczniki ASP wymagają włączenia opcji `asp_tags` i mają następującą postać:

```
<%
  instrukcje skryptu
%>
```

Znaczniki HTML pozwalają umieszczać skrypty PHP z użyciem następującej konstrukcji:

```
<script language="php">
  instrukcje skryptu
</script>
```

Język PHP umożliwia przetwarzanie danych z formularzy, dynamiczne generowanie zawartości stron internetowych, wykonywanie operacji na bazach danych, dynamiczne tworzenie dokumentów PDF i obrazów oraz animacji.

Zadanie:

Zapoznaj się z filmem publikowanym na witrynie internetowej serwisu "Pasja informatyki" https://miroslawzelent.pl/kurs-html/technologie-webowe-html-css-php-javascript-mysql-frameworki/

Sekcje PHP mogą być umieszczane w dowolnym miejscu dokumentu html, a ich liczba jest nieograniczona. Przykładowy dokument html (`index.php`) z umieszczonym skryptem PHP bezpośrednio w kodzie HTML wygląda następująco:

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Pierwszy skrypt PHP</title>
</head>
<body>
  <h1>My first PHP page</h1>
  <?php     // w słowach kluczowych w PHP wielkość liter nie ma znaczenia
    echo "To jest mój pierwszy skrypt<br>";
    ECHO "To jest mój pierwszy skrypt<br>";
    EcHo "To jest mój pierwszy skrypt<br>";
  ?>
</body>
</html>
```

W skryptach PHP można umieszczać **komentarze**. Są one widoczne w kodzie źródłowym skryptu, natomiast w trakcie jego przetwarzania są usuwane i niewidoczne w przeglądarce. W języku PHP występują trzy rodzaje komentarzy:

- Blokowe – zaczyna się od znaków /* a kończy znakami */. Wszystko co znajduje się pomiędzy znakami jest ignorowane podczas przetwarzania skryptu.
- Jednowierszowe – zaczyna się od // i kończy w bieżącej linii skryptu.
- Jednowierszowe uniksowe – są podobne do jednowierszowych. Zaczynają się od # i obowiązują do końca linii.

Przykład zastosowania komentarzy w skryptach PHP na kilka sposobów:

```
<?php
  echo "Przykłady komentarzy w PHP:<br>";
  //komentarz jednowierszowy
  /*komentarz
  blokowy, którego nie widać na stronie */
  #komentarz jednowierszowy uniksowy
  echo "Welcome Home!"; // Outputs a welcome message
  // echo "Welcome Home!";
  /*
  echo "Welcome to my home!";
  echo "Mi casa su casa!";
  */
  echo "<br>Mój dom jest twoim domem!<br>";
  $x = 5 /* + 15 */ + 5;      //zapobieganie wykonywaniu części wewnątrz linii kodu
  echo $x;
?>
```

Zadanie:

Zapoznaj się z opisem podstaw języka skryptowego PHP publikowanych na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/default.asp

Informacje o konfiguracji PHP uzyskamy za pomocą skryptu:

```
<?php phpinfo(); ?>
```

Temat: Instalacja oprogramowania XAMPP.

Aby uruchamiać skrypty PHP w usłudze IIS (Internet Information Services) w systemie Windows, należy zainstalować i skonfigurować PHP jako moduł dla IIS. Podczas instalacji usługi IIS należy w Usłudze sieci Web (Web Services) w sekcji Projektowanie aplikacji (Application Development) zaznaczyć Interfejs CGI (dla FastCGI) oraz opcjonalnie ISAPI Extensions i ISAPI Filters. Następnie należy zainstalować obsługę PHP np. Microsoft Web Platform Installer (Web PI) lub NTS (Non Thread Safe, https://windows.php.net/download/). W przypadku rozpakowania archiwum NTS do folderu np. `c:\php` należy dodać ten katalog do zmiennej `PATH`:

```
setx /M PATH "%PATH%;c:\php"
```

Kolejnym krokiem jest konfiguracja IIS Manager, gdzie przechodzimy do Handler Mappings => Add Module Mapping i dodajemy nowy wpis: `Request path: *.php, Module: FastCgiModule, Executable: c:\php\php-cgi.exe, Name: np. PHP via FastCGI.` Następnie w ustawieniach witryny w Default Document dodajemy do listy `index.php`. Po zatwierdzeniu ustawień w widoku głównym w `FastCGI Settings` należy dodać nowy wpis do `c:\php\php-cgi.exe`. W celu sprawdzenia poprawności działania usługi PHP można w katalogu strony WWW (domyślnie `c:\inetpub\wwwroot`) utworzyć plik `info.php`, który będzie dostępny pod adresem http://localhost/info.php. Przykładowa treść pliku `info.php` (wyświetli konfigurację PHP):

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Skrypty PHP</title>
</head>
<body>
  <h1>My first PHP page</h1>
  <?php phpinfo(); ?>
</body>
</html>
```

Dodatkowo można dostosować pliki `c:\php\php.ini` (skopiuj `php.ini-development` jako `php.ini`) i ewentualnie `c:\inetpub\wwwroot\web.config`. Poniżej przykładowa treść pliku `web.config`:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <add name="PHP_via_FastCGI" path="*.php" verb="*" modules="FastCgiModule"
scriptProcessor="c:\PHP\php-cgi.exe" resourceType="Either" requireAccess="Script"
/>
    </handlers>
    <defaultDocument>
      <files>
        <add value="index.php" />
      </files>
    </defaultDocument>
  </system.webServer>
</configuration>
```

Pakiety darmowego oprogramowania WAMP, XAMPP, Laragon mają gotową konfigurację PHP oraz Apache/MySQL.

Zadanie:

Odszukaj w serwisie internetowym Wikipedii informacje na temat oprogramowania XAMPP.

Zadanie:

Przeprowadź instalację oprogramowania XAMPP (https://www.apachefriends.org/pl/index.html) na maszynie wirtualnej serwera win2022 wg instrukcji przedstawionej na filmie publikowanym na stronie https://miroslawzelent.pl/kurs-php/instalacja-pakietu-xampp/

Uwaga!

- Podczas instalacji oprogramowania XAMPP pozostawić ustawienia domyślne, odznaczyć jedynie wyświetlanie informacji o instalatorze Bitnami.
- Panel kontrolny oprogramowania XAMP uruchomimy plikiem: `c:\xampp\xampp-control.exe`.
- Po zainstalowaniu oprogramowania XAMPP uruchamiamy usługi apache i msql.
- Plik domyślny serwera www umieszczono w katalogu `c:\xampp\htdocs`. Odwołanie do pliku następuje poprzez adres `http://localhost/index.php`. Plik powinien mieć nazwę `index.php` oraz zawierać następującą treść (lub z poprzedniego tematu):

```
<?php
  echo "To jest mój pierwszy skrypt";
  echo phpversion();
?>
```

– Po skopiowaniu strony do katalogu `c:\xampp\htdocs\R8\` w celu jej sprawdzenia należy w przeglądarce wpisać adres http://localhost/r8/index.php.
– Domyślny folder stron serwera www możemy zmienić w pliku konfiguracyjnym `httpd.conf` dostępnym w panelu kontrolnym XAMPP'a, np.: zamieniamy:

```
DocumentRoot "c:\xampp\htdocs"
<Directory "c:\ xampp\htdocs ">
```

na

```
DocumentRoot "c:\pai"
<Directory "c:\pai">
```

Temat: Typy danych w języku PHP.

Zmienne w programowaniu służą do przechowywania danych i wyników wykonywanych operacji. PHP jest językiem słabo typowanym i dynamicznie typowanym, co oznacza, że zmienne mogą przechowywać różne typy danych i nie trzeba ich deklarować z góry. Wartości przypisane zmiennym określają ich typ. Wartości zmiennych mogą przybierać następujące typy (ang. variable types):

– liczbowy (**`integer`** (int) – liczby całkowite, np. 1, -42, 0, **`float`** (double, real) – liczby zmiennoprzecinkowe (z kropką), np. 3.14, -0.001),
– łańcucha znaków (**`string`** – ciągi tekstowe, np. "Hello", 'PHP'),
– tablicowy (**`array`** – tablice, np. [1, 2, 3], ['name' => 'John']),
– obiektowy (**`object`** – obiekty, utworzone z klas),
– logiczny (**`boolean`** (bool) – wartości logiczne, true lub false),
– specjalny (**`resource`** – specjalny typ odwołań do zasobów zewnętrznych, np. połączenia do bazy danych),
– brak wartości, typ specjalny, zmienna nie przechowuje żadnej wartości (**`null`**, np. $var = null;).

Dodatkowo typ całkowity (**`integer`**) może występować w następujących formatach:

– dziesiętnym (domyślny),
– binarnym (0b przed liczbą),
– ósemkowym (0 przed liczbą),
– szesnastkowym (0x przed liczbą).

Zmienne nie muszą być deklarowane i określane rodzajem typu. Inicjalizacja zmiennej następuje podczas jej pierwszego użycia. Zmienne posiadają nazwę oraz przypisaną wartość. Nazwy zmiennych muszą zaczynać się od litery lub znaku podkreślenia i mogą składać się jedynie z liter, cyfr i znaku podkreślenia. **Wielkość liter w nazwach zmiennych ma znaczenie. Nazwę zmiennej w języku PHP poprzedzamy znakiem $.**

Przykład zastosowania zmiennych w języku PHP:

```
<?php
  $x=18;                //zmienna x przyjmuje wartość numeryczną
  $user="Jan Kowalski"; //zmienna łańcuchowa
  $zawod='nauczyciel';  //zmienna łańcuchowa
  $y=$x/3;              //zmienna liczbowa
  echo "Witamy Ciebie $user";
  echo "<br>Dodawanie liczb x oraz y:</br>$x + $y = ";
  echo $x + $y;
  echo "<br>lub<br>";
  echo "$x + $y = ".($x + $y);      //znak kropki łączy tekst i wartość zmiennej
  $txt = "W3Schools.com";
  echo "<br>I love " . $txt . "!<br>";
  echo "<br>I love $txt!<br>";      //można również pominąć znak kropki
  $color = "red";             //wielkość liter przy nazwach zmiennych ma znaczenie
  echo "My car is " . $color . "<br>";
  echo "My house is " . $COLOR . "<br>";
  echo "My boat is " . $coLOR . "<br>";
?>
```

Inny przykład zastosowania zmiennych w języku PHP:

```
<?php
  $pytanie="Jaka jest Twoja odpowiedź?";
```

```
  //zmienna łańcuchowa blokowa (heredoc), EOD - identyfikator
  $zadanie=<<<EOD
    W jaki sposób oznaczamy komentarze w języku PHP?<br>
    $pytanie <br>
EOD;
// powyższy znak końca linii należy zapisać od początku wiersza, bez spacji
  echo $zadanie;
  //zmienna łańcuchowa blokowa (nowdoc) bez rozpoznania zmiennych, EOD -
identyfikator
  $zadanie=<<<'EOD'
    W jaki sposób oznaczamy komentarze w języku PHP?<br>
    $pytanie <br>
EOD;
// powyższy znak końca linii należy zapisać od początku wiersza, bez spacji
  echo $zadanie;
?>
```

W przypadku zmiennych tablicowych przechowują one zbiory danych określonego typu. Elementy tablicy indeksowane są od 0. W PHP tablice definiujemy w następujący sposób:

```
  $tablica = array(wart1, wart2, ..., wartn);
```

Przykład zastosowanie typu tablicowego:

```
<?php
  // definicja i deklaracja tablic
  $tab1=array(1,2, 3,4 ,5);
  $tab2=[3,11, 54,-7,0];
  echo $tab1[0]."<br>";
  echo $tab2[4]."<br>";
  $tab3[0]="Jan";
  $tab3[1]="Anna";
  $tab3[]="Adam";
  $tab3[]="Kasia";
  echo $tab3[3]."<br>";
?>
```

Inny przykład zastosowanie typu tablicowego:

```
<?php
  echo "Sumowanie liczb z listy\n";
  $lista = [1, 2, 3, 4, 5];
  $suma = array_sum($lista);
  echo $suma . "\n";
?>
```

W języku PHP występują również tablice asocjacyjne (skojarzeniowe), w których do indeksowania używa się dowolnych ciągów znaków. W tablicach tego typu występują pary klucz => wartość.

Przykład zastosowanie tablicy asocjacyjnej:

```
<?php
  $uczniowie=array("u1" => "Kowalski", "u2" => "Nowak", "u3"=>"Zieliński");
  $uczniowie["u4"]="Życki";
  echo "Wyświetlenie danych z tablicy:<br>";
  echo "Pierwszy uczeń na liście: ".$uczniowie['u1']."<br>";
  echo "Ostatni uczeń na liście: ".$uczniowie["u4"]."<br>";
?>
```

Inny przykład zastosowania tablicy asocjacyjnej:

```
<?php
  $uczniowie=[
    "u1" => "Kowalski",
    "u2" => "Nowak",
    "u3"=>"Zieliński"
  ];
  $uczniowie["u4"]="Życki";
  echo "Wyświetlenie danych z tablicy:<br>";
  echo "Drugi uczeń na liście: ".$uczniowie['u2']."<br>";
  echo "Ostatni uczeń na liście: ".$uczniowie["u4"]."<br>";
```

```
?>
```

Przykład przypisania tej samej wartości do kilku zmiennych:

```
<?php
  $x = $y = $z = "Fruit";
  echo $x;
  echo " ".$y." ";
  echo $z;
?>
```

Do sprawdzenia typu zmiennej można wykorzystać następujące funkcje:

- `is_array()` – zwraca wartość `TRUE`, gdy zmienna jest typu tablicowego,
- `is_bool()` – zwraca wartość `TRUE`, gdy zmienna jest typu logicznego,
- `is_float()` – zwraca wartość `TRUE`, gdy zmienna jest typu zmiennoprzecinkowego,
- `is_interger()` – zwraca wartość `TRUE`, gdy zmienna jest typu całkowitego,
- `is_null()` – zwraca wartość `TRUE`, gdy zmienna jest typu pustego,
- `is_string()` – zwraca wartość `TRUE`, gdy zmienna jest typu,
- `gettype()` – zwraca typ zmiennej lub `unknow` dla typu nieokreślonego,
- `var_dump()` – zwraca typ zmiennej i wartość.

Przykład sprawdzenia typu zmiennej:

```
<?php
  echo "Funkcja var_dump zwraca typ danych i wartość."<br>";
  $x = 5;
  echo "Zmienna x: ";
  var_dump($x);
  echo "<br>Sprawdzenie innych typów danych:<br>";
  var_dump("John");
  echo "<br>";
  var_dump(3.14);
  echo "<br>";
  var_dump(true);
  echo "<br>";
  var_dump([2, 3, 56]);
  echo "<br>";
  var_dump(NULL);
  echo "<br>";
?>
```

W skryptach PHP można stosować **stałe**. Stała podobnie jak zmienna to również obszar pamięci przechowujący pewną wartość, z tą różnicą, że jej wartość nie może ulec zmianie podczas wykonywania skryptu. Prawidłowa nazwa stałej zaczyna się od litery lub znaku podkreślenia (bez znaku $ przed nazwa stałej). W przeciwieństwie do zmiennych, stałe są automatycznie globalne. Stałe definiowane są w języku PHP za pomocą funkcji `define()` lub za pomocą słowa kluczowego `const`. Funkcja `define()` przyjmuje dwa parametry: nazwę stałej (zasady nazewnictwa jak w przypadku zmiennych) oraz jej wartość a `const` jak poniżej, np.:

```
define("nazwa_stałej","wartość");
const MYCAR = "Volvo";
```

Przykład definiowania stałej w języku PHP:

```
<?php
  $x=3;
  $y=1;
  define("liczba3","6");
  $suma=$x+$y+ liczba3;
  echo "pierwsza zmienna = $x <br>";
  echo "druga zmienna = $y <br>";
  echo "trzecia stała = ". liczba3."<br>";
  echo "Wynik dodawania wynosi: $suma<br>";
  const MYCAR = "Volvo";
  echo MYCAR;
?>
```

Czasami trzeba zmienić zmienną z jednego typu danych na inny, a czasami chcesz, aby zmienna miała określony typ danych. Można to zrobić za pomocą rzutowania. Rzutowanie w PHP wykonuje się za pomocą następujących poleceń:

- (string) - konwertuje na typ danych String,
- (int) - konwertuje na typ danych Integer,
- (float) - konwertuje na typ danych Float,
- (bool) - konwertuje na typ danych Boolean,
- (array) - konwertuje do typu danych Tablica,
- (object) - konwertuje do typu danych Obiekt,
- (unset) - konwertuje na typ danych NULL.

Przykład zastosowania konwertowania typów danych:

```
<!DOCTYPE html>
<html>
<body>
<pre>
<?php
  $a = 5;       // Integer
  $b = 5.34;    // Float
  $c = "hello"; // String
  $d = true;    // Boolean
  $e = NULL;    // NULL

  $a = (string) $a;
  $b = (string) $b;
  $c = (string) $c;
  $d = (string) $d;
  $e = (string) $e;

  //Aby sprawdzić typ dowolnego obiektu w PHP, użyj funkcji var_dump():
  var_dump($a);
  var_dump($b);
  var_dump($c);
  var_dump($d);
  var_dump($e);
?>
</pre>
<p>Należy pamiętać, że podczas rzutowania wartości logicznej na ciąg znaków otrzymuje ona wartość "1", a podczas rzutowania wartości NULL na ciąg znaków jest ona konwertowana na pusty ciąg znaków "".
</p>
<pre>
<?php
  $a = 5;       // Integer
  $b = 5.34;    // Float
  $c = "25 kilometers"; // String
  $d = "kilometers 25"; // String
  $e = "hello"; // String
  $f = true;    // Boolean
  $g = NULL;    // NULL

  $a = (int) $a;
  $b = (int) $b;
  $c = (int) $c;
  $d = (int) $d;
  $e = (int) $e;
  $f = (int) $f;
  $g = (int) $g;

  //Aby sprawdzić typ dowolnego obiektu w PHP, użyj funkcji var_dump():
  var_dump($a);
  var_dump($b);
  var_dump($c);
  var_dump($d);
  var_dump($e);
  var_dump($f);
```

```
  var_dump($g);
?>
</pre>

<p>Należy pamiętać, że podczas rzutowania ciągu rozpoczynającego się liczbą funkcja
(int) używa tej liczby. Jeśli ciąg nie zaczyna się liczbą, funkcja (int) konwertuje
ciągi na liczbę 0.</p>
</body>
</html>
```

Zadanie:

Zapoznaj się z opisem metod konwertowania typów danych w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_casting.asp

W skryptach PHP możemy zastosować funkcje do konwertowania wartości zmiennych pomiędzy różnymi systemami liczbowymi:

- `decbin()` – binarna → dziesiętna,
- `bindec()` – dziesiętna → binarna,
- `dechex()` – dziesiętna → szesnastkowa,
- `hexdec()` – szesnastkowa → dziesiętna,
- `decoct()` – dziesiętna → ósemkowa,
- `octdec()` – ósemkowa → dziesiętna.

Przykład zastosowanie funkcji konwertujących:

```
<?php
  $decimal = 42;
  $binary = decbin($decimal);
  echo "<br>Dziesiętna: $decimal → Binarna: $binary";
  // Wynik: Dziesiętna: 42 → Binarna: 101010
  $binary = "101010";
  $decimal = bindec($binary);
  echo "<br>Binarna: $binary → Dziesiętna: $decimal";
  // Wynik: Binarna: 101010 → Dziesiętna: 42
  // Konwertowanie liczb
  $bin = 0b1010; // binarna liczba 1010 = 10 dziesiętnie
  echo "<br>$bin"; // wynik: 10
  $hex = 0x1A; // szesnastkowa liczba 1A = 26 dziesiętnie
  echo "<br>$hex"; // wynik: 26
  $dec = 26;
  $bin = 0b11010;
  $hex = 0x1A;
  echo ($dec === $bin) ? "<br>Binarnie OK\n" : "Binarnie źle\n";
  echo ($dec === $hex) ? "<br>Szesnastkowo OK\n" : "Szesnastkowo źle\n";
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP, który wygeneruje zamieni przykładowy adres IPv4 z zapisu dziesiętnego na binarny. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_ipv4.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

PHP posiada zbiór funkcji matematycznych, które umożliwiają wykonywanie zadań matematycznych na liczbach. Należą do nich między innymi następujące funkcje:

- `pi()` - zwraca wartość PI,
- `min()` - znajduje najniższą wartość z listy argumentów,
- `max()` - znajduje najwyższą wartość z listy argumentów,
- `abs()` - zwraca wartość bezwzględną (dodatnią) liczby,
- `sqrt()` - zwraca pierwiastek kwadratowy liczby,
- `pow(x,y)` - zwraca x podniesione do potęgi y,
- `round()` - zaokrągla liczbę zmiennoprzecinkową do najbliższej liczby całkowitej,

– `rand()` - generuje liczbę losową.

Zadanie:

Zapoznaj się z opisem funkcji matematycznych dostępnych w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_ref_math.asp

Przykład zastosowania funkcji matematycznych:

```
<?php
  echo (pi()."<br>");
  echo(min(0, 150, 30, 20, -8, -200)."<br>");
  echo(max(0, 150, 30, 20, -8, -200)."<br>");
  echo(abs(-6.7)."<br>");
  echo(sqrt(7)."<br>");
  echo(sqrt(7),2."<br>");
  echo(round(0.50)."<br>");
  echo(round(0.49)."<br>");
  echo(round(sqrt(7))."<br>");
  echo(rand()."<br>");
  echo(rand(1,49)."<br>");
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP, który wygeneruje zestaw sześciu liczb losowych z zakresu od 1 do 49 oraz wyświetli je w tabeli w jednym wierszu. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_lotto.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Typy operatorów w języku PHP.

Operatory to symbole używane do wykonywania operacji na zmiennych i wartościach. PHP posiada wiele rodzajów operatorów:

– `operatory arytmetyczne:`
  + dodawanie,
  - odejmowanie,
  * mnożenie,
  / dzielenie,
  % modulo (reszta z dzielenia),
  ** potęgowanie (od PHP 5.6+),

– `operatory przypisania:`
  = przypisanie,
  += dodanie i przypisanie,
  -= odejmowanie i przypisanie,
  *= mnożenie i przypisanie,
  /= dzielenie i przypisanie,
  %= modulo i przypisanie,
  **= potęgowanie i przypisanie,
  |= suma bitowa i przypisanie,
  & = iloczyn bitowy i przypisanie,
  ^= różnica bitowa i przypisanie,
  <<= przesunięcie bitowe w lewo i przypisanie,
  >>= przesunięcie bitowe w prawo i przypisanie,

– `operatory porównania` (zwracają TRUE lub FALSE)`:`
  == równoważność,
  === identyczność (wartość i typ),
  != lub <> nierówność,
  !== nierówność (różne lub inny typ),
  < mniejsze,
  > większe,

<= mniejsze lub równe,
>= większe lub równe,

- operatory logiczne:
  && AND (koniunkcja),
  || OR (alternatywa),
  ! NOT (negacja),
  XOR (różnica logiczna),
  and, or – alternatywnie, z nieco innymi zasadami priorytetu,
- operatory bitowe (powtórzyć z Wikipedii):
  & AND (iloczyn bitowy, koniunkcja bitowa),
  ```
  0 & 0     0
  0 & 1     0
  1 & 0     0
  1 & 1     1
  ```
  | OR (suma bitowa),
  ```
  0 | 0     0
  0 | 1     1
  1 | 0     1
  1 | 1     1
  ```
  ^ XOR (różnica symetryczna),
  ```
  0 ^ 0     0
  0 ^ 1     1
  1 ^ 0     1
  1 ^ 1     0
  ```
  ~ NOT (negacja),
  ```
  0 ~ 1     1
  1 ~ 0     0
  ```
  << przesunięcie w lewo,
  ```
  10 << 1   100
  ```
  >> przesunięcie w prawo,
  ```
  10 >> 1   01
  ```
- operatory inkrementacji i dekrementacji:
  ++ inkrementacja (np. ++$a preinkrementacja, $a++ postinkrementcja),
  -- dekrementacja, (np. --$a predekrementacja, $a-- postdekrementacja),
- operator łączenia łańcuchów znaków:
  . (kropka) – łączenie tekstów,
- operator warunkowy (ternarny):
  `?: - np. $result = $a ? $b : $c; // (jeżeli prawda to $b, w przeciwnym wypadku $c)`
- operator null coalescing:
  `??` – zwraca wartość pierwszej niepustej zmiennej, np. `$name = $_GET['name'] ?? 'Gość';`

Przykład zastosowanie operatorów arytmetycznych:

```
<?php
  $x=7;
  $y=8;
  $suma=$x+$y;
  $roznica=$x-$y;
  $iloczyn=$x*$y;
  $iloraz=$x/$y;
  $reszta=$x%$y;
  $potega=$x**$y;
  echo "Liczba x = $x <br>";
  echo "Liczba y = $y <br>";
  echo "Suma zmiennych x i y wynosi: $suma<br>";
  echo "Różnica zmiennych x i y wynosi: $roznica<br>";
  echo "Iloczyn zmiennych x i y wynosi: $iloczyn<br>";
  echo "Iloraz zmiennych x i y wynosi: $iloraz<br>";
  echo "Reszta z dzielenia zmiennych x i y wynosi: $reszta<br>";
```

```
  echo "Potęga zmiennej x podniesiona do zmiennej y wynosi: $potega<br>";
?>
```

Przykład zastosowania operatorów bitowych:

```
<?php
  $x=77;
  $y=82;
  $suma_bitowa=$x|$y;
  $roznica_bitowa=$x^$y;
  $iloczyn_bitowy=$x&$y;
  $negacja=~$x;
  echo "Zmienna x = $x oraz zmienna y = $y<br>";
  echo "Suma bitowa zmiennych x i y wynosi: $suma_bitowa<br>";
  echo "Różnica bitowa zmiennych x i y wynosi: $roznica_bitowa<br>";
  echo "Iloczyn bitowy zmiennych x i y wynosi: $iloczyn_bitowy<br>";
  echo "Negacja bitowa zmiennej x wynosi: $negacja<br>";
?>
```

Dla lepszego zrozumienia operacji podanych powyżej należy rozpisać liczby dziesiętne w systemie binarnym.

```
  $x = 77 = 01001101
  $y = 82 = 01010010
  77 & 82 = 01000000 = 64
  77 | 82 = 01011111 = 95
  77 ^ 82 = 00011111 = 31
  ~77     = 10110010 = 178
  77 << 1 = 10011010 = 154
  77 >> 1 = 00100110 = 38
```

Zadanie:

Zmodyfikuj powyższy skrypt tak, aby dodatkowo wyświetlał wszystkie wartości liczbowe binarnie.

Przykład zastosowania operatorów bitowych z danymi wyświetlonymi dodatkowo binarnie:

```
<?php
  $x=77;
  $y=82;
  $suma_bitowa=$x|$y;
  $roznica_bitowa=$x^$y;
  $iloczyn_bitowy=$x&$y;
  $negacja=~$x;
  echo "Zmienna x = $x oraz zmienna y = $y<br>";
  echo "Zmienna x = ".decbin($x)." oraz zmienna y = ".decbin($y)."<br>";
  echo "Suma bitowa zmiennych x i y wynosi: $suma_bitowa<br>";
  echo "Suma bitowa zmiennych x i y wynosi: ".decbin($suma_bitowa)."<br>";
  echo "Różnica bitowa zmiennych x i y wynosi: $roznica_bitowa<br>";
  echo "Różnica bitowa zmiennych x i y wynosi: ".decbin($roznica_bitowa)."<br>";
  echo "Iloczyn bitowy zmiennych x i y wynosi: $iloczyn_bitowy<br>";
  echo "Iloczyn bitowy zmiennych x i y wynosi: ".decbin($iloczyn_bitowy)."<br>";
  echo "Negacja zmiennej x wynosi: $negacja<br>";
  echo "Negacja zmiennej x wynosi: ".decbin($negacja)."<br>";
?>
```

Temat: Instrukcje sterujące if oraz switch w języku PHP.

Instrukcje warunkowe wykonują określone polecenia w zależności od spełnienia warunków logicznych. W języku PHP wyróżniamy instrukcje warunkowe `if` oraz `switch`.

Zadanie:

Zapoznaj się z opisem instrukcji warunkowej `if` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_if_else.asp

Ogólna postać niepełnej instrukcji warunkowej `if`:

```
if(warunek)
{
  instrukcje;
```

```
}
```

Przykład zastosowania niepełnej instrukcji warunkowej `if`:

```
<?php
  $x=17;
  if ($x%2==1)
  {
    echo "Liczba $x jest nieparzysta.";
  }
?>
```

Przykład zastosowania niepełnej instrukcji warunkowej `if` z zapisem skróconym do jednego wiersza:

```
<?php
  $x=12;
  if ($x%2==0) echo " Liczba $x jest parzysta.";
?>
```

Ogólna postać pełnej instrukcji warunkowej `if`:

```
if(warunek)
{
  instrukcje1;
}
else
{
  instrukcje2;
}
```

Przykład zastosowania pełnej instrukcji warunkowej `if`:

```
<?php
  $x=17;
  if($x%2==0)
  {
    echo "Podana liczba $x jest parzysta.";
  }
  else
  {
    echo "Podana liczba $x jest nieparzysta.";
  }
?>
```

Inny przykład zastosowania pełnej instrukcji warunkowej `if` z klauzulą `elseif`:

```
<?php
  $x=127;
  echo "Sprawdzenie, czy podana liczba x jest podzielna przez 35, 5 oraz 7.<br>";
  if($x%35==0)
  {
    echo "Podana liczba $x jest podzielna przez 35.";
  }
  // sprawdzenie, czy x jest podzielne przez 5
  elseif ($x%5==0)
  {
    echo "Podana liczba $x jest podzielna przez 5, ale nie jest podzielna przez
7.";
  }
  // sprawdzenie, czy x jest podzielne przez 7
  elseif ($x%7==0)
  {
    echo "Podana liczba $x jest podzielna przez 7, ale nie jest podzielna przez
5.";
  }
  else
  {
    echo "Podana liczba $x nie jest podzielna przez 5 i nie jest podzielna przez 7,
czyli też nie jest podzielna przez 35.";
  }
?>
```

Inny przykład zastosowania pełnej instrukcji warunkowej `if` z klauzulą `elseif`:

```
<?php
  $t = date("H");
  echo "<p>The hour (of the server) is " . $t;
  echo ", and will give the following message:</p>";
  if ($t < "10")
  {
    echo "Have a good morning!";
  }
  elseif ($t < "20")
  {
    echo "Have a good day!";
  }
  else
  {
    echo "Have a good night!";
  }
?>
```

W powyższym przykładzie zastosowano funkcję `date()`, która służy do formatowania daty i czasu w czytelny sposób. Jest jedną z najczęściej używanych funkcji do pracy z czasem w PHP.

Zadanie:

Zapoznaj się z opisem funkcji `date()` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_date.asp

Składnia funkcji `date()`:

```
date($format, $timestamp = time())
```

gdzie:

`$format` – ciąg znaków określający sposób formatowania daty.

`$timestamp` – (opcjonalny) znacznik czasu UNIX (liczba sekund od 1 stycznia 1970). Jeśli nie podano, używana jest aktualna data i czas.

Przykład użycia funkcji `date()`:

```
<?php
  echo date("Y-m-d H:i:s") . "<br>";
  // Wynik: 2025-09-26 14:41:00 (w zależności od aktualnego czasu)
  echo "Today is " . date("Y.m.d") . "<br>";
  // Wynik: 2025.09.26
  echo "The time is " . date("h:i:sa") . "<br>";
  // Wynik: The time is 08:04:28am
  echo "Today is " . date("l") . "<br>";
  // Wynik: Today is Tuesday
?>
```

Popularne znaki formatujące:

```
Kod   Znaczenie                    Przykład
Y     Rok (cztery cyfry)           2025
y     Rok (dwie cyfry)             25
m     Miesiąc (01–12)              09
d     Dzień miesiąca (01–31)       26
H     Godzina (00–23)              14
i     Minuty (00–59)               41
s     Sekundy (00–59)              00
l     Pełna nazwa dnia tygodnia (L) Friday
D     Skrót dnia tygodnia          Fri
```

Przykład zastosowania funkcji `date()` z pełną instrukcją warunkową `if`:

```
<?php
  $time = date("H");
  echo "<p>The hour (of the server) is " . $time;
  echo ", and will give the following message:</p>";
  if ($time < "12") {
```

```
        echo "Have a good morning!";
    } elseif ($time < "18") {
        echo "Have a good day!";
    } else {
        echo "Have a good night!";
    }
?>
```

Przykład zastosowania pełnej instrukcji warunkowej `if` z warunkami połączonymi za pomocą operatora iloczynu logicznego:

```
<?php
  $x=127;
  echo "Sprawdzenie, czy podana liczba x jest liczbą parzystą i podzielną przez
3.<br>";
  if($x%2==0 && $x%3==0)
  {
    echo "Podana liczba $x jest liczbą parzystą i podzielną przez 3.";
  }
  elseif ($x%2!=0 && $x%3==0)
  {
    echo "Podana liczba $x jest liczbą nieparzystą i podzielną przez 3.";
  }
  elseif ($x%2==0 && $x%3!=0)
  {
    echo "Podana liczba $x jest liczbą parzystą i niepodzielną przez 3.";
  }
  else
  {
    echo "Podana liczba $x jest liczbą nieparzystą i niepodzielną przez 3.";
  }
?>
```

Przykład zagnieżdżonej instrukcji `if`:

```
<?php
  $a = 13;
  if ($a > 10)
  {
    echo "Liczba większa od 10";
    if ($a > 20)
    {
      echo " oraz także większa od 20.";
    }
    else
    {
      echo " ale nie większa od 20.";
    }
  }
?>
```

Zadanie:

Napisz skrypt PHP, który przyjmuje wiek osoby i na podstawie tego wyświetla, czy osoba jest pełnoletnia, czy niepełnoletnia. Osoba jest pełnoletnia, jeśli ma co najmniej 18 lat.

Przykładowe proste rozwiązanie zadania z podanym wiekiem:

```
<?php
  // Zmienna zawierająca wiek osoby
  $wiek = 20;
  // Sprawdzanie, czy osoba jest pełnoletnia
  if ($wiek >= 18)
    {
      echo "Jest pełnoletnia.";
    }
    else
    {
      echo "Jest niepełnoletnia.";
    }
```

```
?>
```

Zadanie:

Napisz prosty program w PHP z formularzem, w którym użytkownik wpisuje swój wiek. Po wysłaniu formularza strona powinna sprawdzić, czy użytkownik jest pełnoletni czy nie.

Przykładowe proste rozwiązanie zadania z zastosowaniem prostego formularza z metodą `GET`:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sprawdź wiek</title>
</head>
<body>
<!-- W metodzie GET w pasku adresu zapisane zostaną przesyłane dane (np. *.php?age=17) -->
<form method="GET">
  Podaj swój wiek:
  <input type="number" name="age">
  <input type="submit" value="Sprawdź">
</form>
<?php
  if ($_SERVER["REQUEST_METHOD"] == "GET")
  {
    if (isset($_GET["age"]))
    {
      $age = $_GET["age"];
      if ($age >= 18)
      {
        echo "Jesteś pełnoletnia.";
      }
      else
      {
        echo "Nie jesteś pełnoletnia.";
      }
    }
  }
?>
</body>
</html>
```

lub z metodą przesyłania `POST` (dla większej liczby danych):

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sprawdź wiek</title>
</head>
<body>
<!-- W metodzie POST dane zostaną przesłane w postaci niejawnej -->
<form method="POST">
  Podaj swój wiek:
  <input type="number" name="age">
  <input type="submit" value="Sprawdź">
</form>
<?php
if (isset($_POST["age"]))
  {
    $age = $_POST["age"];
    if ($age >= 18)
      {
        echo "Jesteś pełnoletni.";
      }
      else
      {
        echo "Nie jesteś pełnoletni.";
      }
```

```
    }
?>
</body>
</html>
```

Zadanie:

Zapoznaj się z opisem typu tablicowego publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/html/html_forms.asp

Atrybuty znacznika `<form>` (https://webkod.pl/kurs-html/tagi/formularz/element-form):

- `name` – nazwa przypisana do formularza,
- `action="https://greszata.pl/wyslij.php"` – adres przekazania danych z formularza,
- `method` – get (domyślnie, przesyłane dane są widoczne poprzez adres strony), post (przesyłane dane są niejawne, dobre rozwiązanie przy większej liczbie danych) – metody przekazania zawartości formularza,
- `onsubmit` –
- `accept-charset="utf-8"` – kodowanie znaków, domyślnie ustawienie dokumentu,
- `autocomplete="on"` – autouzupełnianie wpisywanych wartości,
- `enctype="aplication/urlencoded"` – sposób kodowania danych,
- `novalidate` – wartości formularza nie są wymagane,
- `target="_self" (_blank)` – miejsce wczytania adresu docelowego dla formularza (bieżące/nowe okno).

Atrybuty znacznika `<input>` (https://webkod.pl/kurs-html/tagi/formularz/element-input):

- `name` – nazwa elementu,
- `type` – text, submit, file, number, botton, checkbox, color, date, e-mail, image, password, radio, range, reset,
- `required` – wartość wymagana, gdy będzie pusta zostanie wyświetlony błąd przeglądarki,
- `value` – wartość wyświetlana,
- `max` – maksymalna wartość (liczba, data...),
- `min` – minimalna wartość(liczba, data...),
- `maxlength` – maksymalna liczba znaków,
- `minlength` – minimalna liczba znaków,
- `size="5"` – liczba widocznych znaków w polu,
- `heigth="30"` – wysokość elementu,
- `pattern="[0-9]{4}` – wzorzec elementu wejściowego (cztery znaki od 0 do 9),
- `disabled` – wyłączenie elementu wejściowego,
- `alt` – alternatywny podpis elementu,
- `list="lista"` – rozwijalna lista elementów (`<datalist id="lista"><option>1</option></datalist>`),
- `readonly` – element tylko do odczytu (data, telefon, url...),
- `src` – plik graficzny reprezentujący element,
- `title` – tekst wyświetlany w przypadku błędnego podania danych.

Atrybuty znacznika `<button>` (https://webkod.pl/kurs-html/tagi/formularz/element-button):

- `name` – nazwa elementu,
- `type` – submit, reset, text,
- `autofocus` – czy dany buton ma być traktowany jako element HTML,
- `disabled` – wyłączenie danego przycisku (brak interakcji),
- `formaction` – deklarujemy adres URL, do którego zostaną przekazane dane z formularza,
- `formmethod` – deklaracja sposobu przekazania danych (post, get),
- `formtarget` – deklaracja punktu docelowego do wyświetlenia danych z formularza (_self, _blank, _parent),
- `value="tekst"` – wartość przypisana do elementu button.

Inny przykład zastosowania instrukcji warunkowej `if`:

```
<?php
  $dzialanie = "/";
  $x=21;
  $y=15;
  // Poniżej pominięto nawiasy klamrowe, ponieważ w blokach wykonywane są
pojedyncze instrukcje
  if ($dzialanie=='+')
    echo "Suma liczb $x oraz $y wynosi: ".($x+$y);
  elseif ($dzialanie=='-')
    echo "Różnica liczb $x oraz $y wynosi: ".($x-$y);
  elseif ($dzialanie=='*')
    echo "Iloczyn liczb $x oraz $y wynosi: ".($x*$y);
  elseif ($dzialanie=='/')
    if ($y!=0)
      echo "Iloraz liczb $x przez $y wynosi: ".($x/$y);
    else
      echo "Nie można dzielić przez 0.";
  elseif ($dzialanie=='%')
    echo "Reszta z dzielenia liczb $x przez $y wynosi: ".($x%$y);
  else
    echo "Nie wybrałeś żadnego działania!";
?>
```

Wykonaj zadanie:

Zmodyfikuj skrypt z powyższego przykładu z wykorzystaniem instrukcji warunkowej `if` tak, aby poprawić jego estetykę i działanie (rodzaj działania pobrany od użytkownika). Dodaj również stosowne komunikaty. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_kalkulator.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Operator warunkowy działa podobnie do instrukcji `if`. Ogólna postać operatora warunkowego:

```
warunek ? "wyrażenie, gdy prawda" : "wyrażenie, gdy nieprawda";
```

Przykład zastosowania operatora warunkowego:

```
<?php
  $a = 13;
  $b = $a < 18 ? "Jesteś niepełnoletni." : "Jesteś pełnoletni.";
  echo $b;
?>
```

Inny przykład zastosowania operatora warunkowego:

```
<?php
  echo "Skrypt sprawdzający, czy zmienna a jest liczbą z przedziału <10,99>.<br>";
  $a = 13;
  $warunek = ($a>=10 && $a<=99) ? "Liczba dwucyfrowa." : "Liczba nie jest
dwucyfrowa";
  echo $warunek;
?>
```

Wykonaj zadanie:

Napisz skrypt z wykorzystaniem operatora warunkowego w języku PHP sprawdzający, która z dwóch liczb podanych przez użytkownika jest większa. W skrypcie zastosuj treści informujące o temacie zadania oraz komunikat wynikowy wyświetlony na stronie tekstem sformatowanym jako nagłówek H4 w kolorze niebieskim. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_wieksza.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP sprawdzający znak podanej przez użytkownika liczby (dodatnia, ujemna czy zero). W skrypcie zastosuj treści informujące o temacie zadania oraz komunikat wynikowy wyświetlony na stronie tekstem sformatowanym jako nagłówek H2 w kolorze zielonym. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_liczba.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystaniem instrukcji warunkowej `if`, który sprawdzi, czy z podanych przez użytkownika trzech długości odcinków można zbudować trójkąt. Komunikat wynikowy ma być wyświetlony na stronie tekstem sformatowanym jako nagłówek H3 w kolorze czerwonym dla wyniku negatywnego, a w kolorze zielonym dla wyniku pozytywnego. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_trojkat.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystaniem instrukcji warunkowej `if` z klauzulą `elseif`, który sprawdzi podaną przez użytkownika ocenę w postaci liczby całkowitej i wyświetli dla poszczególnych ocen następujące komunikaty: 1 – niedostateczna praca, 2 – dopuszczająca praca, 3 – dostateczna praca, 4 – dobra praca, 5 - bardzo dobra praca, 6 – celująca praca. W skrypcie zastosuj treści informujące o temacie zadania oraz komunikat wynikowy wyświetlony na stronie tekstem sformatowanym jako nagłówek H4 w kolorze czerwonym dla ocen 1-2, zielonym dla ocen 3-5 oraz niebieskim dla oceny 6. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_ocena.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Przykładowe rozwiązanie do powyższego zadania z użyciem instrukcji warunkowej `if` oraz tablicy asocjacyjnej:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sprawdź ocenę</title>
</head>
<body>
<form method="GET">
  Podaj swoją ocenę:
  <input type="number" name="ocena" required>
  <input type="submit" value="Sprawdź">
</form>
<?php
  if (isset($_GET["ocena"]) && is_numeric($_GET["ocena"]))
  {
    // Zmienna zawierająca ocenę
    $ocena = (int) $_GET["ocena"];
    $komentarze = [
      1 => "Bardzo słaba praca",
      2 => "Słaba praca",
      3 => "Dostateczna",
      4 => "Dobra",
      5 => "Bardzo dobra",
      6 => "Celująca"
    ];
    if (isset($komentarze[$ocena]))
      {
      echo $komentarze[$ocena];
    }
      else
      {
      echo "Niepoprawna ocena";
    }
  }
?>
</body>
</html>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP, który zamieni podany przez użytkownika adres IPv4 z zapisu dziesiętnego na binarny. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Sprawdź poprawność wprowadzonego adresu (liczby z zakresu 0-255). Pracę zapisz w pliku pod nazwą

$nazwisko_$klasa_$gr_ipv4.php i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Przydatne funkcje do rozwiązania powyższego zadania:

- `explode(separator,string,limit)` – funkcja dzieli ciąg znaków na tablicę, parametr limit (opcjonalny) określa liczbę elementów tablicy do zwrócenia,
- `str_pad(string,length,pad_string,pad_type)` – funkcja uzupełnia ciąg znaków do podanej długości, `length` - długość wypełnienia (8), `pad_string` - znak wypełnienia ('0'), `pad_type` - strona wypełnienia (`STR_PAD_BOTH, STR_PAD_LEFT, STR_PAD_RIGHT`),
- `count(array)` – funkcja zwraca liczbę elementów w tablicy,
- `is_numeric(variable)` – funkcja zwraca wartość `true (1)`, jeśli zmienna jest liczbą, w przeciwnym wypadku zwraca `false/nothing`.

Zadanie:

Zapoznaj się z opisem instrukcji `switch` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_switch.asp

Instrukcja `switch` jest instrukcją wyboru. Zastosowane w instrukcji wyrażenie jest porównywane z listą wartości, gdzie przy znalezieniu wartości pasującej program wykona zapisane dla tej wartości instrukcje. W przypadku braku spełnienia któregokolwiek z warunków, wykonane zostaną instrukcje z bloku `default`. Ogólna postać instrukcji `switch`:

```
switch (wyrażenie)
{
  case wart1:
    instrukcje1;
    break;  //przerwanie instrukcji switch i wyjście z bloku
  case wart2:
    instrukcje2;
    break;  //jeżeli zostanie wykonane, to kończy instrukcję switch
  case wartn:
    instrukcjen;
    break;
  default:  //opcjonalnie
    instrukcje_wykonane_przy_braku_dopasowania;
}
```

Przykład zastosowania instrukcji warunkowej wielokrotnego sprawdzania `switch`:

```
<?php
  $favcolor = "red";
  switch ($favcolor)
  {
    case "red":
      echo "Your favorite color is red!";
      break;
    case "blue":
      echo "Your favorite color is blue!";
      break;
    case "green":
      echo "Your favorite color is green!";
      break;
    default:
      echo "Your favorite color is neither red, blue, nor green!";
  }
?>
```

Inny przykład zastosowania instrukcji warunkowej wielokrotnego sprawdzania `switch`:

```
<?php
  $dzialanie = "divide";
  $liczba1=5;
  $liczba2=7;
  switch ($dzialanie)
      {
```

```
            case 'add':
              $wynik = $liczba1+$liczba2;
              break;
            case 'subtract':
              $wynik = $liczba1-$liczba2;
              break;
            case 'multiply':
              $wynik = $liczba1*$liczba2;
              break;
            case 'divide':
              if ($liczba2 !== 0)
              {
                $wynik = round(($liczba1/$liczba2)*100)/100;
              } else
              {
                $wynik = 'Nie można dzielić przez zero';
              }
              break;
            default:
              $wynik = 'Nieznana operacja';
          }
          echo $wynik;
    ?>
```

Inny przykład zastosowania instrukcji warunkowej wielokrotnego sprawdzania `switch`:

```
    <?php
      $dzialanie = "/";
      $x=21;
      $y=15;
      switch ($dzialanie)
      {
        case "+":
          echo "Suma liczb $x oraz $y wynosi: ".($x+$y);
          break;
        case "-":
          echo "Różnica liczb $x oraz $y wynosi: ".($x-$y);
          break;
        case "*":
          echo "Iloczyn liczb $x oraz $y wynosi: ".($x*$y);
          break;
        case "/":
          echo "Iloraz liczb $x przez $y wynosi: ".($x/$y);
          break;
        case "%":
          echo "Reszta z dzielenia liczb $x przez $y wynosi: ".($x%$y);
          break;
        default:
          echo "Nie wybrałeś żadnego działania!";
      }
    ?>
```

Wykonaj zadanie:

Zmodyfikuj skrypt z powyższego przykładu z wykorzystaniem instrukcji `switch` tak, aby poprawić jego estetykę i działanie (dwie liczby oraz rodzaj działania pobrany od użytkownika). Dodaj do działań możliwość wyboru potęgowania oraz pierwiastka kwadratowego z pierwszej liczby. Dla wyboru działania w formularzu zastosuj znacznik `<select>`. Dodaj również stosowne komunikaty na stronie. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_kalkulator.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystaniem instrukcji wyboru `switch`, który sprawdzi podaną przez użytkownika ocenę w postaci liczby całkowitej i wyświetli odpowiedni komentarz w zależności od otrzymanej oceny. Ocenie przyporządkujemy następujące komentarze:

1: "Bardzo słaba praca",

2: "Słaba praca",

3: "Dostateczna",
4: "Dobra",
5: "Bardzo dobra",
6: "Celująca",
Jeśli ocena jest spoza zakresu 1-6, wyświetl informację "Niepoprawna ocena".

W skrypcie zastosuj treści informujące o temacie zadania oraz komunikat wynikowy wyświetlony na stronie tekstem sformatowanym jako nagłówek H3 w kolorze czerwonym dla ocen 1-2, zielonym dla ocen 3-5 oraz niebieskim dla oceny 6. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_ocena.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sprawdź ocenę</title>
</head>
<body>
<form method="GET">
  Podaj swoją ocenę:
  <!--<input type="number" name="ocena" min="1" max="6" required>-->
  <input type="number" name="ocena" required>
  <input type="submit" value="Sprawdź">
</form>
<?php
  if (isset($_GET["ocena"]) && is_numeric($_GET["ocena"]))
  {
    // Zmienna zawierająca ocenę
    $ocena = (int) $_GET["ocena"];
    // Użycie switch do przypisania komentarza na podstawie oceny
    switch ($ocena)
      {
      case 1:
        echo "Bardzo słaba praca";
        break;
      case 2:
        echo "Słaba praca";
        break;
      case 3:
        echo "Dostateczna";
        break;
      case 4:
        echo "Dobra";
        break;
      case 5:
        echo "Bardzo dobra";
        break;
      case 6:
        echo "Celująca";
        break;
      default:
        echo "Niepoprawna ocena";
    }
  }
?>
</body>
</html>
```

Inny przykład zastosowania instrukcji warunkowej wielokrotnego sprawdzania `switch`:

```
<?php
  $d = date("l");;
  switch ($d)
  {
    case "Sunday":
      $dzien = "niedziela";
```

```
        break;
      case "Monday":
        $dzien = "poniedziałek";
        break;
      case "Tuesday":
        $dzien = "wtorek";
        break;
      case "Wednesday":
        $dzien = "środa";
        break;
      case "Thursday":
        $dzien = "czwartek";
        break;
      case "Friday":
        $dzien = "piątek";
        break;
      case  "Saturday":
        $dzien = "sobota";
    }
    echo    "<h3>Skrypt    sprawdza    bieżący    dzień    tygodnia.<br>Dzisiaj    jest:
".$dzien."</h3>";
?>
```

Temat: Pętla for w języku PHP.

Zadanie:

Zapoznaj się z opisem pętli `for` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_looping_for.asp

Pętla `for` powtarza zestaw instrukcji określoną przez licznik ilość razy. Ogólna postać instrukcji `for`:

```
for (start; koniec; krok;)
{
  instrukcje;
}
```

Przykład zastosowania pętli `for`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli.<br>";
  for ($i = 0; $i <= 50; $i+=10)
  {
    echo "Stan licznika pętli: $i <br>";
  }
?>
```

Inny przykład zastosowania pętli `for`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli z wymuszonym przerwaniem.<br>";
  for ($i = 0; $i <= 5; $i++)
  {
    if ($i == 3) break;
    echo "Stan licznika pętli: $i <br>";
  }
?>
```

Inny przykład zastosowania pętli `for`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli bez jednej wartości.<br>";
  for ($i = 5; $i <= 0; $i--)
  {
    if ($i == 3) continue;
    echo "Stan licznika pętli: $i <br>";
  }
?>
```

Inny przykład zastosowania pętli `for`:

```
<?php
  echo "Sumowanie liczb z listy:\n";
  $lista = [1, 2, 3, 4, 5];
  $suma = 0;
  // count($lista) daje długość tablicy
  for ($i = 0; $i < count($lista); $i++) {
    // $lista[$i] dostęp do elementu
    $suma += $lista[$i];
  }
  echo "<br>Suma liczb zapisanych w tablicy wynosi: $suma \n";
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystaniem pętli `for`, który wyświetli liczby nieparzyste z przedziału liczb podanych przez użytkownika (min(-100), max(100)) oraz ich sumę. Pamiętaj, aby skrypt poprawnie wyświetlał liczby, gdy zakres jest odwrócony (malejący) oraz gdy użytkownik poda te same liczby początku i końca zakresu, oraz gdy wprowadzone dane nie są liczbami. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_parzyste.html** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Pomocne do rozwiązania powyższego przykładu:

```
<?php
$a = -33;
$b = 19;
for ($x = $a; $x <= $b; $x++) {
  // if (abs($x%2) == 1) continue;
  if ($x%2 == 1 or $x%2 == -1) continue;
  $suma += $x;
  echo $x."<br>";
}
  echo "wynik to: $suma <br>";
?>
```

Inny przykład zastosowania pętli `for`:

```
<?php
  echo "Skrypt sumujący 5 losowych liczb całkowitych z zakresu od 1 do 10.";
  $suma = 0;
  echo "<br>Zestaw wylosowanych liczb: ";
  for ($i=1; $i<=5; $i++)
  {
    $tab[$i]=rand(1,10);
    echo $tab[$i]."\t";
    $suma+=$tab[$i];
  }
  echo "<br>Suma wylosowanych liczb wynosi: $suma";
?>
```

Zadanie:

Napisz skrypt PHP, który wypisze tabliczkę mnożenia dla liczby podanej przez użytkownika. Skrypt ma wyświetlić wyniki mnożenia tej liczby przez liczby od 1 do 10.

Przykładowe rozwiązanie do powyższego zadania z określoną liczbą:

```
<?php
  echo "Skrypt wyświetlający wynik mnożenia określonej liczby przez liczby 1 do
10.<br>";
  // Można pobrać liczbę od użytkownika (np. z formularza)
  $liczba = 7; // Zmienna z przykładową liczbą
  echo "Wybrana liczba to: $liczba<br>";
  echo "Tabliczka mnożenia wygląda następująco:<br>";
  // Pętla for do wypisania tabliczki mnożenia
  for ($i = 1; $i <= 10; $i++)
  {
    // Obliczamy wynik mnożenia
    $wynik = $liczba * $i;
    // Wyświetlamy wynik mnożenia
```

```
    echo $liczba . " x " . $i . " = " . $wynik . "<br>";
  }
?>
```

Przykładowe rozwiązanie do powyższego zadania z formularzem, który pozwoli użytkownikowi wpisać liczbę:

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Tabliczka mnożenia</title>
</head>
<body>
<h2>Podaj liczbę do tabliczki mnożenia</h2>
<form method="POST">
  <label for="liczba">Liczba:</label>
  <input type="number" name="liczba" id="liczba" required>
  <input type="submit" value="Pokaż tabliczkę">
</form>
<?php
  // Sprawdzenie, czy formularz został wysłany
  if (isset($_POST["liczba"])) {
    // Pobranie liczby z formularza
    $liczba = (int)$_POST["liczba"];
    echo "<h3>Tabliczka mnożenia dla liczby $liczba:</h3>";
    // Pętla for do wypisania tabliczki mnożenia
    for ($i = 1; $i <= 10; $i++)
    {
      $wynik = $liczba * $i;
      echo "$liczba × $i = $wynik<br>";
    }
  }
?>
</body>
</html>
```

Zadanie:

Napisz skrypt PHP, który wyświetli sumę liczb od 1 do 100, ale tylko tych, które są liczbami podzielnymi przez 3 lub 5.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
  // Zmienna do przechowywania sumy
  $suma = 0;
  // Pętla for, która iteruje od 1 do 100
  for ($i = 1; $i <= 100; $i++)
  {
    // Sprawdzamy, czy liczba jest podzielna przez 3 lub 5
    if ($i % 3 == 0 || $i % 5 == 0)
    {
      // Dodajemy liczbę do sumy
      $suma += $i;
    }
  }
  // Wyświetlamy wynik
  echo "Suma liczb podzielnych przez 3 lub 5 w zakresie od 1 do 100 wynosi: " .
$suma;
?>
```

Przykład zastosowania zagnieżdżonej pętli `for`:

```
<?php
  echo "Skrypt wyświetlający liczby losowe z przedziału od 1 do 49.";
  echo '<table style="width:400px; height:200px;">';
  for ($i=1; $i<=5; $i++)
  {
    echo "<tr>";
    echo "<td width=100px>Zestaw ".$i."</td>";
    for ($j=1;$j<=6;$j++)
```

```
        {
          echo "<td width=50px>".(rand(1,49))."</td>";
        }
        echo "</tr>";
      }
      echo "</table>";
    ?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystaniem pętli for, który wyświetli tabliczkę mnożenia dla liczb od 1 do 10 w tabeli z obramowaniem w kolorze niebieskim. W skrypcie zastosuj i wyświetl treści informujące o temacie zadania. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_tabliczka.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Zadanie:

Napisz skrypt PHP, który przyjmuje z formularza dwie liczby: rozmiar wierszy i kolumn, a następnie wyświetli dynamicznie tabelę, w której każda komórka zawiera numer porządkowy (rosnąco od 1).

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Dynamiczna tabela</title>
  <style>
    table {
      border-collapse: collapse;
    }
    td {
      border: 1px solid black;
      padding: 8px;
      text-align: center;
      width: 50px;
    }
  </style>
</head>
<body>
<h2>Generowanie dynamicznej tabeli</h2>
<form method="POST">
  <label for="wiersze">Liczba wierszy:</label>
  <input type="number" name="wiersze" id="wiersze" required min="1">
  <br><br>
  <label for="kolumny">Liczba kolumn:</label>
  <input type="number" name="kolumny" id="kolumny" required min="1">
  <br><br>
  <input type="submit" value="Generuj tabelę">
</form>
<?php
  if (isset($_POST["wiersze"]) $$ isset($_POST["kolumny"]))
  {
    // Pobranie danych z formularza
    $wiersze = (int)$_POST["wiersze"];
    $kolumny = (int)$_POST["kolumny"];
    // Walidacja danych
    if ($wiersze < 1 || $kolumny < 1)
    {
      echo "<p style='color:red;'>Wiersze i kolumny muszą być większe niż 0.</p>";
    }
    else
    {
      echo "<h3>Tabela $wiersze × $kolumny:</h3>";
      echo "<table>";
      $licznik = 1; // Zmienna do numerowania komórek
      // Główna pętla do tworzenia tabeli
      for ($i = 1; $i <= $wiersze; $i++)
      {
```

```
            echo "<tr>";
            for ($j = 1; $j <= $kolumny; $j++)
            {
              echo "<td>$licznik</td>";
              $licznik++;
            }
              echo "</tr>";
          }
          echo "</table>";
        }
      }
    ?>
    </body>
    </html>
```

Zadanie:

Zmodyfikuj powyższy skrypt PHP, np. tak, aby pokolorować parzyste komórki na jasnozielono oraz podświetlić cały wiersz na szaro po najechaniu myszką (hover).

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Dynamiczna Tabela</title>
    <style>
        table {
            border-collapse: collapse;
            margin-top: 20px;
        }
        td {
            border: 1px solid #333;
            padding: 10px;
            text-align: center;
        }
        .even {
            background-color: #ccffcc; /* jasnozielony */
        }
        tr:hover {
            background-color: #e0e0e0; /* szary */
        }
    </style>
</head>
<body>
    <form method="post">
        <label for="rows">Liczba wierszy:</label>
        <input type="number" name="rows" id="rows" required min="1">
        <br>
        <label for="cols">Liczba kolumn:</label>
        <input type="number" name="cols" id="cols" required min="1">
        <br><br>
        <input type="submit" value="Generuj tabelę">
    </form>
    <?php
      if (isset($_POST["rows"]) && isset($_POST["cols"]))
        {
        $rows = intval($_POST["rows"]);
        $cols = intval($_POST["cols"]);
        echo "<table>";
        $counter = 1;
        for ($i = 0; $i < $rows; $i++) {
            echo "<tr>";
            for ($j = 0; $j < $cols; $j++) {
                $class = ($counter % 2 == 0) ? "even" : "";
                echo "<td class='$class'>$counter</td>";
                $counter++;
            }
            echo "</tr>";
```

```
            }
            echo "</table>";
        }
        ?>
    </body>
    </html>
```

Zadanie:

Napisz skrypt w języku PHP z wykorzystaniem instrukcji `for`, który wygeneruje 5 zestawów liczb po 6 unikalnych liczb losowych z zakresu 1–49, bez użycia funkcji.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
  echo "<h2>Wylosowane zestawy liczb:</h2>";
  for ($i = 1; $i <= 5; $i++) {
    $liczby = range(1, 49);     // Tworzy tablicę liczb od 1 do 49 (uporządkowaną)
    shuffle($liczby);           // Tasuje tablicę
    // array_slice() zwraca elementy tablicy od wskazanego i tyle ile podano
    $zestaw = array_slice($liczby, 0, 6); // Pobiera pierwsze 6 liczb (wycinek)
    sort($zestaw);              // Sortuje liczby rosnąco
    echo "Zestaw $i: " . implode(", ", $zestaw) . "<br>"; // Opcja dodatkowa
  }
?>
```

Wyjaśnienie do powyższego skryptu:

- w skrypcie użyte są funkcje `range()` i `shuffle()` do losowania liczb,
- funkcja `array_slice()` wybiera 6 pierwszych liczb z tabeli po tasowaniu,
- funkcja `sort()` porządkuje liczby rosnąco w zestawie,
- funkcja `implode()` łączy liczby z tablicy w ciąg tekstowy.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie funkcji `for()`, który wygeneruje 5 zestawów po 6 unikalnych liczb losowych z zakresu 1–49 i wyświetla je w formie tabeli HTML.

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Losowe zestawy liczb</title>
    <style>
        table {
            border-collapse: collapse;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #333;
            padding: 10px;
            text-align: center;
        }
        th {
            background-color: #eee;
        }
    </style>
</head>
<body>
    <h2 style="text-align:center;">5 zestawów losowych liczb (1-49)</h2>
    <table>
        <tr>
            <th>Zestaw</th>
            <th>Liczby</th>
        </tr>
        <?php
        for ($i = 1; $i <= 5; $i++) {
```

```
                $allNumbers = range(1, 49);       // Tworzy tablicę liczb od 1 do 49
                shuffle($allNumbers);             // Miesza tablicę
                $selected = array_slice($allNumbers, 0, 6); // Pobiera pierwsze 6 liczb
                sort($selected);                  // Opcjonalnie: sortuje rosnąco
                echo "<tr><td>$i</td><td>" . implode(", ", $selected) . "</td></tr>";
            }
            ?>
        </table>
    </body>
    </html>
```

Wykonaj zadanie:

Zmodyfikuj powyższy skrypt w taki sposób, aby wszystkie wygenerowane liczby wyświetlane były w osobnych komórkach tabeli oraz żeby tabela miała obramowanie w kolorze zielonym. W skrypcie zastosuj i wyświetl treści informujące o temacie zadania. Dla lepszego odczytu w pierwszej kolumnnie numer zestawu poprzedź słowem "Zestaw". Zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_lotto_for.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Przykład zastosowanie pętli for do wylistowania elementów tablicy:

```
<?php
  echo "Obliczanie średniej ocen<br>";
  $oceny = [5, 4, 3, 5, 2];
  echo "Przykładowa lista ocen:<br>";
  for ($i = 0; $i < count($oceny); $i++) {
    echo $oceny[$i]." ";
  }
  // funkcja array_sum()sumuje elementy z tablicy
  // funkcja count()zlicza elementy z tablicy
  echo "<br>Średnia ocen w tablicy wynosi: " . array_sum($oceny)/count($oceny);
?>
```

Zadanie:

Napisz skrypt w języku PHP, który wygeneruje trójkąt Pascala o zadanej liczbie wierszy (np. 10). Każdy wiersz powinien być wyświetlony jako linia HTML, a liczby w trójkącie powinny być wyrównane do środka, aby przypominały kształt trójkąta.

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Trójkąt Pascala</title>
    <style>
        .row {
            text-align: center;
            font-family: monospace;
            margin: 5px 0;
        }
    </style>
</head>
<body>
    <?php
    $rows = 10; // Możesz zmienić na dowolną liczbę

    for ($i = 0; $i < $rows; $i++) {
        echo "<div class='row'>";
        // Dodaj odstępy dla wyrównania
        for ($space = 0; $space < $rows - $i; $space++) {
            echo "  ";
        }

        $number = 1;
        for ($j = 0; $j <= $i; $j++) {
            echo $number . "  ";      //   - oznaczenie spacji
```

```
                $number = $number * ($i - $j) / ($j + 1);
            }
            echo "</div>";
        }
        ?>
    </body>
    </html>
```

Wykonaj zadanie:

Zmodyfikuj powyższy skrypt w języku PHP w taki sposób, aby dodatkowo:

- pobierał od użytkownika liczbę wierszy trójkąta Pascala,
- liczby parzyste były wyświetlane na jasnoniebieskim tle,
- liczby pierwsze były wyświetlane na jasnoczerwonym tle,
- pozostałe liczby miały domyślne tło.

W skrypcie zastosuj pętle for oraz dodaj treści informujące o temacie zadania. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_pascal.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `for()`, który pobierze od użytkownika zakres liczb naturalnych od min 2 do max 100 i wyświetli wszystkie liczby pierwsze z tego zakresu.

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Liczby pierwsze w zakresie</title>
</head>
<body>
<h2>Podaj zakres liczb, aby znaleźć liczby pierwsze</h2>
<form method="post">
    <label for="poczatek">Początkowa liczba:</label>
    <input type="number" name="poczatek" id="start" min="2" required>
    <label for="koniec">Końcowa liczba:</label>
    <input type="number" name="koniec" id="stop" min="2" required>
    <br>
    <input type="submit" value="Pokaż liczby pierwsze">
</form>
<?php
//if (isset($_POST["poczatek"]) && isset($_POST["koniec"])) {
if   ($_SERVER["REQUEST_METHOD"]   ==   "POST"   &&   isset($_POST["poczatek"])   &&
is_numeric($_POST["poczatek"])         &&         isset($_POST["koniec"])         &&
is_numeric($_POST["koniec"])) {
  $poczatek = $_POST["poczatek"];
  $koniec = $_POST["koniec"];
  for ($i = $poczatek; $i <= $koniec; $i++) {
    $pierwsza = true;
    for ($j = 2; $j <= sqrt($i); $j++) {
        if ($i % $j == 0) {
            $pierwsza = false;
            break;
        }
      }
    if ($pierwsza) {
        echo "$i jest liczbą pierwszą<br>";
    }
  }
}
?>
</body>
</html>
```

Wykonaj zadanie:

Zmodyfikuj powyższy skrypt tak, aby poprawnie działał w przypadku, gdy użytkownik poda pierwszą liczbę większą od drugiej (odwrócony zakres).

Temat: Pętla while w języku PHP.

Zadanie:

Zapoznaj się z opisem pętli `while` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_looping_while.asp

Pętla `while` zawiera instrukcje, które będą wykonane, gdy zostanie spełniony warunek pętli (wartość `true`).
Ogólna składnia pętli `while`:

```
while (warunek)         //nagłówek pętli
{                       //ciało pętli
  instrukcje;
}
```

Przykład zastosowania pętli `while`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli.<br>";
  while ($i < 6)
  {
    echo $i;
    $i++;
  }
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli z wymuszonym
zatrzymaniem.<br>";
  $i=0;
  while ($i < 6)
  {
    $i++;
    if ($i==4) break;
    echo $i;
  }
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli bez jednej wartości.<br>";
  $i=1;
  while ($i < 6)
  {
    $i++;
    if ($i==4) continue;
    echo $i;
  }
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo "Sumowanie liczb z listy:\n";
  $lista = [1, 2, 3, 4, 5];
  $suma = 0;
  // sprawdzenie, czy tablica nie jest pusta
  while (!empty($lista)) {
    $suma += $lista[0];
    array_shift($lista); // usuwa pierwszy element z listy
  }
  echo "<br>Suma liczb zapisanych w tablicy wynosi: $suma \n";
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo "Skrypt wyświetlający liczby parzyste z zakresu od -10 do 10.<br>";
  $i=-10;
  while ($i <= 10)
  {
    if ($i % 2 == 0)
      echo $i."\t";
    $i++;
  }
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo "Skrypt wyświetlający liczby nieparzyste z zakresu od -10 do 10.<br>";
  $i=10;
  while ($i >= -10)
  {
    if ($i % 2 != 0)
      echo $i."\t";
    $i--;
  }
?>
```

Inny przykład zastosowania pętli `while` z odwróconym zakresem:

```
<?php
  echo "Skrypt wyświetlający liczby nieparzyste z odwróconego zakresu.<br>";
  $poczatek = 16;
  $koniec = -6;
  if ($poczatek > $koniec) {
    $temp = $poczatek;
    $poczatek = $koniec;
    $koniec = $temp;
  }
  while ($poczatek <= $koniec) {
    if ($poczatek % 2 != 0 ) echo $poczatek . ", ";
    $poczatek++;
  }
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  $x=1357;
  echo "Skrypt wyświetlający liczbę $x w odwróconej kolejności.<br>";
  while ($x > 0)
  {
    // reszta z dzielenia przez 10 wyświetli ostatnią cyfrę
    echo $x%10;
    // dzielimy liczbę przez 10 i zaokrąglamy do liczby całkowitej
    $x=floor($x/10);
  }
  echo "<br>Przykład odwrócenia kolejności cyfr w liczbie
zmiennoprzecinkowej.<br>";
  $a = 23.123;
  $b = (string)$a;
  $c = strrev((string)$a);
  echo $c . "<br>";
  //die;
  echo strrev((string)13.57);
?>
```

Inny przykład zastosowania pętli `while`:

```
<?php
  echo '<h2 style="text-align:center;">
    Skrypt wyświetlający liczby losowe z przedziału od 1 do 49.<h2>';
  echo "<table align='center' border='1'><tr>";
  $i=1;
  while ($i<=6)
  {
    $a=rand(1,49);
    echo "<td width=50px>".$a."</td>";
    $a++;
    $i++;
  }
  echo "</tr></table>";
?>
```

Wykonaj zadanie:

Zmodyfikuj skrypt z powyższego przykładu tak, aby wyświetlał tyle zestawów liczb losowych (np. do 100) oraz z takiego zakresu liczb (np. od 1 do 100) oraz tyle liczb w zestawie (np. do 10), jakie poda użytkownik. Jako pętle wykorzystaj tylko **`while`**. Skrypt powinien sprawdzać poprawność wprowadzonych przez użytkownika danych. Liczby z każdego zestawu mają być posortowane oraz wyświetlane w kolejnych wierszach tabeli HTML. Dla lepszego odczytu dodaj na początku tabeli kolumnę z opisem numeru zestawu (np. "Zestaw $x"). Zadbaj również o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_lotto_while.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Inny przykład zastosowania pętli `while`:

```
<?php
  $a=15;
  $b=78;
  echo "Skrypt wyznaczający największy wspólny dzielnik dwóch liczb: $a i $b.<br>";
  while ($a != $b)
  {
    if ($a < $b)
    {
      // zamiana liczb miejscami, aby różnica była dodatnia
      $zm=$a;
      $a = $b;
      $b = $zm;
    }
    $a = $a - $b;
  }
  echo "Największy wspólny dzielnik podanych liczb wynosi: $a";
?>
```

Wykonaj zadanie:

Utwórz skrypt, który z liczb całkowitych z zakresu od 1 do 100 wyświetli te, które są nieparzyste i podzielne przez 3. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_dzielniki_3.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie funkcji `for()` oraz `while()`, który wygeneruje 5 zestawów po 6 unikalnych liczb losowych z zakresu 1–49 i wyświetla je w formie tabeli HTML.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
echo "<h2>Wylosowane zestawy liczb:</h2>";
echo "<table border='1' cellpadding='8' cellspacing='0'>";
echo "<tr><th>Zestaw</th><th>Liczby</th></tr>";
for ($i = 1; $i <= 5; $i++) {
    $zestaw = [];
```

```
        while (count($zestaw) < 6) {    //warunkiem jest długść tablicy (6 różnych
    liczb)
            $liczba = rand(1, 49);
            if (!in_array($liczba, $zestaw)) {
                $zestaw[] = $liczba;
            }
        }
        sort($zestaw); // sortowanie rosnąco
        $liczbyCiag = implode(", ", $zestaw);  //łączy elementy tablicy z separatorem
        echo "<tr><td>Zestaw $i</td><td>$liczbyCiag</td></tr>";
    }
    echo "</table>";
    ?>
```

Wyjaśnienie do powyższego skryptu:

- skrypt używa funkcji rand(1, 49) do losowania pojedynczych liczb podanego zakrewsu,
- następnie sprawdza za pomocą funkcji in_array(), czy liczba już występuje w zestawie,
- gdy zestaw ma 6 unikalnych liczb, skrypt sortuje go i wyświetla w tabeli.

Przykładowe rozwiązanie do powyższego zadania z wynikami podanymi w tabeli HTML:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Losowe zestawy liczb</title>
    <style>
        table {
            border-collapse: collapse;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid #333;
            padding: 10px;
            text-align: center;
        }
        th {
            background-color: #eee;
        }
    </style>
</head>
<body>
    <h2 style="text-align:center;">5 zestawów losowych liczb (1–49)</h2>
    <table>
        <tr>
            <th>Zestaw</th>
            <th>Liczby</th>
        </tr>
        <?php
        for ($i = 1; $i <= 5; $i++) {
            $numbers = [];
            while (count($numbers) < 6) {
                $num = rand(1, 49);
                if (!in_array($num, $numbers)) {
                    $numbers[] = $num;
                }
            }
            sort($numbers); // opcjonalnie: sortowanie rosnąco
            echo "<tr><td>$i</td><td>" . implode(", ", $numbers) . "</td></tr>";
        }
        ?>
    </table>
</body>
</html>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie z pośród pętli tylko funkcji `while()`, który wygeneruje 5 zestawów po 6 unikalnych liczb losowych z zakresu 1–49 i wyświetla je w formie tabeli HTML.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
      echo  '<h2  style="text-align:center;">Skrypt  wyświetlający  liczby  losowe  z
przedziału od 1 do 49.<br>';
      echo "Wylosowane zestawy liczb:</h2>";
      echo "<table border='1' cellpadding='8' cellspacing='0'>";
      echo "<tr><th width=35%>Zestaw</th><th>Liczby</th></tr>";
      $i = 1;
      while ($i <= 5) {
            $zestaw = [];
            while (count($zestaw) < 6) {
                  $liczba = rand(1, 49);
                  if (!in_array($liczba, $zestaw)) {
                        $zestaw[] = $liczba;
                  }
            }
            sort($zestaw); // sortowanie rosnąco
            $liczbyTekst = implode(", ", $zestaw);
            echo "<tr><td>Zestaw $i</td><td>$liczbyTekst</td></tr>";
            $i++;
      }
      echo "</table>";
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `while()`, który pobierze od użytkownika zakres liczb naturalnych od min 2 do max 100 i wyświetli wszystkie liczby pierwsze z tego zakresu.

Przykładowe rozwiązanie do powyższego zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Liczby pierwsze w zakresie</title>
</head>
<body>
  <h2>Podaj zakres liczb, aby znaleźć liczby pierwsze</h2>
  <form method="post">
    <label for="poczatek">Początkowa liczba:</label>
    <input type="number" name="poczatek" id="poczatek" min="2" required>
    <label for="koniec">Końcowa liczba:</label>
    <input type="number" name="koniec" id="koniec" min="2" required><br><br>
    <input type="submit" value="Pokaż liczby pierwsze"><br><br>
  </form>
  <?php
      if (isset($_POST["poczatek"]) && isset($_POST["koniec"])) {
            $poczatek = $_POST["poczatek"];
            $koniec = $_POST["koniec"];
            $i = $poczatek;
            while ($i <= $koniec) {
                  $isPrime = true;
                  $j = 2;
                  while ($j <= sqrt($i)) {
                        if ($i % $j == 0) {
                              $isPrime = false;
                              break;
                        }
                        $j++;
                  }
                  if ($isPrime) {
                        echo "$i jest liczbą pierwszą<br>";
                  }
                  $i++;
            }
```

```
        }
    ?>
    </body>
    </html>
```

Temat: Pętla do...while w języku PHP.

Zadanie:

Zapoznaj się z opisem pętli `do...while` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_looping_do_while.asp

Pętla `do...while` zawiera instrukcje, które będą wykonane co najmniej raz, a następnie zostanie sprawdzony warunek pętli (wartość `true` decyduje o dalszym wykonaniu instrukcji). Ogólna składnia pętli `do...while`:

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

Przykład zastosowania pętli `do...while`:

```
<?php
  echo "Skrypt wyświetlający stan licznika pętli.<br>";
  $i=1;
  do {
    echo "<br>Wartość licznika: $i.";
    $i++;
  }
  while ($i <= 5);      //warunek zakończenia pętli
?>
```

Inny przykład zastosowania pętli `do...while`:

```
<?php
  echo "Skrypt wyświetlający stan licznika pętli.<br>";
  $i=10;
  do {
    echo "<br>Wartość licznika: $i.";
    $i--;
  }
  while ($i >= 5);
?>
```

Inny przykład zastosowania pętli `do...while` z niespełnionym warunkiem:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Pętla do...while</title>
</head>
<body>
  <?php
      $licznik = 7;
      do {
            echo "Aktualna wartość liczby: $licznik<br>";
            $licznik++;
      } while ($licznik < 5);
  ?>
  <p>As you can see, the code is executed once, even if the condition is never
true.</p>
</body>
</html>
```

Inny przykład zastosowania pętli `do...while`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli z wymuszonym
zatrzymaniem.<br>";
```

```
  $i=0;
  do {
    $i++;   //inkrementacja koniecznie przed instrukcją break
    if ($i==4) break;
    echo $i . " ";
  }  while ($i <= 10)
?>
```

Inny przykład zastosowania pętli `do...while`:

```
<?php
  echo "Skrypt wyświetlający wartości licznika pętli bez jednej wartości.<br>";
  $i=0;
  do {
    $i++;   //inkrementacja koniecznie przed instrukcją continue
    if ($i==4) continue;
    echo $i . " ";
  } while ($i <= 10);
?>
```

Inny przykład zastosowania pętli `do...while`:

```
<?php
  echo "Skrypt wyświetlający liczby nieparzyste z zakresu od -10 do 10.<br>";
  $i=-10;
  do {
    if ($i % 2 != 0)
      echo $i.",\t";
    $i++;
  } while ($i <= 10);
?>
```

Wykonaj zadanie:

Napisz skrypt w PHP, który przeszukuje tablicę liczb całkowitych w poszukiwaniu liczby podanej przez użytkownika. Użyj pętli `do...while()`, aby przechodzić przez elementy tablicy, aż znajdziesz szukaną liczbę lub dojdziesz do końca tablicy.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
  $tablica = [3, 7, 12, 5, 9, 21, 4];
  $szukana = 9; // Można podstawić np. wartość z formularza
  $index = 0;
  $znaleziono = false;
  do {
    if ($tablica[$index] === $szukana) {
        $znaleziono = true;
        break;
    }
    $index++;
  } while ($index < count($tablica));
  if ($znaleziono) {
    echo "Liczba $szukana została znaleziona na pozycji $index.";
  } else {
    echo "Liczba $szukana nie występuje w tablicy.";
  }
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `do...while()`, który pobierze od użytkownika zakres liczb naturalnych od min 2 do max 100 i wyświetli wszystkie liczby pierwsze z tego zakresu. Liczby mają być wyświetlane w kolejnych komórkach tabeli HTML o liczbie kolumn podanej przez użytkownika. Zadbaj również o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Do wykonania tego zadania wykorzystaj skrypt z poprzedniego tematu lekcji, który wyświetla liczby pierwsze. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_do_while_ pierwsze.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `do...while()`, który pobierze od użytkownika liczbę elementów tablicy oraz liczbę, która będzie szukana w tej tablicy. Elementy tablicy mają być generowane losowo z zakresu od 1 do 100 oraz wyświetlone na stronie. Zadbaj również o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_ do_while_tablica.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Pętla foreach w języku PHP.

Zadanie:

Zapoznaj się z opisem pętli `foreach` w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_looping_foreach.asp

Pętla `foreach` w języku PHP służy do iteracji po elementach tablicy. Ogólna składnia pętli `foreach`:

```
foreach ($tablica as $wartosc)
{
  instrukcje;
}
```

lub dla tablicy asocjacyjnej:

```
foreach ($tablica as $klucz => $wartosc)
{
  instrukcje;
}
```

Przykład zastosowania pętli `foreach`:

```
<?php
  echo "Skrypt wyświetlający nazwy kolorów z tablicy.<br>";
  echo "Zdefiniowane nazwy kolorów:<br>";
  $colors = array("red", "green", "blue", "yellow");
  foreach ($colors as $x)
  {
    echo "$x <br>";
  }
?>
```

Inny przykład zastosowania pętli `foreach`:

```
<?php
  $colors = array("red", "green", "blue", "yellow");
  foreach ($colors as $x)
  {
    if ($x == "blue") break;        //warunek zatrzymania pętli
    echo "$x <br>";
  }
?>
```

Inny przykład zastosowania pętli `foreach`:

```
<?php
  $colors = array("red", "green", "blue", "yellow");
  foreach ($colors as $x)
  {
    if ($x == "blue") continue;     //warunek pominięcia iteracji
    echo $x." ";
  }
  echo "<br>";
  foreach ($colors as &$x)
  {
    if ($x == "blue") $x = "pink"; //zmiana wartości elementu tablicy
    echo $x." ";
  }
?>
```

Inny przykład zastosowania pętli `foreach`:

```
<?php
  $imiona = ["Anna", "Bartek", "Celina", "Darek"];
  array_push($imiona, "Maria");
  foreach ($imiona as $imie) {
    echo "Cześć, $imie!<br>";
  }
?>
```

Inny przykład zastosowania pętli `foreach`:

```
<?php
  echo "Sumowanie liczb z listy\n";
  $lista = [1, 2, 3, 4, 5];
  $suma = 0;
  foreach ($lista as $liczba) {
    $suma += $liczba;
  }
  echo "<br>Suma liczb zapisanych w tablicy wynosi: $suma \n";
?>
```

Inny przykład zastosowania pętli `foreach`:

```
<?php
  echo "Skrypt obliczający sumę liczb zapisanych w tablicy.<br>";
  echo "Wartości liczb zapisanych w tablicy:<br>";
  $liczby = [5, 11, 3, 9, -6, 0, -17, 4];
  array_push($liczby, -2, 18);
  sort($liczby);
  foreach ($liczby as $x) {
    echo $x." | ";
    $s+=$x;
  }
  echo "<br>Suma liczb zapisanych w tablicy wynosi: $s<br>";
  echo "<br>minimum: ".$liczby[0];
  echo "<br>maximum: ".$liczby[(count($liczby)-1)];
  echo "<br>Inaczej obliczona suma liczb w tablicy: ".array_sum($liczby);
?>
```

Przykład zastosowania pętli `foreach` z dostępem do kluczy (indeksów):

```
<?php
  $oceny = ["Jan" => 5, "Wojtek" => 4, "Tomek" => 3];
  $oceny["Ania"] = 6;
  foreach ($oceny as $uczen => $ocena) {
    echo "$uczen otrzymał(a) ocenę: $ocena<br>";
  }
?>
```

Inny przykład zastosowania pętli `foreach` z dostępem do kluczy (indeksów):

```
<?php
  echo "Skrypt wykorzystujący tablicę asocjacyjną (indeksem jest klucz).<br>";
  echo "Lista uczniów zapisana w tablicy:<br>";
  $uczniowie = array("Jan" => "Nowak", "Anna" => "Kowalska", "Piotr" => "Adamski");
  $uczniowie += ["Wojtek" => "Mejer","Maria" => "Kuraś"];
  foreach ($uczniowie as $klucz => $wartosc) {
    echo $klucz." ".$wartosc."<br>";
  }
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `foreach()`, który obliczy średnią ocen uczniów.

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
  $oceny = [
    "Ala" => 5,
    "Bartek" => 4,
    "Celina" => 3,
    "Darek" => 5,
```

```
    "Ela" => 2
  ];
  $suma = 0;
  $liczbaUczniow = count($oceny);
  foreach ($oceny as $uczen => $ocena) {
    echo "$uczen otrzymał(a) ocenę: $ocena<br>";
    $suma += $ocena;
  }
  $srednia = $suma / $liczbaUczniow;
  echo "Średnia ocen: " . round($srednia, 2);
  //echo "Średnia ocen: " . round(array_sum($oceny)/count($oceny)), 2);
?>
```

Wykonaj zadanie:

Napisz skrypt w języku PHP z wykorzystanie pętli `foreach()`, który obliczy średnią ocen dla poszczególnych uczniów oraz średnią ogólną z dokłanością do dwóch miejsc po przecinku. Pobierz dane o uczniach z tablicy wielowymiarowej. Przykładowa tablica wielowymiarowa:

```
<?php
  $uczniowie = [
    "Ala" => [5, 4, 5],
    "Bartek" => [3, 4, 2],
    "Celina" => [5, 5, 5],
    "Darek" => [2, 3, 3],
    "Ela" => [4, 4, 4]
  ];
?>
```

Przykładowe skrócone obliczenie średnich ocen dla poszczególnych uczniów:

```
foreach($uczniowie as $uczen => $oceny) {
      echo "$uczen ma średnią: ".round(array_sum($oceny)/count($oceny), 2)."<br>";
}
```

Przykładowe rozwiązanie do powyższego zadania:

```
<?php
  $uczniowie = [
    "Ala" => [5, 4, 5],
    "Bartek" => [3, 4, 2],
    "Celina" => [5, 5, 5],
    "Darek" => [2, 3, 3],
    "Ela" => [4, 4, 4]
  ];
  $sumaWszystkich = 0;
  $liczbaOcen = 0;
  foreach ($uczniowie as $imie => $oceny) {
    $suma = array_sum($oceny);
    $srednia = $suma / count($oceny);
    echo "$imie: " . implode(", ", $oceny) . " — Średnia: " . round($srednia, 2) .
"<br>";
    $sumaWszystkich += $suma;
    $liczbaOcen += count($oceny);
  }
  $sredniaGlobalna = $sumaWszystkich / $liczbaOcen;
  echo "Średnia wszystkich uczniów: " . round($sredniaGlobalna, 2);
?>
```

W powyższym przykładzie można zastosować zmienną w pętli a potem obliczyć średnią wszystkich uczniów (bez zaokrąglania wartości średnich, aby uniknąć błędu w obliczeniach):

```
...
$sumaSrednich+=array_sum($oceny)/count($oceny);
}
$sredniaGlobalna = $sumaSrednich/count($uczniowie);
```

Wykonaj zadanie:

Zmodyfikuj powyższy skrypt tak, aby posortował uczniów według średniej ocen oraz pokazywał tylko tych, którzy mają średnią powyżej 4,00. Rozbuduj tablicę o kolejne pięć rekordów oraz dodaj kolejne

przykładowe oceny uczniom, np. do pięciu. Dodaj również formularz dla możliwości wprowadzania dodatkowych danych do tablicy (klucz i wartość) przez użytkownika strony. Zadbaj o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_foreach.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Tablice w języku PHP.

Tablica to specjalna zmienna, która może przechowywać wiele wartości pod jedną nazwą. Dostęp do wartości można uzyskać poprzez odwołanie się do numeru indeksu lub nazwy. W PHP istnieją trzy typy tablic:
- tablice indeksowane – tablice z indeksem numerycznym,
- tablice asocjacyjne – tablice z nazwanymi kluczami,
- tablice wielowymiarowe – tablice zawierające jedną lub więcej tablic.

Zadanie:

Zapoznaj się z opisem funkcji w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_arrays.asp

W tablicach indeksowanych każdy element ma numer indeksu. Domyślnie pierwszy element ma indeks 0, drugi element ma indeks 1 itd.

Przykład zastosowania tablicy indeksowanej:

```
<?php
	echo "<br>Obliczanie średniej ocen<br>";
	// tablica zwykła
	$oceny = [5, 4, 3, 5, 2];
	echo "Przykładowa lista ocen:<br>";
	for ($i = 0; $i < count($oceny); $i++) {
		// wyświetlenie wartości konkretnego element tablicy indeksowej
		echo $oceny[$i]." ";
	}
	echo "<br>Średnia ocen w tablicy wynosi: " . array_sum($oceny)/count($oceny);
?>
```

Inny przykład zastosowania tablicy indeksowanej:

```
<?php
	echo "<br>Obliczanie średniej ocen<br>";
	// tablica zwykła
	$oceny = array(5, 4, 3, 5, 2);
	echo "Przykładowa lista ocen:<br>";
	for ($i = 0; $i < count($oceny); $i++) {
		// wyświetlenie wartości konkretnego element tablicy indeksowej
		echo $oceny[$i]." ";
	}
	echo "<br>Średnia ocen w tablicy wynosi: " . array_sum($oceny)/count($oceny);
?>
```

Przykład zastosowania tablicy indeksowanej z różnymi typami danych:

```
<?php
	// create array:
	$myArr = array("Volvo", 15, ["apples", "bananas"]);
	$myArr[0]();
	echo "<br>";
	var_dump($myArr);
	echo "<br>";
	print_r($myArr);
?>
```

Przykład zastosowania tablicy asocjacyjnej:

```
<?php
	echo "<br>Obliczanie średniej ocen<br>";
	// tablica asocjacyjna
```

```
	$oceny['Adam'] = [5, 4, 3, 5, 2];
	echo "Przykładowa lista ocen:<br>";
	foreach ($oceny['Adam'] as $ocena) {
		echo $ocena . " ";
	}
	echo "<br>Średnia ocen: " . array_sum($oceny['Adam'])/count($oceny['Adam']);
?>
```

Przykład zastosowania tablicy asocjacyjnej dwuwymiarowej:

```
<?php
	echo "<br>Obliczanie średniej ocen<br>";
	// deklaracja elementu tablicy asocjacyjnej dwuwymiarowej
	$oceny['Adam']['Nowak'] = [5, 4, 3, 5, 2];
	echo "Przykładowa lista ocen:<br>";
	foreach ($oceny['Adam']['Nowak'] as $ocena) {
		echo $ocena . " ";
	}
	echo "<br>Średnia ocen w tablicy wynosi: " .
array_sum($oceny['Adam']['Nowak'])/count($oceny['Adam']['Nowak']);
	echo "<br>Wydruk tablicy poleceniem var_dump()<br>";
	var_dump($oceny);
	echo "<br>Wydruk tablicy poleceniem print_r()<br>";
	print_r($oceny);
?>
```

Inny przykład zastosowania tablicy asocjacyjnej dwuwymiarowej:

```
<pre>
<?php
	// deklaracja tablicy asocjacyjnej dwuwymiarowej
	$uczniowieOceny = [
		"Jan" => ["Kowalski" => [5, 4, 5]],
		"Anna" => ["Nowak" => [3, 4, 2]],
		"Piotr" => ["Wiśniewski" => [5, 5, 5]],
		"Maria" => ["Wójcik" => [5, 6, 6]],
		"Kamil" => ["Zieliński" => [4, 3, 5]],
		"Alicja" => ["Malinowska" => [3, 2, 2]],
		"Jolanta" => ["Czekalska" => [1, 2, 4]],
		"Paulina" => ["Kuraś" => [2, 3, 3]]
	];
	$uczniowieOceny['Adam']['Nowak'] = [5, 4, 3, 5, 2];

	foreach ($uczniowieOceny as $imie => $nazwiskoOceny) {
		foreach ($nazwiskoOceny as $nazwisko => $oceny) {
			$srednia = array_sum($oceny) / count($oceny);
		// PHP_EOL (wymaga <pre>) == <br> == \r\n
			echo "$imie $nazwisko - średnia: " . round($srednia, 2) . PHP_EOL;
		}
	}
?>
</pre>
```

Inny przykład zastosowania tablicy asocjacyjnej dwuwymiarowej:

```
<pre>
<?php
	$uczniowieOceny = [
		"Jan" => ["Kowalski" => [5, 4, 5]],
		"Anna" => ["Nowak" => [3, 4, 2]],
		"Piotr" => ["Wiśniewski" => [5, 5, 5]],
		"Maria" => ["Wójcik" => [5, 6, 6]],
		"Kamil" => ["Zieliński" => [4, 3, 5]],
		"Alicja" => ["Malinowska" => [3, 2, 2]],
		"Jolanta" => ["Czekalska" => [1, 2, 4]],
		"Paulina" => ["Kuraś" => [2, 3, 3]]
	];
	$wszystkieOceny = [];
	foreach ($uczniowieOceny as $imie => $nazwiskoOceny) {
		foreach ($nazwiskoOceny as $nazwisko => $oceny) {
			$wszystkieOceny = array_merge($wszystkieOceny, $oceny);
```

```
            }
        }
        $sredniaKlasy = array_sum($wszystkieOceny) / count($wszystkieOceny);
        echo "Średnia klasy: " . round($sredniaKlasy, 2) . PHP_EOL;
    ?>
    </pre>
```

Inny przykład zastosowania tablicy asocjacyjnej dwuwymiarowej:

```
    <pre>
    <?php
        $uczniowieOceny = [
            "Jan" => ["Kowalski" => [5, 4, 5]],
            "Anna" => ["Nowak" => [3, 4, 2]],
            "Piotr" => ["Wiśniewski" => [5, 5, 5]],
            "Maria" => ["Wójcik" => [5, 6, 6]],
            "Kamil" => ["Zieliński" => [4, 3, 5]],
            "Alicja" => ["Malinowska" => [3, 2, 2]],
            "Jolanta" => ["Czekalska" => [1, 2, 4]],
            "Paulina" => ["Kuraś" => [2, 3, 3]]
        ];
        $wyniki = [];
        foreach ($uczniowieOceny as $imie => $nazwiskoOceny) {
            foreach ($nazwiskoOceny as $nazwisko => $oceny) {
                $srednia = array_sum($oceny) / count($oceny);
                $wyniki["$imie $nazwisko"] = $srednia;
            }
        }
        // najlepszy uczeń array_keys() zwraca klucz
        $najlepszyUczen = array_keys($wyniki, max($wyniki))[0];
        $najlepszaSrednia = max($wyniki);
        // najsłabszy uczeń
        $najsłabszyUczen = array_keys($wyniki, min($wyniki))[0];
        $najsłabszaSrednia = min($wyniki);
        echo "Najlepszy uczeń: $najlepszyUczen (średnia: " . round($najlepszaSrednia,
    2) . ")" . PHP_EOL;
        echo     "Najsłabszy     uczeń:     $najsłabszyUczen     (średnia:     "     .
    round($najsłabszaSrednia, 2) . ")" . PHP_EOL;
    ?>
    </pre>
```

Inny przykład zastosowania tablicy asocjacyjnej dwuwymiarowej:

```
    <pre>
    <?php
        $uczniowieOceny = [
            "Jan" => ["Kowalski" => [5, 4, 5]],
            "Anna" => ["Nowak" => [3, 4, 2]],
            "Piotr" => ["Wiśniewski" => [5, 5, 5]],
            "Maria" => ["Wójcik" => [5, 6, 6]],
            "Kamil" => ["Zieliński" => [4, 3, 5]],
            "Alicja" => ["Malinowska" => [3, 2, 2]],
            "Jolanta" => ["Czekalska" => [1, 2, 4]],
            "Paulina" => ["Kuraś" => [2, 3, 3]]
        ];
        $wyniki = [];
        foreach ($uczniowieOceny as $imie => $nazwiskoOceny) {
            foreach ($nazwiskoOceny as $nazwisko => $oceny) {
                $srednia = array_sum($oceny) / count($oceny);
                $wyniki["$imie $nazwisko"] = $srednia;
            }
        }
        // sortowanie malejąco po średniej
        arsort($wyniki);
        echo "Ranking klasy:\n";
        $pozycja = 1;
        foreach ($wyniki as $uczen => $srednia) {
            echo $pozycja . ". $uczen - średnia: " . round($srednia, 2) . "\n";
            $pozycja++;
        }
```

```
?>
</pre>
```

Temat: Funkcje tablic w języku PHP.

Funkcje tablic są funkcjami wbudowanymi do języka PHP, a służą do wykonywania operacji na tablicach.
Do zliczania elementów tablicy służą funkcje:

- `count($tablica)` – zwraca liczbę elementów w tablicy,
- `sizeof($tablica)` – alias funkcji `count()`.
- `array_unique($tablica)` – pobiera tablicę i zwraca nową tablicę bez powtarzających się wartości,
- `array_merge($tablica1, $tablica2, …)` – łączy elementy wybranych tablic i zwraca tablicę wynikową.

Przykład zastosowania funkcji tablicowych do zliczania elementów tablicy:

```
<?php
  $imiona=array("Ala","Adam","Tomasz","Janusz","Kasia","Irena");
  $dlugosc=count($imiona);    // lub sizeof($imiona);
  echo "Tablica imiona zawiera $dlugosc elementów.<br>";
  echo "Wszystkie elementy tablicy:<br>";
  for ($i=0;$i<$dlugosc;$i++)
  {
    echo "element[$i]=$imiona[$i]<br>";
  }
?>
```

Funkcje służące do sortowania elementów tablicy:

- `sort()` – sortowanie tablicy od wartości najmniejszej do największej (rosnąco),
- `rsort()` – sortowanie tablicy od wartości największej do najmniejszej (malejąco),
- `asort()` – sortowanie tablicy asocjacyjnej według zawartości od najmniejszej do największej,
- `arsort()` – sortuje tablicę asocjacyjną w kolejności malejącej według wartości (odwrotne do `asort()`),
- `ksort()` – sortowanie tablicy asocjacyjnej według klucza od najmniejszej do największej,
- `krsort()` – sortuje tablicę asocjacyjną w kolejności malejącej według klucza (odwrotne do `ksort()`).

Inne wbudowane funkcje w języku PHP, których można używać na tablicach:

- `key()` – pobiera klucz z tablicy,
- `array_keys()` – zwraca wszystkie klucze tablicy,
- `array_push()` – wstawia jeden lub więcej elementów na koniec tablicy,
- `array_sum()` – zwraca sumę wartości w tablicy,
- `shuffle()` – tasuje (miesza losowo) tablicę,
- `array_splice()` – usuwa i zastępuje określone elementy tablicy,
- i inne.

Przykład zastosowania funkcji tablicowych do sortowania elementów tablicy:

```
<?php
  $imiona=array("Ala","Adam","Tomasz","Janusz","Kasia","Irena");
  $dlugosc=sizeof($imiona);
  echo "Tablica imiona przed sortowaniem:<br>";
  foreach ($imiona as $i)
    {
      echo "$i - ";
    }
  sort($imiona);
  echo "<br><br>Tablica imiona po posortowaniu:<br>";
  foreach ($imiona as $i)
    {
      echo "$i - ";
    }
?>
```

Przykład zastosowania funkcji tablicowych do sortowania elementów tablicy asocjacyjnej:

```
<?php
```

```
        $osoby = [
          "Jan" => "Kowalski",
          "Anna" => "Nowak",
          "Piotr" => "Wiśniewski",
          "Maria" => "Wójcik",
          "Kamil" => "Zieliński",
          "Alicja" => "Malinowska",
          "Jolanta" => "Czekalska",
          "Paulina" => "Kuraś"
        ];
        echo "<b>Tablica osób drukowana poleceniem print_r():</b><br>";
        // Wyświetlenie tablicy
        print_r($osoby);
        echo "<b>Tablica osób przed posortowaniem:</b><br>";
        foreach ($osoby as $klucz=>$i)
        {
            echo "$i $klucz<br>";
        }
        asort($osoby);
        echo "<b>Tablica osób po posortowaniu według zawartości (nazwisk):</b><br>";
        foreach ($osoby as $klucz=>$i)
        {
            echo "$i $klucz<br>";
        }
        ksort($osoby);
        echo "<b>Tablica osób po posortowaniu według klucza (imion):</b><br>";
        foreach ($osoby as $klucz=>$i)
        {
            echo "$i $klucz<br>";
        }
    ?>
```

Wykonaj zadanie:

Utwórz skrypt, w którym zdefiniujesz 10-elementową tablicę i wpiszesz do niej następujące liczby: 120, 4, 1, -32, -12, 177, 43, -3, 345, 11. Następnie posortuj elementy tablicy rosnąco, a wynik wyświetl w oknie przeglądarki. Dodaj również formularz dla możliwości wprowadzania dodatkowych liczb do tablicy przez użytkownika strony. Zadbaj o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_sortowanie.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Funkcje daty i czasu w języku PHP.

Do funkcji wykonujących operacje na dacie i czasie należą:

- `TIME()` – zwraca aktualny czas w sekundach, który upłynął od 1 stycznia 1970 00:00:00 (tzw. znacznik czasu),
- `MICROTIME()` – zwraca znacznik czasu z dokładnością do milionowej części sekundy,
- `GETDATE()` – zwraca tablicę asocjacyjną zawierającą informacje o podanej dacie, dostępne klucze:
  - seconds,
  - minutes,
  - hours,
  - mday – dzień miesiąca,
  - wday – dzień tygodnia (0 – niedziela, 6 – sobota),
  - mon – miesiąc w postaci liczby,
  - year – rok w postaci liczby czterocyfrowej,
  - yday – numer danego dnia roku,
  - weekday – nazwa dnia tygodnia,
  - month – nazwa miesiąca,
  - 0 – aktualny znacznik czasu,
- `DATE(format[,znacznik czasu])` - pozwala na formatowanie daty i czasu, dostępne znaczniki:
  - d – dzień miesiąca w postaci liczby,
  - D – dzień miesiąca w postaci trzyliterowej,
  - l – nazwa dnia tygodnia,
  - w – numer dnia tygodnia,
  - z – dzień roku,
  - F – nazwa miesiąca,

- m – miesiąc w postaci liczby,
- M – trzyliterowa nazwa miesiąca,
- t – liczba dni w danym miesiącu (od 28 do 31),
- L – informacja czy rok jest przestępny (1 tak, 0 nie),
- Y – czterocyfrowa forma roku,
- y – dwucyfrowa forma roku,
- a – przed lub po południu (am, pm),
- A - przed lub po południu (AM, PM),
- g – godzina w formacie 12-godzinnym,
- G – godzina w formacie 24-godzinnym,
- h – godzina w formacie 12-godzinnym (01 do 12),
- H – godzina w formacie 24-godzinnym (00 do 23),
- i – minuty z zerami wiodącymi,
- s – sekundy z zerami wiodącymi,

– `MKTIME()` – zwraca znacznik czasu podanej daty, dostępne argumenty:

- `godzina,`
- `minuta,`
- `sekunda,`
- `miesiąc,`
- `dzień miesiąca,`
- `rok.`

Przykład zastosowania funkcji czasu:

```
<?php
  $data=getdate();
  $dzien=$data["mday"];       //ustawienie dnia miesiąca
  $miesiac=$data["mon"];
  $rok=$data["year"];
  if ($dzien<10) $dzien="0".$dzien;
  if ($miesiac<10) $miesiac="0".$miesiac;
  echo "Dzisiaj jest dzień: $dzien-$miesiac-$rok <br>";
  echo date("y-m-d")."<br>";
  echo date("d F Y")."<br>";
  echo date("g:i a")."<br>";
  $data=mktime(11,25,3,10,30,2025);
  echo date("d-m-Y G:i",$data);
?>
```

Zadanie:

Więcej informacji na temat funkcji daty i czasu dostępnych w języku PHP znajdziesz w serwisie w3schools: https://www.w3schools.com/php/php_ref_date.asp

Wykonaj zadanie:

Napisz skrypt w języku PHP, który pobierze od użytkownika datę i godzinę urodzenia a następnie wyświetli na ekranie następujące informacje:

- nr dnia w roku urodzenia,
- nazwa dnia tygodnia w roku urodzenia,
- liczba godzin jaka upłynęła od godziny narodzin,
- liczba dni jaka upłynęła od dnia urodzenia,
- liczba tygodni jaka upłunęła od dnia urodzenia,
- liczba miesięcy jaka upłunęła od dnia urodzenia,
- czas jaki upłynął od dnia urodzenia (format('%y lat, %m mies, %d dni, %h godz, %i min, %s sek').

Zadbaj o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_time.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Funkcje służące do formatowania ciągu znaków w języku PHP.

Zadanie:

Więcej informacji na temat funkcji formatowania ciągu znaków dostępnych w języku PHP znajdziesz w serwisie w3schools: https://www.w3schools.com/php/php_ref_string.asp

Przykładowe funkcje:
- `n12br()` – automatycznie wstawia znak
 przed każdym znakiem końca linii,
- `wordwrap()` – formatuje tekst w postaci kolumny o podanej szerokości,
- `strtoupper()` – zamienia litery na wielkie,
- `mb_strtoupper()` – zamienia litery na wielkie z polskimi znakami,
- `strtolower()` – zamienia litery na małe,
- `mb_strtolower()` – zamienia litery na małe z polskimi znakami,
- `ucfirst()` – zamienia pierwszą literę tekstu na dużą,
- `ucwords()` – zamienia pierwsze litery wyrazów na duże.

Przykład zastosowania funkcji formatowania tekstu:
```
<?php
  $tekst="Alicja w krainie czarów i Alicja po drugiej stronie lustra.";
  echo "<u>Oryginalny tekst:</u><br>".$tekst."<br>";
  echo "<u>Tekst po zastosowaniu funkcji strtolower:</u><br>".strtolower($tekst)."<br>";
  echo "<u>Tekst po zastosowaniu funkcji ucfirst:</u><br>".ucfirst($tekst)."<br>";
  echo "<u>Tekst po zastosowaniu funkcji ucwords:</u><br>".ucwords($tekst)."<br>";
?>
```

Funkcje służące do usuwania ciągu znaków (spacje, tabulacje, końce linii):
- `trim()` – usuwa białe znaki z początku i końca danego ciągu znaków,
- `ltrim()` – usuwa białe znaki z początku podanego ciągu znaków,
- `rtrim()` – usuwa białe znaki z końca podanego ciągu znaków,
- `strlen()` – podaje długość łańcucha znaków.

W przypadku potrzeby usunięcia innych znaków należy do funkcji podać znaki do usunięcia, np.:
```
trim($tekst, "abc");
```

Przykład zastosowania funkcji usuwania znaków z tekstu:
```
<?php
  $tekst="Alicja w krainie czarów i Alicja po drugiej stronie lustra.";
  echo "Długość tekstu wynosi: ".strlen($tekst)."<br>";
  echo "Tekst po zastosowaniu funkcji trim:<br>".trim($tekst)."<br>";
?>
```

Funkcje służące do wyszukiwania podciągów znaków w podanym łańcuchu znaków:
- `strstr()` – sprawdza czy w podanym tekście jest dany podciąg, gdy nie to zwraca FALSE,
- `strpos()` – sprawdza, czy szukany podciąg znajduje się w tekście od podanego miejsca,
- `substr()` – zwraca część ciągu źródłowego,
- `strtok()` – pozwala na podzielenie tekstu.

Przykład zastosowania powyższych funkcji:
```
<?php
  $tekst="Ala Nowak, ul. Jedności 9, 75-410 Koszalin, tel. 504-321-001";
  echo "<u>Adres: </u>".strstr($tekst, "ul.")."<br>";
  echo "<u>Telefon: </u>".substr($tekst, 50)."<br>";
  echo "<b>Podzielony tekst: </b><br>";
  $t=",");
  $podtekst=strtok($tekst,$t);
  while (is_string($podtekst))
  {
    if ($podtekst)
    {
      echo $podtekst."<br>;
    }
    $podtekst=strtok($t);
  }
?>
```

Funkcje służące do porównywania ciągów znaków:

- `strcmp("tekst1","tekst2")` – porównuje długość ciągów, zwraca 0 gdy ciągi są równe,
- `strcasecmp()` – porównuje ciągi nie uwzględniając wielkości liter.

Wykonaj zadanie:

Utwórz skrypt, w którym zmiennej $przyslowie przypiszesz treść dowolnego przysłowia. Następnie zastosuj odpowiednią funkcję i sprawdź, czy w podanym przysłowiu znajduje się ciąg "do", oraz wyświetl odpowiedni komunikat. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_tekst.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Funkcje i ich zastosowanie w języku PHP.

Funkcje to fragment kodu, który może być wielokrotnie wykorzystywany w różnych miejscach programu. Wywołanie funkcji polega na podaniu jej nazwy oraz listy argumentów. W języku PHP istnieje wiele funkcji wbudowanych, które są przydatne podczas tworzenia aplikacji internetowych. Można również tworzyć własne funkcje.

Zadanie:

Zapoznaj się z opisem funkcji w języku PHP publikowanym na witrynie internetowej serwisu w3schools: https://www.w3schools.com/php/php_functions.asp

Ogólna postać funkcji w języku PHP:

```
function nazwa_funkcji(arg1, ..., argn)        //nagłówek
{
  instrukcje;                                  //ciało funkcji
}
nazwa_funkcji(arg1, ..., argn)                 //wywołanie funkcji
```

Przykład zastosowania funkcji bez parametrów:

```
<?php
  function myMessage() {
    echo "Hello world!";
  }
  myMessage();
?>
```

Przykład zastosowania funkcji z jednym parametrem:

```
<?php
  function familyName($fname) {
    echo "$fname Refsnes.<br>";
  }
  familyName("Jani");
  familyName("Hege");
  familyName("Kai Jim");
?>
```

Przykład zastosowania funkcji z dwoma parametrami:

```
<?php
  echo "Skrypt wykorzystujący funkcję sumującą dwie liczby.<br>";
  function suma($a,$b) {
    echo "Suma liczb $a oraz $b wynosi: ".($a+$b);
  }
  suma(4,9);
?>
```

Przykład zastosowania funkcji z wieloma parametrami:

```
<?php
  echo "Skrypt wykorzystujący funkcję obliczającą wyrażenie: 2a-0,5b+0,5c.<br>";
  function wyrazenie($a,$b,$c) {
    echo "Wartość wyrażenia dla liczb $a, $b oraz $c wynosi: ".(2*$a-0.5*$b+$c/2);
  }
  wyrazenie(4,2,-6);
```

```
?>
```

Przykład zastosowania funkcji zwracającej wartość:

```
<?php
  function sum($x, $y) {
    return $x + $y;     // funkcja zwracająca wartość
  }
  echo "7 + 13 = " . sum(7,13) . "<br>";
  echo "2 + 4 = " . sum(2,4);
?>
```

Inny przykład zastosowania funkcji zwracającej wartość:

```
<?php
  echo "Skrypt wykorzystujący funkcję obliczającą wyrażenie: 2a-0,5b+0,5c.<br>";
  function wyrazenie($a,$b,$c) {
    $wynik=(2*$a-0.5*$b+$c/2);
    return $wynik;      // zwracana wartość
  }
  echo "Wartość wyrażenia dla podanych liczb wynosi: ".wyrazenie(4,2,-6);
?>
```

Przykład zastosowania funkcji z wartością domyślną argumentu funkcji:

```
<?php
      function setHeight($minheight = 50) {
        echo "The height is: $minheight <br>";
      }
      setHeight(350);
      setHeight();
?>
```

Inny przykład zastosowania funkcji z argumentami domyślnymi:

```
<?php
  echo "Skrypt obliczający pole trójkąta.<br>";
  function pole_trojkata($a=3,$b=4) // wartości domyślne mogą zostać zastąpione
  {
    return ($a*$b)/2;
  }
  echo    "Wartość    pola    trójkąta    z    wartościami    domyślnymi    funkcji:
".pole_trojkata()."<br>";
  echo    "Wartość    pola    trójkąta    z    wartościami    podanymi    do    funkcji:
".pole_trojkata(2,3);
?>
```

Przykład zastosowania funkcji jako wartość zmiennej typu tablicowego:

```
<?php
  // function example:
  function myFunction() {
    echo "This text comes from a function";
  }
  // create array:
  $myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);
  // wywołanie funkcji z elementu tablicy:
  $myArr[3]();
  echo "<br>";
  var_dump($myArr);
  echo "<br>";
  print_r($myArr);
?>
```

W PHP zmienne można deklarować w dowolnym miejscu skryptu. **Zakres zmiennej** to część skryptu, w której można odwołać się do zmiennej i jej użyć. W skryptach PHP istnieją trzy różne zakresy zmiennych:

- lokalny – są definiowane wewnątrz funkcji i mają zasięg tylko w tej funkcji, poza tą funkcją zmienne nie są widoczne, domyślnie po zakończeniu działania funkcji zmienne są usuwane,
- globalny – są definiowane w bloku głównym skryptu i są dostępne w całym skrypcie oprócz wnętrza funkcji (domyślnie),

– superglobalny – są dostępne w każdym miejscu skryptu, zmienne odczytywane z tablicy.

Dostęp do zmiennych globalnych wewnątrz funkcji można uzyskać poprzez zastosowanie słowa kluczowego `global` lub za pomocą zmiennej superglobalnej `$GLOBALS`.

Przykład zastosowania funkcji ze zmienną lokalną:

```
<?php
  echo "Skrypt obliczający pole powierzchni kwadratu.<br>";
  function kwadrat($a)
  {
    $pole=$a*$a;
    return $pole;
  }
  echo "Pole kwadratu o boku 4 wynosi: ".kwadrat(4);
  echo "<br>Długość boku kwadratu wynosi: $a";       //zmienna a nie jest dostępna
?>
```

Zwykle po wykonaniu funkcji wszystkie jej zmienne zostają usunięte.

```
<?php
  function myTest() {
    $x = 1;      //zmienna zostanie zapamiętana tylko podczas wykonania funkcji
    echo $x;
    $x++;
  }
  myTest();      // 1
  echo "<br>";
  myTest();      // 1
  echo "<br>";
  myTest();       // 1
?>
```

Czasami jednak nie chcemy, aby zmienna lokalna została usunięta. Potrzebujemy jej na przykład w dalszej części skryptu. Aby zapamiętać wartość zmiennej lokalnej należy przy pierwszej deklaracji zmiennej użyć słowa kluczowego `static`, np.

```
<?php
  function myTest() {
    static $x = 0;       //zmienna zostanie zapamiętana
    echo $x;
    $x++;
  }
  myTest();      // 1
  echo "<br>";
  myTest();      // 2
  echo "<br>";
  myTest();       // 3
?>
```

Przykład zastosowania funkcji z odwołaniem do zmiennej globalnej:

```
<?php
  echo "Skrypt obliczający pole powierzchni kwadratu.<br>";
  $a=4;
  function kwadrat()
  {
    global $a;
    $a+=1;  //zmieniona zostanie wartość zmiennej globalnej
    $pole=$a*$a;
    return $pole;
  }
  echo "Pole kwadratu o boku $a wynosi: ".kwadrat();
  echo "<br>Długość boku kwadratu wynosi: $a";
?>
```

Przykład zastosowania funkcji ze zmienną superglobalną:

```
<?php
  echo "Skrypt obliczający pole powierzchni kwadratu.<br>";
```

```
  $a=4;
  function kwadrat()
  {
    $pole=$GLOBALS['a']* $GLOBALS['a'];
    return $pole;
  }
  echo "Pole kwadratu o boku $a wynosi: ".kwadrat();
?>
```

Inny przykład zastosowania funkcji ze zmienną superglobalną:

```
<?php
  $x=2;
  function wypisz() {
    $x=$GLOBALS['x'];
    $x++;         //zmieniona zostanie wartość kopii zmiennej
    echo "Wartość \$x wewnątrz funkcji wynosi ".$x."</br>";
  }
  wypisz();
  echo "Wartość \$x poza funkcją wynosi ".$x."</br>";
?>
```

Przykład zastosowania funkcji z argumentem przekazywanym przez wartość:

```
<?php
  echo "Skrypt sprawdzający działanie zmiennych w funkcji.<br>";
  $x=7;
  function zwieksz_liczbe($x)
  {
    $x+=5;  // działanie dotyczy tylko wnętrza funkcji
    echo "<br>Wartość x wewnątrz funkcji: ".$x;      // 12
  }
  echo "<br>Wartość x przed wywołaniem funkcji: ".$x;// 7
  zwieksz_liczbe($x);
  echo "<br>Wartość x po wywołaniu funkcji: ".$x;    // 7
?>
```

Przykład zastosowania funkcji z argumentem przekazywanym przez referencję (tzw. przezwisko):

```
<?php
  echo "Skrypt sprawdzający działanie zmiennych w funkcji.<br>";
  $x=7;
  function zwieksz_liczbe(&$x)     // wstawiamy znak & (ampersand) przed argumentem
  {
    $x+=5;  // działanie dotyczy również zmiennej globalnej
  }
  echo "Wartość x przed wywołaniem funkcji: ".$x;    // 7
  zwieksz_liczbe($x);
  echo "<br>Wartość x po wywołaniu funkcji: ".$x;    // 12
?>
```

Przykład zastosowania funkcji z zadeklarowanym typem zwracanej wartości (funkcja declare musi być w pierwszej linii kodu bez żadnych znaków poprzedzających, skrypt PHP musi być przed kodem HTML, kodowanie UTF-8 bez BOM):

```
<?php
declare(strict_types=1); // strict requirement
  function addNumbers(float $a, float $b)
  : float // deklaracja typu wyniku funkcji
  {
    return $a + $b;
  }
  echo addNumbers(1.2, 5.2)
?>
```

Inny przykład zastosowania funkcji z zadeklarowanym typem zwracanej wartości:

```
<?php
declare(strict_types=1);
  function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
```

```
  }
  $result = addNumbers(1.2, 5.2);
?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Funkcje w PHP</title>
</head>
<body>
  <?php echo $result; ?>
</body>
</html>
```

Inny przykład zastosowania funkcji, który generuje i wyświetla 5 zestawów po 6 unikalnych liczb losowych z zakresu od 1 do 49:

```
<?php
  function generujZestawLiczb($iloscLiczb = 6, $zakres = 49) {
    $liczby = range(1, $zakres);
    shuffle($liczby);
    // array_slice() zwraca elementy tablicy od wskazanego i tyle ile podano
    return array_slice($liczby, 0, $iloscLiczb);
  }
  echo "<h2>Wylosowane zestawy liczb:</h2>";
  for ($i = 1; $i <= 5; $i++) {
    $zestaw = generujZestawLiczb();
    sort($zestaw); // opcjonalnie: sortowanie rosnąco
    echo "Zestaw $i: " . implode(", ", $zestaw) . "<br>";    //funkcja implode()
łączy elementy tablicy
  }
?>
```

Wyjaśnienie do powyższego skryptu:

- używa funkcji `shuffle()` do losowego tasowania tablicy liczb od 1 do 49,
- wybiera pierwsze 6 liczb jako zestaw (zapobiega powtórzeniom),
- powtarza działanie 5 razy, tworząc 5 zestawów liczb losowych,
- sortuje liczby rosnąco w każdym zestawie dla lepszej czytelności.

Przykład zastosowania funkcji z zadeklarowanym innym typem zwracanej wartości niż typy argumentów:

```
<?php
  declare(strict_types=1); // strict requirement
  function addNumbers(float $a, float $b)
  : int // deklaracja typu wyniku funkcji
  {
    return (int)($a + $b);
  }
  echo addNumbers(1.2, 5.2)
?>
```

Przykład zastosowania kilku funkcji w jednym pliku (może sprawiać problemy z przejrzystością treści):

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Funkcje w języku PHP</title>
  <meta charset="utf-8">
  <style>
    table,td,th
    {
      margin: auto;
      border: 2px blue double;
      border-collapse: collapse;
    }
    th, .figura
    {
      background: #003366;
      height: 50px;
```

```
        color: white;
      }
    </style>
  </head>
  <body>
  <?php
    function trojkat($a,$b)
    {
      echo "Pole powierzchni trójkąta o podstawie $a oraz wysokości $b wynosi:
".(($a*$b)/2);
    }
    function prostokat($a,$b)
    {
      echo "Pole powierzchni prostokąta o bokch $a i $b wynosi: ".($a*$b);
    }
    function trapez($a,$b,$h)
    {
      $pole=(($a*$b)*$h)/2;
      echo "Pole powierzchni trapezu o podstawach $a i $b oraz wysokości $h wynosi:
".$pole;
    }
    function kolo($r)
    {
      $pole=3.14*$r*$r;
      echo "Pole powierzchni koła o promieniu $r wynosi: ".$pole;
    }
  ?>
    <table>
    <tr>
      <th>Nazwa figury</th>
      <th>Rysunek</th>
      <th>Przykładowe pole</th>
    </tr>
    <tr>
      <td class="figura">TRÓJKĄT</td>
      <td><img src="trojkat.png"></td>
      <td>
          <?php trojkat(4,7);?>
        </td>
    </tr>
    <tr>
      <td class="figura">PROSTOKĄT</td>
      <td><img src="prostokat.png"></td>
      <td>
        <?php prostokat(6,9); ?>
        </td>
    </tr>
    <tr>
      <td class="figura">TRAPEZ</td>
      <td><img src="trapez.png"></td>
      <td>
        <?php trapez(6,9,4); ?>
        </td>
    </tr>
    <tr>
      <td class="figura">KOŁO</td>
      <td><img src="kolo.png"></td>
      <td>
  <?php kolo(3); ?>
        </td>
    </tr>
    </table>
  </body>
  </html>
```

W języku PHP można wykorzystać instrukcje do dołączania skryptów z plików zewnętrznych, przez co możemy wielokrotnie wykorzystać kod zapisany w innych plikach. Do dyspozycji mamy następujące instrukcje:

- `include 'plik.php'` – jeżeli plik nie zostanie znaleziony, skrypt PHP będzie dalej wykonywany,
- `require 'plik.php'` – jeżeli plik nie zostanie odnaleziony, skrypt PHP przestanie działać.

Przykład zastosowania importu zewnętrznego skryptu:

```
// treść pliku php.include.skrypt.php
<?php
echo "<p>Testujemy dołączanie skryptów.</p>";
?>

<!-- // treść głównego skryptu, np. php.include.php -->
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Include w języku PHP</title>
</head>
<body>
<div class="container">
      <label>Zastosowanie funkcji include w języku PHP.</label>
      <?php
            include 'php.include.skrypt.php';
      ?>
</div>
</body>
</html>
```

Przykład zastosowania funkcji zdefiniowanych w zewnętrznym pliku (poprawia czytelność):

```
// treść pliku funkcje.php
<?php
  define("PI", 3.141592653589793); // Pi z wyższą precyzją

  function trojkat($a, $b)
  {
    if (is_numeric($a) && is_numeric($b) && $a > 0 && $b > 0) {
        $area = ($a * $b) / 2;
        echo "Pole powierzchni trójkąta o podstawie $a oraz wysokości $b wynosi: "
. number_format($area, 2) . "<br>";
    } else {
        echo "Podano nieprawidłowe dane dla trójkąta.<br>";
    }
  }

  function prostokat($c, $d)
  {
    if (is_numeric($c) && is_numeric($d) && $c > 0 && $d > 0) {
        $area = $c * $d;
        echo "Pole powierzchni prostokąta o bokach $c i $d wynosi: " .
number_format($area, 2) . "<br>";
    } else {
        echo "Podano nieprawidłowe dane dla prostokąta.<br>";
    }
  }

  function trapez($e, $f, $h)
  {
    if (is_numeric($e) && is_numeric($f) && is_numeric($h) && $e > 0 && $f > 0 &&
$h > 0) {
        $pole = (($e + $f) / 2) * $h;
        echo "Pole powierzchni trapezu o podstawach $e i $f oraz wysokości $h
wynosi: " . number_format($pole, 2) . "<br>";
    } else {
        echo "Podano nieprawidłowe dane dla trapezu.<br>";
    }
  }

  function kolo($r)
  {
```

```
    if (is_numeric($r) && $r > 0) {
        $p = PI * $r * $r;
        echo "Pole powierzchni koła o promieniu $r wynosi: " . number_format($p, 2)
. "<br>";
    } else {
        echo "Podano nieprawidłowe dane dla koła.<br>";
    }
  }
?>

<!-- // treść głównego skryptu, np. figury.php -->
<?php
  // dołączenie zewnętrznego pliku ze zdefiniowanymi funkcjami
  include("funkcje.php");
?>
<!DOCTYPE HTML>
<html>
<head>
  <title>Funkcje w języku PHP</title>
  <meta charset="utf-8" />
  <style>
    table, td, th {
      margin: auto;
      border: 2px blue double;
      border-collapse: collapse;
    }
    th, .figura {
      background: #003366;
      height: 50px;
      color: white;
    }
      td {
        width: 150px;
      }
  </style>
</head>
<body>
  <table>
    <tr>
      <th><h2>Nazwa figury</h2></th>
      <th><h2>Rysunek</h2></th>
      <th><h2>Przykładowe pole</h2></th>
    </tr>
    <tr>
      <td class="figura">TRÓJKĄT</td>
      <td><img src="trojkat.png" alt="Trójkąt"></td>
      <td>
        <?php trojkat(4,7); ?>
      </td>
    </tr>
    <tr>
      <td class="figura">PROSTOKĄT</td>
      <td><img src="prostokat.png" alt="Prostokąt"></td>
      <td>
        <?php prostokat(6,9); ?>
      </td>
    </tr>
    <tr>
      <td class="figura">TRAPEZ</td>
      <td><img src="trapez.png" alt="Trapez"></td>
      <td>
        <?php trapez(6,9,4); ?>
      </td>
    </tr>
    <tr>
      <td class="figura">KOŁO</td>
      <td><img src="kolo.png" alt="Koło"></td>
      <td style="width: 250px;">
        <?php kolo(3); ?>
```

```
        </td>
      </tr>
    </table>
  </body>
  </html>
```

Uwaga!

<!--?php ... ?--> - takie znaczniki wyświetlają się w Inspektorze przeglądarki Firefox, gdy skrypt PHP nie działa, a wskazują, że kod wewnątrz nich jest kodem PHP i powinien być przetworzony przez serwer WWW. Znaczniki pojawiają się, gdy plik ma rozszerzenie html i serwer niepoprawnie interpretuje kod PHP (pomimo poprawnego zapisu **<?php ... ?>**).

Wykonaj zadanie:

Utwórz skrypt, w którym zdefiniujesz funkcję o nazwie pierwsza(). Funkcja ma sprawdzić, czy podana liczba jest liczbą pierwszą i wyświetlić odpowiedni komunikat w oknie przeglądarki. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_pierwsza.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz skrypt, w którym dołączysz z zewnętrznego pliku **funkcje.php** zdefiniowane dwie funkcje: pitagoras() oraz pole(). Pierwsza z nich ma sprawdzić, czy trzy podane długości odcinków tworzą trójkąt prostokątny. Druga natomiast ma obliczyć pole powierzchni trójkąta. Zadbaj o wyświetlenie pełnych komunikatów nt. zadania oraz o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_funkcje.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Do obliczenia pola powierzchni wykorzystaj wzór Herona:

```
P = sqrt(p(p-a)(p-b)(p-c))
```

gdzie

P – pole powierzchni trójkąta,
p – połowa obwodu trójkąta (p = (a+b+c)/2),
a, b, c – boki trójkąta.

Temat: Pobieranie danych z formularza w języku PHP.

Do pobierania danych z przeglądarki internetowej do serwera wykorzystujemy formularze HTML, które wykorzystują skrypty PHP do przesłania danych. Formularze umieszcza się między znacznikami otwierającym `<form>` i kończącym `</form>`.

Przykład wykorzystania formularza:

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
  <meta charset="utf-8">
</head>
<body>
  <div id="instrukcja" align="center"><h1>Skrypt pobierający dane z
formularza.</h1><br><br></div>
  <div id="tytul" align="left"><h1>Kasa biletowa online:</h1></div>
  <form action="oblicz.php" method="post" name="zamowienie">
    <p><b>Podaj liczbę biletów normalnych (30 zł/szt):</b><br>
    <input type="text" name="normalne">
    <p><b>Podaj liczbę biletów ulgowych (20 zł/szt):</b><br>
    <input type="text" name="ulgowe">
    <p><input type="submit" name="przycisk" value="Zamów"></p>
  </form>
</body>
</html>
```

Wyjaśnienie do powyższego pliku:

- atrybut `ACTION` wskazuje, że dane zostaną przesłane do skryptu `oblicz.php`,
- dostępne metody wysyłania danych: `GET` – wykorzystywana do przesyłanie niewielkiej liczby parametrów, dane przekazywane są za pomocą adresu URL strony, dane są zapisywane w tablicy `$_GET`, `POST` – wykorzystuje nagłówek strony, umożliwia przekazywanie większej liczby parametrów, które nie są widoczne w oknie przeglądarki, dane zapisywane są w tablicy `$_POST`,
- odwołania do poszczególnych pól formularza mają postać: `$nazwa_zmiennej=$_GET['nazwa_pola'];` lub `$nazwa_zmiennej=$_POST['nazwa_pola'];`

Przykładowa treść skryptu `oblicz.php` do powyższego przykładu:

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
  <meta charset="utf-8">
  <style>
    #container
    {
        width: 550px;
        height: 150px;
        border: 2px solid #440000;
        text-align: center;
        background-color: #cc9966;
        margin-left: auto;
      margin-right: auto;
    }
    #blok
      {
        width: 350px;
        float: left;
      }
      img
      {
        width: 200px;
        height: 150px;
      }
      #obraz
      {
        float: left;
      }
  </style>
</head>
<body>
  <div    id="instrukcja"    align="center"><h1>Skrypt    pobierający    dane    z
formularza.</h1><br><br></div>
  <div id="container">
  <div id="blok">
  <?php
    echo "<p>Dokonałeś następującego zamówienia: </p>";
    $norm=$_POST['normalne'];       // liczba biletów normalnych
    $ulg=$_POST['ulgowe'];          // liczba biletów ulgowych
    echo "<ul><li>bilety normalne: $norm sztuki</li>";
    echo "<li>bilety ulgowe: $ulg sztuki</li></ul>";
      $calosc=$norm*30+$ulg*20;
      echo "<h4>Koszt Twojego zamówienia wynosi: $calosc zł.</h4>";
  ?>
  </div>
  <div id="obraz"><img src="logo.png"></div>
  </div>
</body>
</html>
```

Inny przykład zastosowania formularza z listą rozwijalną (zgłoszenie.php):

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
```

```
    <meta charset="utf-8">
    <style>
      table
      {
        width: 100%;
        height: 300px;
        border: 2px solid #440000;
        background-image: url("tlo.jpg");
        margin: auto;
      }
      input
      {
        background: #7777770;
        color: white;
        font-family: Verdana;
        height: 40px;
      }
      #formularz
      {
        width: 400px;
        height: 300px;
        margin: auto;
      }
      #result
      {
        width: 400px;
        height: 200px;
        margin: auto;
        border: 2px groove #666655;
        text-align: center;
      }
      h3
      {
        color: blue;
      }
      h2
      {
        text-align: center;
      }
    </style>
  </head>
  <body>
    <div id="instrukcja" align="center"><h1>Skrypt pobierający dane z formularza wraz
  z listą wyboru.</h1><br><br></div>
    <div id="formularz">
      <form action="" method="post" name="zgloszenie">
        <table>
          <tr><td colspan="2"><h2>Formularz zgłoszeniowy</h2></td></tr>
          <tr>
            <td>Podaj imię:</td>
            <td><input type="text" name="imie"></td>
          </tr>
          <tr>
            <td>Podaj nazwisko:</td>
            <td><input type="text" name="nazwisko"></td>
          </tr>
          <tr>
            <td>Wybierz wykształcenie:</td>
            <td>
              <select name="wiedza">
                <option>wyższe</option>
                <option>średnie</option>
                <option>podstawowe</option>
              </select>
            </td>
          </tr>
          <tr>
            <td>Adres e-mail:</td>
            <td><input type="email" name="mail"></td>
```

```
            </tr>
            <tr>
              <td colspan="2" style="text-align: right;">
                <input type="submit" name="przycisk" value="Zampisz uczestnika"></td>
            </tr>
            </table>
          </form>
        </div>
        <div id="result">
        <?php
          if (isset($_POST["age"]))
          {
            $imie=$_POST['imie'];
            $nazwisko=$_POST['nazwisko'];
            $wiedza=$_POST['wiedza'];
            $mail=$_POST['mail'];
            echo "<p>Zapisaliśmy użytkownika o następujących danych: </p>";
            echo "<h3>$imie $nazwisko</h3>";
            echo "<h4>Wykształcenie: $wiedza</h4>";
            echo "<h4>E-mail: $mail</h4>";
          }
        ?>
        </div>
      </body>
      </html>
```

Walidacja formularza może zostać przeprowadzona za pomocą następujących funkcji:

- `isSet()` – sprawdzającej, czy dane pole zostało wypełnione, zwraca `TRUE` lub `FALSE`,
- `empty()` – sprawdzającej, czy wprowadzona zmienna jest pusta, zwraca `TRUE` lub `FALSE`,
- `is_array()` – sprawdzającej, czy podana dana jest tablicą, zwraca `TRUE` lub `FALSE`,
- `is_bool()` – sprawdzającej, czy podana dana jest typu logicznego, zwraca `TRUE` lub `FALSE`,
- `is_double()` – sprawdzającej, czy podana dana jest liczbą zmiennoprzecinkową, zwraca `TRUE` lub `FALSE`,
- `is_float()` – sprawdzającej, czy podana dana jest liczbą zmiennoprzecinkową, zwraca `TRUE` lub `FALSE`,
- `is_int()` – sprawdzającej, czy podana dana jest liczbą całkowitą, zwraca `TRUE` lub `FALSE`,
- `is_null()` – sprawdzającej, czy podana dana jest typu `NULL`, zwraca `TRUE` lub `FALSE`,
- `is_numeric()` – sprawdzającej, czy podana dana jest liczbą, zwraca `TRUE` lub `FALSE`.

Przykładowa treść funkcji dokonujących walidacji:

```
<?php
  if (isset($_POST["imie"]))
  {
    echo "Masz na imię ".$_POST['imie'];
  }
  if      (!empty($_POST['nazwisko'])      &&      !empty($_POST['wiedza'])      &&
!empty($_POST['email']))
  {
    echo "Wpisałeś wszystkie dane.";
  }
  else
  {
    echo "Nie wprowadzono wszystkich danych.";
  }
?>
```

Przykład zmodyfikowanego skryptu `oblicz.php` sprawdzającego, czy podane dane są liczbami:

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
  <meta charset="utf-8">
  <style>
    #container {
        width: 550px;
        height: 150px;
```

```
            border: 2px solid #440000;
            text-align: center;
            background-color: #cc9966;
            margin-left: auto;
            margin-right: auto;
        }
        #blok {
            width: 350px;
            float: left;
        }
        img {
            width: 200px;
            height: 150px;
        }
        #obraz {
            float: left;
        }
    </style>
</head>
<body>
    <div    id="instrukcja"    align="center"><h1>Skrypt    pobierający    dane    z
formularza.</h1><br><br></div>
    <div id="container">
    <div id="blok">
    <?php
      // sprawdzenie, czy zostały wypełnione pola
      if (isset($_POST['normalne']) && isset($_POST['ulgowe'])) {
        // sprawdzenie, czy podane wartości są liczbami
        if (is_numeric($_POST['normalne']) && is_numeric['ulgowe'])) {
          echo "<p>Dokonałeś następującego zamówienia: </p>";
          $norm=$_POST['normalne'];   // liczba biletów normalnych
          $ulg=$_POST['ulgowe'];            // liczba biletów ulgowych
          echo "<ul><li>bilety normalne: $norm sztuki</li>";
          echo "<li>bilety ulgowe: $ulg sztuki</li></ul>";
          $calosc=$norm*30+$ulg*20;
          echo "<h4>Koszt Twojego zamówienia wynosi: $calosc zł.</h4>";
        }
      }
    ?>
    </div>
    <div id="obraz"><img src="logo.png"></div>
    </div>
</body>
</html>
```

Wykonaj zadanie:

Utwórz stronę internetową, w której zastosujesz wymienione w serwisie w3schools typy danych stosowane w formularzach (https://www.w3schools.com/html/html_form_input_types.asp) oraz dodatkowo listę rozwijalną (`<select>`). Dla każdego typu zastosuj etykietę, która opisze określony typ pola input, przedstaw kod HTML danego inputu oraz podaj wartości przykładowe lub domyślne jeżeli będzie to możliwe w wyglądzie inputu na stronie. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_formularz.html** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz aplikację internetową, która za pomocą formularza umożliwi zakup zeszytów gładkich (2 zł/szt.), zeszytów w linie (2,5 zł/szt.) oraz zeszytów w kratkę (2,7 zł/szt.). Aplikacja ma zawierać skrypt działający po stronie serwera oraz ma wyświetlać podsumowanie dokonanego zakupu. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_zeszyty.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz aplikację internetową, która za pomocą formularza pobierze podaną przez użytkownika kwotę i za pomocą pola typu radio pozwoli wybrać rozdaj paliwa (benzyna 98, benzyna 95, olej napędowy),

a następnie za pomocą przycisku **Oblicz** wyświetli ilość paliwa dostępnego za tę kwotę za granicą (np. w Niemczech) i w Polsce. Aplikacja ma zawierać skrypt działający po stronie serwera. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_paliwo.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz aplikację internetową, która za pomocą formularza pobierze podane przez użytkownika dane rejestracyjne do dowolnego serwisu. W formularzu zastosuj minimum sześć różnych typów inputów. Przeprowadź pełną walidację wszystkich pól po stronie serwera, a ewentualne pola z błędami wyświetl w czerwonej ramce lub z czerwonym tekstem etykiety. Po poprawnym wprowadzeniu danych przez użytkownika i przesłaniu danych z formularza do serwera metodą POST wyświetl na ekranie potwierdzenie rejestracji z zapisanymi na serwerze wartościami. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_rejestracja.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Pliki cookies w języku PHP.

Pliki cookies (tzw. ciasteczka) to tekstowe porcje danych, które są przechowywane w przeglądarce użytkownika na lokalnym komputerze a przesyłane do serwera zdalnego w postaci nagłówka http o nazwie Set-Cookie. Pliki te służą głównie do identyfikacji użytkownika. Jedno ciasteczko to jedna nazwa przechowująca jedną wartość (ciąg znaków). Obecnie pliki cookies przechowywane są lokalnie w pliku bazy danych, np. w przeglądarce Firefox jest to plik `cookies.sqlite` zapisany w profilu użytkownika. Aby utworzyć pliki cookies w skrypcie PHP, należy zastosować następującą funkcję:

```
setcookie($nazwa, $wart, $czas, $sciezka, $domena, $bezp);
```

Poszczególne argumenty funkcji `setcookie()` oznaczają:

- `$nazwa` – nazwa pliku ciasteczka, argument wymagany (pozostałe argumenty są opcjonalne),
- `$wart` – wartość przechowywana w pliku cookie,
- `$czas` – znacznik czasu określający wygaśnięcie pliku cookie,
- `$sciezka` – określa ścieżkę dostępu na serwerze, do której zostanie ograniczona dostępność cookie,
- `$domena` – określa adres, dla którego cookie jest dostępne,
- `$bezp` – wartość `TRUE` wymusza bezpieczne przesłanie cookie przez protokół **`https`**, natomiast wartość `FALSE` (wartość domyślna) pozwala na wysyłanie cookie przez zwykłe połączenie **`http`**.

Funkcję `setcookie()` należy wywołać zanim jakiekolwiek dane zostaną przesłane do przeglądarki, dlatego nalży umieścić ją przed znacznikiem `<html>`, np.:

```
<?php
  setcookie('moje');
?>
<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>Ciasteczka w PHP</title>
</head>
<body>
</body>
</html>
```

Pliki cookies umieszczane są na serwerze w tablicy **`$_cookie`**. Obiekty tablicy indeksowane są poprzez nazwy plików, do których dostęp uzyskujemy następującym poleceniem:

```
$_COOKIE['nazwa_pliku'];
```

Przykład skryptu PHP odczytującego dane z pliku cookie:

```
<?php
  // setcookie('moje');       // sprawdź działanie, gdy odkomentujesz
  if (!isset($_COOKIE['moje']))
  {
    setcookie('moje','ciastko1', time()+5, "/", "localhost", false);
```

```
    $info="Ciasteczko nie jest ustawione.";
  }
  else
  {
    $info="Ciasteczko jest ustawione i ma wartość: " . $_COOKIE['moje'];
  }
?>
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Ciasteczka w PHP</title>
</head>
<body>
  <?php
    echo $info;
  ?>
</body>
</html>
```

Wyjaśnienie do powyższego skryptu:

- funkcja `isset()` sprawdza, czy do serwera został przesłany plik moje, jeżeli `FALSE`, to funkcja `setcookie()` ustawi cookie o nazwie `'moje'` i wartości `'ciastko1'`,
- zmienna `$info` przechowuje informację (komunikat) nt. ciasteczka.

W celu usunięcia plików cookies należy zamknąć okno przeglądarki lub wykonać następujące polecenia:

```
setcookie('moje','ciastko1',time()-100);
setcookie('moje','');
setcookie("moje", FALSE);
```

Przykład zastosowania pliku cookie przechowującego datę ostatniej wizyty strony. Utworzenie pliku cookie:

```
<?php
  $data_waznosci = time() + 2592000;      // 30 dni
  setcookie("odwiedziny", date("F jS - g:ia"), $data_waznosci);
?>
```

Sprawdzenie daty odwiedzin z pliku cookie "odwiedziny":

```
<?php
  if (isset($_COOKIE['odwiedziny'])) {
      $data_odwiedzin = $_COOKIE['odwiedziny'];
      echo "Witamy ponownie! <br> Ostatni raz odwiedziłeś nas: " . $data_odwiedzin;
  } else {
      echo "Witamy na naszej stronie!";
  }
?>
```

Wykonaj zadanie:

Utwórz skrypt w języku PHP, który za pomocą formularza pobierze podane przez użytkownika imię i nazwisko oraz zapisze je w plikach cookies. Jeżeli wprowadzone dane zostaną powtórzone, to powinien pojawić się stosowny komunikat na stronie.

Wykonaj zadanie:

Rozbuduj skrypt z poprzedniego zadania tak, aby dodatkowo za pomocą formularza została pobrana data urodzin użytkownika, a skrypt powinien wyświetlać informację, za ile dni użytkownik będzie obchodził urodziny.

Temat: Sesje w języku PHP.

Mechanizm sesji pozwala na śledzenie działań użytkownika na stronie. Informacje danej sesji przechowywane są na serwerze w tablicy `$_SESSION`. Funkcja `session_start()` rozpoczyna daną sesję, natomiast funkcja `session_destroy()` służy do jej zakończenia. Przykład wykorzystania powyższych funkcji:

```
<?php
  session_start();
  echo "Identyfikator bieżącej sesji: <b>" . session_id() . "</b>";
```

```
  session_destroy();
?>
```

W celu zapisania zmiennej danej sesji należy zastosować polecenie:

```
$_SESSION['nazwa_zmiennej']=wartość;
```

W celu sprawdzenia zmiennej danej sesji należy zastosować polecenie:

```
isset($_SESSION['nazwa_zmiennej']);
```

W celu usunięcia zmiennej przed zakończeniem danej sesji należy zastosować polecenie:

```
unset($_SESSION['nazwa_zmiennej']);
```

Przykład:

```
<?php
  session_start();
  if (!isset($_SESSION['liczba'])) {
    $_SESSION['liczba']=1;
  } else {
    $_SESSION['liczba']=$_SESSION['liczba']+1;
    echo "<br>Odwiedziłeś naszą stronę: " . $_SESSION['liczba'];
  }
?>
```

Przykład zastosowania połączenia śledzenia sesji:

Plik `logowanie.php` zawiera procedury autoryzacji użytkownika i formularz logowania. Plik `wyloguj.php` zawiera procedury wylogowania. Strona po zalogowaniu zapisana jest w pliku `glowna.php`. W zmiennej sesyjnej `$log` powinna zostać zapisana nazwa zalogowanego użytkownika.

```
<?doctype html>
```

Przykładowa zawartość pliku `logowanie.php`:

```
<?php
  session_start();
  if (isset($_SESSION['log'])) {
      header('location: glowna.php');
      exit();
  } elseif (isset($_POST['nazwa']) && isset($_POST['haslo'])) {
      if ($_POST['nazwa'] == 'tomek' && $_POST['haslo'] == 'haselko') {
            $_SESSION['log'] = $_POST['nazwa'];
            header('location: glowna.php');
            exit();
      } else {
            echo "Nieprawidłowe dane logowania<br>";
      }
  }
?>
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <title>Session</title>
  <style>
      .container {
            background: #ccc;
            padding: 20px;
            padding-bottom: 50px;
            border: 2px #777 double;
            border-collapse: collapse;
            border-radius: 12px;
            box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
            width: 60%;
            heigh: auto;
            color: #000;
            text-align: center;
            font-size: 18px;
            margin: 20px 10px 20px 10px;
            margin-left:auto;
```

```
            margin-right:auto;
      }
      label {
            font-size: 20px;
            font-weight: bold;
      }
      #log {
            font-weight: bold;
            font-size: 14pt;
      }
      table {
            width: 30%;
            margin-left:auto;
            margin-right:auto;
      }
      td {
            text-align: left;
      }
  </style>
</head>
<body>
<div class="container">
      <label>Skrypt, który zawiera autoryzację użytkownika z wykorzystaniem
sesji</label>
      <form action="logowanie.php" method="post">
            <p id="log"> Zaloguj się do strony</p>
            <table>
                  <tr><td>Login:</td><td><input type="text" name="nazwa" value=""
size="25"></td></tr>
                  <tr><td>Hasło:</td><td><input type="password" name="haslo"
value="" size="25"></td></tr>
                  <tr><td></td><td><input type="submit" value="Zaloguj
się"></td></tr>
            </table>
      </form>
</div>
</body>
</html>
```

Przykładowa zawartość pliku `glowna.php`:

```
<?php
      session_start();
      if (!isset($_SESSION['log'])) {
            header('location: logowanie.php');
            exit();
      }
?>
<!DOCTYPE html>
<html lang="pl">
<head>
      <meta charset="utf-8" />
      <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
      <title>Session</title>
  <style>
      .container {
            background: #ccc;
            padding: 20px;
            padding-bottom: 50px;
            border: 2px #777 double;
            border-collapse: collapse;
            border-radius: 12px;
            box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
            width: 60%;
            heigh: auto;
            color: #000;
            text-align: center;
            font-size: 18px;
            margin: 20px 10px 20px 10px;
            margin-left:auto;
            margin-right:auto;
```

```
        }
        label {
                font-size: 20px;
                font-weight: bold;
        }
        #log {
                font-weight: bold;
                font-size: 14pt;
        }
        table {
                width: 30%;
                margin-left:auto;
                margin-right:auto;
        }
        td {
                text-align: center;
        }
    </style>
</head>
<body>
<div class="container">
        <label>
                Skrypt, który zawiera autoryzację użytkownika z wykorzystaniem sesji<br>
                Jesteś na stronie głównej.<br>
        </label>
        <?php
                $imie = ucfirst($_SESSION['log']);
                echo "Witaj " . $imie;
        ?>
        <form action="wyloguj.php" method="post">
                <p id="log">Przed opuszczeniem strony wyloguj się!</p>
                <table>
                        <tr><td><input type="submit" value="Wyloguj"></td></tr>
                </table>
        </form>
</div>
</body>
</html>
```

Przykładowa zawartość pliku `wyloguj.php`:

```
<?php
        session_start();
        if (isset($_SESSION['log'])) {
                unset($_SESSION['log']);
        } else {
                header('location: logowanie.php');
                exit;
        }
        $s = session_destroy();
?>
<!DOCTYPE html>
<html lang="pl">
<head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
        <title>Session</title>
        <style>
                .container {
                        background: #ccc;
                        padding: 20px;
                        padding-bottom: 50px;
                        border: 2px #777 double;
                        border-collapse: collapse;
                        border-radius: 12px;
                        box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
                        width: 60%;
                        heigh: auto;
                        color: #000;
```

```
                        text-align: center;
                        font-size: 18px;
                        margin: 20px 10px 20px 10px;
                        margin-left:auto;
                        margin-right:auto;
                  }
                  label {
                        font-size: 20px;
                        font-weight: bold;
                  }
                  #log {
                        font-weight: bold;
                        font-size: 14pt;
                  }
                  table {
                        width: 30%;
                        margin-left:auto;
                        margin-right:auto;
                  }
                  td {
                        text-align: center;
                  }
      </style>
</head>
<body>
<div class="container">
      <label>Skrypt, który zawiera autoryzację użytkownika z wykorzystaniem
sesji</label>
      <form action="logowanie.php" method="post">
            <p id="log">Wylogowałeś się ze strony.</p>
            <table>
                  <tr><td><input type="submit" value="Logowanie"></td></tr>
            </table>
      </form>
</div>
</body>
</html>
```

Temat: Obsługa plików w języku PHP.

Obsługa plików w języku PHP odbywa się w trzech etapach:
- otwarcie pliku istniejącego lub utworzenie nowego,
- wprowadzenie danych do pliku,
- zamknięcie pliku.

W celu sprawdzenia, czy plik istniej należy użyć funkcji:

```
file_exists('nazwa_pliku');
```

W celu otworzenia pliku należy użyć funkcji:

```
fopen(nazwa_pliku, tryb_otwierania);
```

Obsługiwane tryby otwarcia dla funkcji fopen():
- `"r"` – tylko do odczytu, jeśli plik nie istnieje zwracane jest FALSE,
- `"r+"` – plik do odczytu i zapisu, jeśli plik nie istnieje zwracane jest FALSE,
- `"w"` – plik do zapisu, zawartość pliku jest wyzerowana,
- `"w+"` – plik do odczytu i zapisu, zawartość pliku jest wyzerowana,
- `"a"` – plik do zapisu, dopisuje na końcu pliku, tworzy plik, gdy nie istnieje,
- `"a+"` – plik do odczytu i zapisu, dopisuje na końcu pliku, tworzy plik, gdy nie istnieje,
- "x" - zapis, tworzy nowy plik, jeśli istnieje zwracany jest błąd.
- "x+" - odczyt i zapis; tworzy nowy plik, jeśli istnieje zwracany jest błąd.

Podstawową funkcją do zapisu danych do pliku jest `fwrite()`.Funkcja `fwrite()` wymaga następujących argumentów:

- zmienną plikową (nazwa pliku i lokalizacja),
- dane przekazywane do pliku (zapis),
- opcjonalnie – maksymalną liczbę wprowadzanych znaków.

Składnia polecenia:

```
fwrite(resource $handle, string $string, int $length = NULL): int|false
```

Funkcja zapisuje podany ciąg znaków (`$string`) do pliku, a opcjonalnie można określić maksymalną liczbę bajtów do zapisania (`$length`). Funkcja zwraca liczbę zapisanych bajtów lub false w przypadku błędu.

Przykład zastosowania funkcji obsługi plików w języku PHP:

```
<?php
  $file = fopen("example.txt", "w");
  fwrite($file, "Hello, world!"); //zapisuje tekst
  fclose($file);
?>
```

Inny przykład zastosowania funkcji obsługi plików w języku PHP:

```
<?php
  if (file_exists('wizytowka.txt')) {
      echo "Podany plik został znaleziony.";
  } else {
      echo "Tworzony jest plik wizytowka.txt.";
      $plik=fopen("wizytowka.txt", "a");
      $dane="Jan Kowalski";
      fwrite($plik, $dane);
      fclose($plik);
  }
?>
```

Funkcją równoważną (alias) do `fwrite()` jest funkcja `fputs()`. Po zakończeniu działań na pliku należy go zamknąć za pomocą funkcji `fclose()`.

Przykład zastosowania funkcji `fputs()` w języku PHP:

```
<?php
  $file = fopen("example.txt", "w");
  fputs($file, "Hello, world!"); //to samo jak fwrite
  fclose($file);
?>
```

Przykład wpisania liczb losowych do pliku:

```
<?php
  $plik=fopen("liczby.txt","w");
  for ($i=1;$i<=6;$i++) {
    $tab[$i]=rand(1,49);
    fwrite($plik, $tab[$i] . "\n");
  }
  fclose($plik);
?>
```

Przykład zastosowania zapisu danych do pliku:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Zarządzanie plikami w języku PHP</title>
    <style>
            .container {
                  background: #ccc;
                  padding: 20px;
                  border: 2px #777 double;
                  border-collapse: collapse;
                  border-radius: 12px;
```

```
                        box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
                        width: 60%;
                        heigh: auto;
                        color: #000;
                        text-align: center;
                        font-size: 18px;
                        margin: 20px 10px 20px 10px;
                        margin-left:auto;
                        margin-right:auto;
                }
                label {
                        font-size: 20px;
                        font-weight: bold;
                }
        </style>
</head>
<body>
<div class="container">
      <label>Skrypt tworzący plik i zapisujący do niego dane.</label>
      <br>
      <?php
            if (file_exists('wizytowka.txt')) {
                  echo "<br>Podany plik 'wizytowka.txt' już istnieje.";
            } else {
                  $plik=fopen("wizytowka.txt","a");
                  $dane="Jan Kowalski";
                  fwrite($plik,$dane);
                  fclose($plik);
                  echo "<br>Utworzono plik wizytowka.txt.";
            }
      ?>
</div>
</body>
</html>
```

Inny przykład zastosowania zapisu danych do pliku:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Trójkąt Pascala z pobieraniem</title>
    <style>
        .row {
            text-align: center;
            font-family: monospace;
            margin: 5px 0;
        }
        .even {
            background-color: #cce5ff;
        }
        .prime {
            background-color: #ffcccc;
        }
        .cell {
            display: inline-block;
            padding: 5px 10px;
            margin: 2px;
            border-radius: 4px;
        }
            .container {
                  background: #ccc;
                  padding: 20px;
                  border: 2px #777 double;
                  border-collapse: collapse;
                  border-radius: 12px;
                  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
                  width: 60%;
                  heigh: auto;
```

```
                color: #000;
                text-align: center;
                font-size: 18px;
                margin: 20px 10px 20px 10px;
                margin-left:auto;
                margin-right:auto;
            }
            label {
                font-size: 20px;
                font-weight: bold;
            }
    </style>
</head>
<body>
  <div class="container">
      <label>Generuj trójkąt Pascala</label>
    <form method="post">
        <label for="rows">Liczba wierszy:</label>
        <input type="number" name="rows" id="rows" min="1" max="30" required>
        <input type="submit" value="Generuj i pobierz">
    </form>
    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $rows = intval($_POST["rows"]);
        $filename = "trojkat_pascala.txt";
        $output = "";
        echo "<h3>Trójkąt Pascala ($rows wierszy)</h3>";
        for ($i = 0; $i < $rows; $i++) {
            echo "<div class='row'>";
            //for ($space = 0; $space < $rows - $i; $space++) {
            //    echo "  ";
            //}
            $number = 1;
            for ($j = 0; $j <= $i; $j++) {
                $class = "cell";
                if ($number % 2 == 0) {
                    $class .= " even";
                }
                // Sprawdzenie liczby pierwszej
                $isPrime = true;
                if ($number < 2) {
                    $isPrime = false;
                } else {
                    for ($k = 2; $k <= sqrt($number); $k++) {
                        if ($number % $k == 0) {
                            $isPrime = false;
                            break;
                        }
                    }
                }
                if ($isPrime) {
                    $class .= " prime";
                }
                echo "<span class='$class'>$number</span>";
                $output .= $number . " ";
                $number = $number * ($i - $j) / ($j + 1);
            }
            echo "</div>";
            $output .= "\n";
        }
            if (file_exists('trojkat_pascala.txt')) {
                echo "<br>Podany plik 'trojkat_pascala.txt' już istnieje.";
            } else {
                echo "<br>Utworzono plik trojkat_pascala.txt.";
                $plik=fopen("trojkat_pascala.txt","a");
                fwrite($plik,$output);
                fclose($plik);
            }
        }
```

```
    ?>
  <div>
</body>
</html>
```

Temat: Odczytywanie zawartości plików w języku PHP.

Odczytywanie zawartości pliku odbywa się w kilku etapach:
- otwarcie pliku,
- odczytanie danych z pliku,
- zamknięcie pliku.

Do odczytania zawartości pliku można użyć jednej z następujących funkcji:
- `fread()` – odczytanie określonej liczby znaków,
- `fgets()` – odczytanie pierwszej linii danych z pliku,
- `fgetc()` – dane odczytywane po znaku.

Przykład zastosowania funkcji odczytu zawartośći pliku w języku PHP:

```
<?php
  if (file_exists('wizytowka.txt')) {
      echo "<br>Podany plik 'wizytowka.txt' już istnieje.";
      $plik=fopen("wizytowka.txt","r");
      echo "<br>Zawartość pliku: " . fgets($plik);
      fclose($plik);
  } else {
      $plik=fopen("wizytowka.txt", "a");
      $dane="Jan Kowalski";
      fwrite($plik, $dane);
      fclose($plik);
      echo "<br>Utworzono plik wizytowka.txt.";
  }
?>
```

Przykłady zastosowania powyższych funkcji odczytu zawartośći pliku w języku PHP:

```
<?php
  $plik=fopen("wizytowka.txt","r");
  $tresc=fread($plik,4);
  echo $tresc;
  fclose($plik);
?>

<?php
  $plik=fopen("liczby.txt","r");
  $i=0;
  while (!foef($plik)) {
    $liczba[$i]=fgets($plik);
    echo $liczba[$i]."<br>";
    $i++;
  }
  fclose($plik);
?>

<?php
  $plik=fopen("liczby.txt","r");
  $i=0;
  while (!foef($plik)) {
    $liczba[$i]=fgetc($plik);
    echo $liczba[$i]."<br>";  // każda cyfra zostanie wyświetlona w odrębnym wierszu
    $i++;
  }
  fclose($plik); // ?
?>
```

Inny przykład zastosowania zapisu oraz odczytu danych z pliku:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Zarządzanie plikami w języku PHP</title>
    <style>
      .container {
            background: #ccc;
            padding: 20px;
            border: 2px #777 double;
            border-collapse: collapse;
            border-radius: 12px;
            box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
            width: 60%;
            heigh: auto;
            color: #000;
            text-align: center;
            font-size: 18px;
            margin: 20px 10px 20px 10px;
            margin-left:auto;
            margin-right:auto;
      }
      label {
            font-size: 20px;
            font-weight: bold;
      }
    </style>
</head>
<body>
<div class="container">
      <label>Skrypt tworzący plik i zapisujący do niego dane.</label>
      <br>
      <?php
            if (file_exists('wizytowka.txt')) {
                  echo "<br>Podany plik 'wizytowka.txt' już istnieje.";
                  $plik=fopen("wizytowka.txt","r");
                  echo "<br>Zawartość pliku: " . fgets($plik);
                  fclose($plik);
            } else {
                  $plik=fopen("wizytowka.txt","a");
                  $dane="Jan Kowalski";
                  fwrite($plik,$dane);
                  fclose($plik);
                  echo "<br>Utworzono plik wizytowka.txt.";
            }
      ?>
</div>
</body>
</html>
```

Inne funkcje związane z obsługą plików:

- `filesize('nazwa_pliku')` – odczytanie wielkości pliku w bajtach,
- `touch('nazwa_pliku')` – uworzenie pliku,
- `unlink('nazwa_pliku')` – usunięcie pliku,
- `feof('nazwa_pliku')` – uworzenie pliku,
- `readfile('nazwa_pliku')` – odczyt danych z pliku, funkcja zwraca liczbę odczytanych bajtów,
- `file_get_contents('nazwa_pliku')` – odczyt danych z pliku jako ciąg tekstowy,
- `file('nazwa_pliku')` – odczyt danych z pliku jako ciąg tekstowy,
- `mkdir('nazwa_katalogu')` – utworzenie katalogu,
- `rmdir('nazwa_katalogu')` – usunięcie katalogu,
- `opendir('nazwa_katalogu')` – otworzenie katalogu,
- `readdir('nazwa_katalogu')` – odczytanie zawartości katalogu,
- `closedir('nazwa_katalogu')` – zamknięcie katalogu (bez nazwy zostanie zamknięty ostatni katalog),
- `FILE_IGNORE_NEW_LINES` – ignoruje znak końca linii,

- `FILE_IGNORE_NEW_LINES` – ignoruje znak końca linii,
- `FILE_SKIP_EMPTY_LINES` – ignoruje puste linie,
- `FILE_USE_INCLUDE_PATH` – należy podać nazwy katalogów, które zostaną przeszukane,
- `exit('komunikat')` – natychmiastowe zakończenie wykonywania skryptu,
- `die('komunikat')` – natychmiastowe zakończenie wykonywania skryptu.

Wykonaj zadanie:

Utwórz skrypt, który wylosuje 15 liczb z przedziału <1,50> i zapisze je do pliku o nazwie dane.txt. Następnie pobierz te dane i wyświetl w oknie przeglądarki tylko liczby parzyste oraz oblicz ich sumę, a wynik zapisz w nowym pliku o nazwie wynik.txt. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_liczby.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz skrypt, który otworzy istniejący plik o nazwie osoby.txt, w którym są zapisane imiona i nazwiska pięciu osób. Pobierz te dane do tablicy asocjacyjnej, posortuj według nazwisk, a następnie uporządkowane zapisz do pliku o nazwie posortowane.txt. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_osoby.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Napisz skrypt w języku PHP, który wygeneruje sześć zestawów po sześć liczb losowych z zakresu od 1 do 49 i zapisze te liczby w pliku o nazwie lotto.txt. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_plik_lotto.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Przykładowe rozwiązanie zadania:

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Zarządzanie plikami w języku PHP</title>
    <style>
      .container {
            background: #ccc;
            padding: 20px;
            border: 2px #777 double;
            border-collapse: collapse;
            border-radius: 12px;
            box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
            width: 60%;
            heigh: auto;
            color: #000;
            text-align: center;
            font-size: 18px;
            margin: 20px 10px 20px 10px;
            margin-left:auto;
            margin-right:auto;
      }
      label {
            font-size: 20px;
            font-weight: bold;
      }
    </style>
</head>
<body>
<div class="container">
      <label>Skrypt generujący 6 zestawów liczb losowych z zakresu od 1 do
49.</label>
      <br>
      <?php
      if (file_exists('lotto.txt')) {
```

```
                echo "<br>Podany plik 'lotto.txt' już istnieje.";
                $plik = fopen("lotto.txt","r");
                echo "<br>Zawartość pliku:<br>";
                while (!feof($plik)) {
                        echo fgets($plik) . "<br>";
                }
                fclose($plik);
        } else {
                $plik = fopen("lotto.txt","a");
                $allNumbers = range(1, 49);  //tworzy tablicę
                for ($i=1; $i<=6; $i++) {
                        shuffle($allNumbers);         //miesza tablicę
                        $tab = array_slice($allNumbers, 0, 6);   //tworzy nową tablicę z
sześciu pierwszych elementów
                        sort($tab);                          //sortuje tablicę
                        $dane = "Zestaw $i: " . implode(" ", $tab) . "\n";
                        fwrite($plik, $dane);
                }
                fclose($plik);
                echo "<br>Utworzono plik lotto.txt.";
        }
        ?>
    </div>
    </body>
    </html>
```

Wykonaj zadanie:

Zmodyfikuj powyższy skrypt w języku PHP tak, aby użytkownik mógł zdecydować ile wygenerować zestawów oraz czy nadpisać lub dodać te liczby do pliku o nazwie lotto.txt. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_plik_lotto.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Temat: Połączenie z bazą danych za pomocą języka PHP.

Bazy danych stanowią zbiór danych dla aplikacji internetowych. Umożliwiają pobieranie informacji, które są prezentowane użytkownikowi strony internetowej. Do obsługi baz danych wykorzystuje się systemy zarządzania bazami danych. Najpopularniejszym systemem zarządzania bazami danych jest MySQL.

W języku PHP współpraca z bazami MySQL może być realizowana poprzez:
- funkcje MySQLi,
- bibliotekę PDO (PHP Data Objects).

Etapy komunikacji języka PHP z MSQL:
- nawiązanie połączenia z MySQL i wybór bazy danych,
- utworzenie zapytania i jego wykonanie,
- odebranie rezultatów i wyświetlenie ich na stronie internetowej,
- rozłączenie z MySQL.

Do połączenia z bazą danych służy funkcja `mysqli_connect()`. Jeżeli przed funkcją użyjemy znaku @, to nie zostanie wyświetlone ostrzeżenie wygenerowane przez PHP. Funkcja posiada następujące argumenty:
- nazwę hosta – IP lub nazwa domenowa serwera przechowującego bazę danych ewentualnie z numerem portu usługi serwera (domyślnie 3306),
- nazwę użytkownika (domyślnie root),
- hasło użytkownika (domyślnie puste),
- nazwę bazy danych.

Przykład wywołania nawiązania i zakończenia połączenia w języku PHP:

```
$polaczenie=mysqli_connect('localhost', 'klient','','sklep');
mysqli_close($polaczenie);    //zwraca TRUE, gdy powodzenie
```

lub

```
$mysqli->close();
```

Przykład zastosowania połączenia z bazą danych:

```
<?php
  $polaczenie=@mysqli_connect('localhost', 'root', '', 'sklep');
  if (!$polaczenie) {
    exit ("Błąd połączenia z serwerem danych.");
  } else {
    echo "Połączyłeś się z bazą danych sklepu internetowego.";
  }
  mysqli_close($polaczenie);
?>
```

Inny przykład zastosowania połączenia z bazą danych:

```
<?php
  $serwer = 'localhost:3306';
  $login = 'root';
  $haslo = ' ';
  $baza = 'sklep';
  $polaczenie = mysqli_connect($serwer , $login, $haslo, $baza);
  if (mysqli_connect_errno()) {
    die ("Błąd połączenia z serwerem danych: " . mysqli_connect_error());
  } else {
    echo "Połączyłeś się z bazą danych sklepu internetowego.<br>";
  }
  mysqli_close($polaczenie);
?>
```

Temat: Wysyłanie zapytań do bazy danych i ich wyświetlanie w języku PHP.

Po nawiązaniu połączenia z wybraną bazą danych można wysyłać do niej zapytania, np. w celu odczytania danch. Zapytania realizowane są za pomocą funkcji `mysqli_query()` zapisanej w postaci:

```
mysqli_query(identyfikator.polaczenia, 'zapytanie'); // zwraca TRUE lub FALSE
mysqli_query(identyfikator.polaczenia, 'zapytanie', stan_wyniku);
```

Argument `stan_wyniku` dla funkcji `mysqli_query()` stosuje się opcjonalnie, a określa on sposób przetwarzania wyniku (domyślnie MYSQLI_STORE_RESULT). Jeżeli w wyniku zapytania pobrano dane z serwera (np. poleceniem `SELECT`), to funkcja `mysqli_query()` zwraca obiekt `mysqli_result`. W przypadku, gdy zapytanie nie pobierało danych (np. użyto polecenia `INSERT`), funkcja zwraca wartość `true`. Natomiast gdy wykonanie zapytania nie powiodło się, funkcja zwraca wartość `false`.

Natomiast do odczytu danych z bazy stosujemy np. następujące funkcje:

```
mysqli_fetch_array(identyfikator.polaczenia);
mysqli_num_rows(identyfikator.polaczenia);      // zwraca liczbę wierszy wyniku odp.
mysqli_affect_rows(identyfikator.polaczenia);   // podaje liczbę zmodyfikowanych rekordów
```

Przykład zapytania do bazy danych w celu pobrania danych:

```
$result = mysqli_query($connect, 'SELECT * FROM uczniowie');
```

lub

```
$sql = ('SELECT * FROM uczniowie');
$result = $mysqli->query($sql);
```

Przykład zapytania do bazy danych w celu dodania danych:

```
$wstaw = mysqli_query($connect, "INSERT INTO uczniowie (imie, nazwisko, login,
haslo) VALUES ('Leon', 'Zawodowy', 'zawodowy', 'haselko')";
```

Przykład zapytania do bazy danych i odczytanie wyniku funkcjami:

```
$result = mysqli_query($connect, 'SELECT * FROM uczniowie');
echo "<br>Liczba rekordów: " . mysqli_num_rows($result);
```

lub

```
echo "<br>Liczba rekordów: " . ($result->num_rows);
```

Przykład zastosowania połączenia z bazą danych oraz odczytania danych:

```
<?php
  $sql = new mysqli('localhost', 'user', 'pass', 'szkola');
  $result = $sql -> query('SELECT * FROM uczniowie');
  while ($row = $result -> fetch_assoc()) {
    echo "<br>Login: " . $row['imie'];
    //echo "<br>Nr: " . $row['id'] . " Login: " . $row['imie'];
  }
  mysqli_close($sql);
?>
```

Inny przykład zastosowania połączenia z bazą danych oraz odczytania danych:

```
<?php
  $sql = new mysqli('localhost', 'user', 'pass', 'szkola');
  $result = $sql -> query('SELECT * FROM uczniowie');
  while ($rekord = $result -> fetch_array()) {
      print "".$rekord[0]." - ".$rekord[5]." - ".$rekord[10]."<br>";
  }
  mysqli_close($polaczenie);
?>
```

Inny przykład zastosowania połączenia z bazą danych i odczytania danych:

```
<?php
  $polaczenie=@mysqli_connect('localhost', 'klient','','szkola')
  if (!$polaczenie) {
    exit "Błąd połączenia z serwerem danych.";
  } else {
    $pytanie=mysqli_query($polaczenie, 'SELECT nazwisko,imie,login FROM uczniowie
WHERE nazwisko like "A%" ORDER by nazwisko');
    echo "<ol>";
    while ($result=mysqli_fetch_array($pyt)) { // tablica asocjacyjna
      echo "<li>".$result['nazwisko']." ".$result['imie']." - ".$result['login']."
</li>;
    }
    echo "</ol>";
  }
  mysqli_close($polaczenie);
?>
```

Przykład zastosowania połączenia z bazą danych i odczytania danych oraz pętlą `for` odczytującą kolejne wiersze:

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
  <meta charset="utf-8">
  <style>
    table, td, th
    {
      border:2px solid gray;
    }
    th
    {
      background-color: #dddddd;
    }
    th, td
    {
      width: 100px;
    }
  </style>
</head>
<body>
  <div id="instrukcja" align="center"><h1>Skrypt pobierający dane z bazy
danych.</h1><br><br></div>
  <?php
    $polaczenie=@mysqli_connect('localhost', 'klient','','sklep')
    if (!$polaczenie)
    {
```

```
      exit "Błąd połączenia z serwerem danych.";
    }
    else
    {
      $pytanie=mysqli_query($polaczenie, 'SELECT nazwisko,imie,data_urodzenie FROM
pracownicy WHERE zawod="informatyk" and plec="k"');
      $ile_wierszy=mysqli_num_rows($pytanie)
      echo "<table><tr><th>Imię</th><th>Nazwisko</th><th>Data urodzenia</th>";
      for ($i=0;$i<$ile_wierszy;$i++
      }
        $result=mysqli_fetch_array($zapytanie);
        echo
"<tr><td>".$result['nazwisko']."</td><td>".$result['imie']."</td><td>".$result['dat
a_urodzenia'].</td></tr>;
      }
    echo "</table>";
    }
    mysqli_close($polaczenie);
  ?>
</body>
</html>
```

Przykład zastosowania połączenia z bazą danych i dodania danych:

```
<?php
  $polaczenie=@mysqli_connect('localhost', 'root','','firma')
  if (!$polaczenie)
  {
    exit "Błąd połączenia z serwerem danych.";
  }
  else
  {
    mysqli_query($polaczenie, "INSERT into placówki(id_placowki,nazwa,miasto,adres)
VALUES (7,'ZS9','Koszalin','Jedności 9'), (8,'SP1','Koszalin','Zwycięstwa 131')");
    $ile_dodano=mysqli_affected_rows($polaczenie);   // tablica asocjacyjna
    echo "Liczba dodanych rekordów: ".$ile_dodano;
  }
  mysqli_close($polaczenie);
?>
```

Przykład zastosowania połączenia z bazą danych oraz śledzenia sesji:

Plik `logowanie.php` zawierający formularz logowania oraz procedury autoryzacji (zmienna sesyjna przyjmuje wartość loginu użytkownika i przekierowuje do strony `wyloguj.php` lub przyjmuje wartość 0 i przekierowuje do strony `zarejestruj.php`):

```
<?doctype html>
<html lang="pl">
<?php
  session_start();
  if (isset($_POST['login']) && isset($_POST['haslo']))
  {
    $login=$_POST['login'];
    $haslo=$_POST['haslo'];
    $polaczenie=mysqli_connect('localhost','root','','szkola');
    $z1=mysqli_query($polaczenie,'select login, haslo from uzytkownicy');
    while ($z2=mysqli_fetch_array($z1))
    {
      if ($login==$z2['login'] && $haslo==$z2['haslo'])
      {
        $_SESSION['zalogowany']=$z2['login'];
        header('location: wyloguj.php');
        exit();
      }
      else
      {
        $_SESSION['zalogowany']=0;
        header('location: zarejestruj.php');
      }
    }
```

```
  }
?>
<head>
  <meta charset="utf-8">
  <title>Sesje w języku PHP</title>
</head>
<body>
  <br><b>Zaloguj się na stronę.</b>
  <form action="" method="POST">
    <br>Login: <input type="text" name="login">
    <br>Hasło: <input type="password" name="haslo">
    <br><input type="submit" value="Zaloguj">
  </form>
</body>
</html>
```

Plik `wyloguj.php` umożliwiający powrót do strony logowania jeżeli nie istnieje zmienna sesyjna:

```
<?doctype html>
<html lang="pl">
<head>
  <meta charset="utf-8">
  <title>Sesje w języku PHP</title>
</head>
<body>
<?php
  session_start();
  if (isset($_SESSION['zalogowany']))
  {
    echo "Jesteś zalogowany jako: ".$_SESSION['zalogowany'];
  }
  session_destroy();
?>
  <a href="logowanie.php">Powrót do strony logowania</a>
</body>
</html>
```

Plik `zarejestruj.php` umożliwiający zarejestrowanie nowego użytkownika:

```
<?doctype html>
  <?php
    session_start();
    if (isset($_POST['imie']) && isset($_POST['nazwisko']) && $_POST['login'] &&
isset($_POST['haslo']))
    {
      $imie=$_POST['imie'];
      $nazwisko=$_POST['nazwisko'];
      $login=$_POST['login'];
      $haslo=$_POST['haslo'];
      $_SESSION['rejestracja']=1;
      $polaczenie=mysqli_connect('localhost','root','','szkola');
      $z1=mysqli_query($polaczenie,"insert into uzytkownicy (imie, nazwisko, login,
        haslo) values ('$imie','$nazwisko','$login','$haslo')");
      header('location: logowanie.php');
      exit();
    }
  ?>
<html lang="pl">
<head>
  <meta charset="utf-8">
  <title>Sesje w języku PHP</title>
</head>
<body>
  <b>Zarejestruj się na stronie.</b>
  <form action="" method="POST">
    <br>Imię: <input type="text" name="imie">
    <br>Nazwisko: <input type="text" name="nazwisko">
    <br>Login: <input type="text" name="login">
    <br>Hasło: <input type="password" name="haslo">
    <br><input type="submit" value="Zarejestruj się">
  </form>
</body>
```

```
</html>
```

Wykonaj zadanie:

Do powyższych skryptów dodaj do pliku `zaloguj.php` przycisk, który przekieruje użytkownika do pliku `zarejestruj.php`, oraz utwórz skrypt `pokaz.php`, który wyświetli listę zarejestrowanych użytkowników w lokalnej bazie `szkola` w tabeli `uczniowie`. Następnie dodaj do skryptu `pokaz.php` możliwość wyboru przez użytkownika informacji wyświetlanych na stronie wg następującej listy:

- liczba wszystkich użytkowników zarejestrowanch w bazie,
- wykaz użytkowników, których imię to Jan,
- wykaz użytkowników, których nzwisko zaczyna się na literę K,
- wykaz tylko identyfikatorów pominiętych (brakujących),
- wykaz tylko identyfikatorów i loginów wszystkich użytkowników,
- wykaz wszystkich informacji o wszystkich użytkownikach.

Zastosuj odpowiednie komunikaty dl aużytkownika i zadbaj o estetykę strony. Wszystkie utworzone w zadaniu pliki zapisz w katalogu o nazwie **$nazwisko_$klasa_$gr_mysql**, następnie skompresuj i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Wykonaj zadanie:

Utwórz aplikację internetową, która za pomocą formularza logowania umożliwi użytkownikowi (login: **Dyr2026**, hasło: **quertyiuop!**) na dostęp do danych bazy **szkola**. Po zalogowaniu powinna wyświetlić się strona z listą uczniów (imię i nazwisko) oraz ich ocenami z poszczególnych przedmiotów. Dane zapisane są w bazie o nazwie **szkola**. Zastosuj odpowiednie komunikaty i zadbaj o estetykę strony. Pracę zapisz w pliku pod nazwą **$nazwisko_$klasa_$gr_login.php** i prześlij do nauczyciela w postaci załącznika na adres greszata@zs9elektronik.pl.

Ćwiczenie:

Utworzymy aplikację internetową, która wyświetli informacje o ofercie **hurtowni owoców** (str. 423). Strona wyświetli dane firm korzystających z hurtowni. Przykładowy kod źródłowy strony hurtowni:

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
  <title>Skrypty w języku PHP</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
  <div   id="instrukcja"   align="center"><h1>Strona   wyświetlająca   dane   firm
korzystających z hurtowni.</h1><br><br></div>
  <section id="container">
  <section id="baner">
    Hurtownia owoców
  </section>
  <section id="prawy">
    <?php
      $polaczenie=@mysqli_connect('localhost', 'root','','skupow');
        $z1=mysqli_query($polaczenie,'select * from firma');
        echo "<p>Firmy korzystające z hurtowni</p>";
        while ($row=mysqli_fetch_array($z1))
      {
        echo              "<b>".$row['Nazwa'].'</b>,              '.$row['Miasto'].'
',.$row['Adres'].'<br>';;
      }
        $z2=mysqli_query($polaczenie,'Select  Nazwa,  ilosc,  Data  zamowienia  from
owoce,  zamowieni  where  IdFirmy=1  and  owoce.IdOwocu=zamowienie.IdOwocu  order  by
Nazwa');
        echo "<p>Zamówienia firmy JaśKowalski</p>";
        echo    '<table><tr><th>Nazwa    owocu</th><th>Ilość    w    kg</th><th>Data
zamówienia</th></tr>';
        while ($row=mysqli_fetch_array($z2))
        {
```

```
            echo
"<tr><td>".$row['Nazwa'].'</td><td>',$row['ilosc'].'</td><td>'.$row['Datazamowienia
'].'</td></tr>';
          }
          echo "</tables>";
        mysqli_close($polaczenie);
      ?>
    </section>
    <section id="galeria">
      <table>
        <tr>
          <td><img src="czeresnia.png" class="obrazki"></td>
          <td><img src="gruszka.png" class="obrazki"></td>
          <td><img src="truskawka.png" class="obrazki"></td>
          <td><img src="sliwka.png" class="obrazki"></td>
          <td><img src="jagoda.png" class="obrazki"></td>
        </tr>
      </table>
    </section>
    </section>
</body>
</html>
```

Przykładowy kod źródłowy pliku opisującego wygląd strony hurtowni (`style.css`):

```
#container
{
      height: 90%;
}
#baner
{
      height: 10%;
      font-size: 50px;
      background-color: #ccff33;
      text-align: center;
      letter-spacing: 15px;
}
#galeria
{
      background-color: #ccff66;
      height: 20%;
}
#prawy
{
      background-color: #fcffbc6;
      height: 70%;
      text-align: center;
}
table
{
      padding-top: 20px;
      text-align: center;
      margin-top: 0px;
      font-weight: bold;
      color: #990000;
}
.obrazki
{
      width: 100px;
      height: 100px;
}
```

Temat: Biblioteka PDO w języku PHP.

Biblioteka PDO (ang. PHP Data Objects) udostępnia metody, które umożliwiają komunikację z bazami danych. W celu połączenia z bazą danych należy stworzyć obiekt klasy PDO o następujących danych:

- rodzaj bazy danych i sposób połączenia, sterownik, adres serwera, nazwa bazy danych, numer portu,
- nazwa użytkownika uprawnionego do połączenia oraz hasło – mogą zostać pominięte,
- dodatkowe informacje związane z połączenia – nie są wymagane, opisują połączenie.

Ogólna składnia wywołania połączenia:

```
PDO(data_source, user, password, options);
```

Instrukcja uproszczona dla MSQL ma następującą postać:

```
mysqli:host=nazwa_serwera;port=numer_portu_uslugi;dbname=nazwa_bazy;
```

Przykład zastosowania połączenia z bazą danych serwera MySQL dostępnym na lokalnym komputerze:

```
<?php
  $polaczenie="mysqli:host=localhost;dbname=firma";
    try
    {
      // Tworzymy nowe połączenie PDO
      $pol=new PDO($polaczenie,"root","");
      echo "Udało się połączyć z serwerem.";
    }
    catch (PDOExpection $x)
    {
      echo 'Błąd połączenia '.$e->getMessage();
      exit;
    }
?>
```

Inny przykład zastosowania połączenia z bazą danych serwera MySQL:

```
<?php
      $host = 'localhost';
      $dbname = 'nazwa_bazy';
      $username = 'użytkownik';
      $password = 'hasło';
      try {
            $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
            // Ustawiamy tryb raportowania błędów
            $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            // Jeśli połączenie się powiedzie, można wykonać dalsze operacje
            echo "Połączenie z bazą danych powiodło się.";
      } catch (PDOException $e) {
            // Jeśli wystąpi błąd (np. brak bazy danych), wyświetlamy komunikat
            echo "Nie można połączyć się z bazą danych: " . $e->getMessage();
      }
?>
```

Po nawiązaniu połączenia zapytanie do bazy danych wykonujemy za pomocą metody `query()`, np:

```
$obiekt_PDO->query("Zapytanie");
```

Odpowiedź z serwera na zapytanie stanowi obiekt klasy `PDOStatement`.

Przykład zastosowania połączenia z bazą danych serwera MySQL oraz wyświetleniem wyników zapytania:

```
<?php
  $polaczenie="mysqli:host=localhost;dbname=firma";
    try
    {
      $pol=new PDO($polaczenie,"root","");
      echo "Udało się połączyć z serwerem.";
      echo "<br>Wykaz zarobków informatyków firmy:<br>";
      $query=$pol->query('SELECT imie, nazwisko, pensja FROM pracownicy WHERE
zawod="informatyk" ORDER BY pensja DESC');
      foreach ($query as $wiersz)
      {
        echo $wiersz['imie']." ".$wiersz['nazwisko']." ".$wiersz['pensja']."<br>";
      }
    }
    catch (PDOExpection $x)
    {
      echo 'Błąd połączenia '.$e->getMessage();
      exit;
    }
```

```
?>
```

Dlaczego PDO jest lepsze niż mysqli?

Cecha	PDO	mysqli
Obsługa wielu baz (MySQL, PostgreSQL, SQLite…)	✓	✗ tylko MySQL
Named parameters (:email)	✓	✗
Łatwiejsze prepared statements	✓	✗
Lepsza obsługa błędów	✓	✗
Kod czytelniejszy	✓	✗

Dlatego większość nowych projektów używa PDO.

Temat: Projektowanie programów.

Proces projektowania oprogramowania może obejmować następujące procesy:
- tworzenie nowego produktu programistycznego na indywidualne zamówienie klienta,
- modernizację bądź rozbudowę istniejących systemów informatycznych lub aplikacji,
- modyfikację istniejących modułów w celu wykorzystania ich do nowych zastosowań.

W każdym procesie tworzenia oprogramowania można wyróżnić następujące etapy:
- planowanie – polegające na zdefiniowaniu problemu, który trzeba rozwiązać, oraz dokładna jego analiza i wybór metody jego rozwiązania,
- implementacja (kodowanie) – polegająca na zapisaniu algorytmu w postaci kodu źródłowego (pomysł (koncepcja) => analiza problemu (struktura obiektowa) => projekt (algorytmy, struktury danych) => wybór języka programowania => kodowanie => optymalizacja kodu (szybkość, rozmiary) => kompilacja => dokumentacja => testowanie => konserwacja),
- kompilacja – zamiana kodu źródłowego na binarny,
- konsolidacja – polegająca na łączeniu plików obiektowych i bibliotek w program wykonywalny (statyczna lub dynamiczna),
- testowanie – polegające na wykrywaniu błędów,
- wdrażanie i optymalizacja – mające na celu ulepszenie programu,
- użytkowanie i ewaluacja.

Projekt oprogramowania to opis struktury oprogramowania, które będzie poddane implementacji, opis danych przetwarzanych przez program oraz opis użytych algorytmów.

Temat: Dokumentowanie tworzonych aplikacji.

Działanie twórcy programu mają na celu:
- tworzeniu dokumentacji technicznej programu,
- tworzeniu dokumentacji obsługi programu dla użytkownika (ogólny opis sytemu, wymagania sprzętowe, instrukcja obsługi programu, instrukcja instalacji i opis ewentualnych problemów, informacje nt. pomocy technicznej, opis wersji programu).

Dokumentację techniczną mogą tworzyć:
- opisy wymagań zleceniodawcy lub napytania przetargowe,
- specyfikacje techniczne (np. zastosowany język programowania), architektura aplikacji,
- modele, schematy, procedury lub algorytmy,
- raporty, umowy, e-maile lub inne formy komunikacji między zleceniodawcą a autorami aplikacji,
- listingi kodu źródłowego wraz z komentarzami,
- repozytorium programu (historia zmian kodu aplikacji co umożliwia śledzenie zmian kodu oraz kto je wprowadził, możliwość cofania zmian, możliwość tworzenia kopii oprogramowania),
- opis procesu testowania (np. raporty z testów).

Ważną funkcję w dokumentowaniu aplikacji odgrywają komentarze. W każdy języku programowania komentarze są ignorowane przez interpreter lub kompilator. Przykłady komentarzy:

- skrypty Windows: `REM` lub `::`,
- skrypty Linux: `#`,
- skrypty C++: `//!` lub `/*! -- */` lub `///` lub `/** -- **/`,
- skrypty Python: `///` lub `#`.

Repozytorium to zapisana historia zmian kodu, która umożliwia śledzenie postępów w tworzeniu aplikacji. Repozytorium oprogramowania to miejsce, w którym przechowuje się, organizuje i udostępnia pliki związane z oprogramowaniem. Może zawierać kod źródłowy, dokumentację, historię zmian, wersje programu, a także pliki binarne gotowe do instalacji. Repozytoria są używane zarówno przez programistów, jak i użytkowników końcowych. Zalety stosowania repozytorium:

- śledzenie pełnej historii zmian pojawiających się w plikach,
- podgląd wcześniejszych zmian i możliwości sprawdzenia, kiedy i kto je utworzył,
- możliwość cofania zmian,
- możliwość sprawnego przeprowadzania testów i kopii bezpieczeństwa.

Dokumentacja obsługi programu komputerowego dla użytkownika powinna być jasna, zrozumiała i kompleksowa. Jej celem jest umożliwienie użytkownikowi efektywnego korzystania z aplikacji bez konieczności posiadania specjalistycznej wiedzy. Oto główne elementy, które powinna zawierać dokumentacja:

1. Wstęp
- Cel programu: Krótkie wyjaśnienie, do czego służy program, w jakim celu został stworzony i jakie problemy rozwiązuje.
- Grupa docelowa: Kto jest głównym użytkownikiem programu, np. zaawansowani użytkownicy, przedsiębiorcy, czy osoby bez doświadczenia technicznego.
- Wymagania systemowe: Informacje o minimalnych wymaganiach sprzętowych i oprogramowania, takich jak system operacyjny, pamięć RAM, procesor, wolne miejsce na dysku, zależności (np. biblioteki zewnętrzne, wersje JDK, .NET itp.).

2. Instalacja i konfiguracja
- Kroki instalacji: Przewodnik krok po kroku, jak zainstalować program na różnych systemach operacyjnych (Windows, macOS, Linux itp.).
- Instrukcje konfiguracji: Jak skonfigurować program po instalacji (np. konfiguracja połączeń z bazą danych, ustawienia użytkownika).
- Wymagane składniki zewnętrzne: Jeśli program wymaga dodatkowych narzędzi (np. bazy danych, serwera webowego), powinna być instrukcja jak je zainstalować i skonfigurować.

3. Pierwsze kroki
- Uruchomienie programu: Jak uruchomić aplikację po zainstalowaniu.
- Interfejs użytkownika: Krótkie wprowadzenie do interfejsu, w tym główne elementy okna aplikacji (menu, paski narzędzi, przyciski, okna dialogowe).
- Podstawowe operacje: Przewodnik po najczęstszych operacjach w programie, np. tworzenie nowego dokumentu, otwieranie pliku, zapisywanie.

4. Szczegółowe funkcje programu
- Opis funkcji i narzędzi: Szczegółowy opis wszystkich funkcji programu, ich przeznaczenia i sposobu działania. Należy uwzględnić wszystkie dostępne opcje w programie.
- Przykłady użycia: Dla każdej funkcji mogą być podane konkretne przykłady, które pomogą użytkownikowi zrozumieć, jak działa dana funkcja.
- Zakładki i ustawienia: Jeżeli program posiada zakładki, ustawienia użytkownika lub opcje konfiguracji, dokumentacja powinna wyjaśniać, jak je dostosować do swoich potrzeb.

5. Porady i triki
- Wskazówki dotyczące wydajności: Jak zoptymalizować używanie programu, np. zmniejszyć zużycie pamięci, przyspieszyć procesy.
- Rozwiązywanie problemów: Najczęstsze problemy i błędy, z jakimi użytkownicy mogą się spotkać, oraz jak je rozwiązać.
- Krótkie skróty klawiaturowe: Lista skrótów klawiaturowych ułatwiających pracę z programem.

6. FAQ (Najczęściej zadawane pytania)
- Problemy związane z instalacją: Jak poradzić sobie z problemami instalacyjnymi (np. komunikaty o błędach, problemy z zależnościami).

- Problemy podczas użytkowania: Jak rozwiązywać problemy z funkcjami programu (np. aplikacja się zawiesza, nie można otworzyć pliku).
- Bezpieczeństwo: Jak dbać o bezpieczeństwo danych, np. w przypadku aplikacji online, jak przechowywać dane w chmurze.

7. Aktualizacje i wsparcie
- Proces aktualizacji: Jak regularnie aktualizować program, aby zachować zgodność z najnowszymi wersjami i poprawkami bezpieczeństwa.
- Kanały wsparcia: Gdzie użytkownicy mogą zgłaszać problemy lub uzyskać pomoc, np. forum, adres e-mail wsparcia technicznego, link do bazy wiedzy.
- Zmiany w wersjach: Lista zmian w nowych wersjach (changelog), aby użytkownik wiedział, co zostało poprawione, dodane lub usunięte.

8. Zabezpieczenia i prywatność
- Polityka prywatności: Jakie dane program zbiera od użytkownika, jak są one wykorzystywane, i jakie mają zabezpieczenia.
- Bezpieczne przechowywanie danych: Jeśli program przechowuje wrażliwe dane (np. hasła, dane osobowe), powinien zawierać wskazówki, jak zapewnić ich bezpieczeństwo.
- Szyfrowanie i bezpieczeństwo połączeń: Jeśli program komunikuje się przez Internet (np. aplikacje chmurowe), jak zapewnia bezpieczeństwo tych połączeń (np. szyfrowanie SSL/TLS).

9. Słownik terminów
- Definicje i wyjaśnienia terminów technicznych: Dla użytkowników, którzy nie są obeznani z terminologią, przydatne mogą być definicje techniczne używanych w programie pojęć.

10. Załączniki
- Dodatkowe materiały: Instrukcje, pliki konfiguracyjne, przykłady danych, które mogą pomóc w pełni wykorzystać program.
- Licencje i informacje prawne: Wszelkie istotne informacje o licencji programu, zgodności z prawem, użyciu bibliotek zewnętrznych.

Format dokumentacji:
- Dokumentacja powinna być dostępna w różnych formatach, w zależności od potrzeb:
- Online: Interaktywna dokumentacja dostępna na stronie internetowej (może zawierać filmy instruktażowe, FAQ, itp.).
- Offline: PDF, e-book, plik do pobrania lub drukowana wersja.

Dodatkowe zasoby:
- Instrukcje wideo: Często użytkownicy preferują wizualne przedstawienie procesu, dlatego warto dodać tutoriale wideo, które krok po kroku pokażą, jak korzystać z programu.
- Wsparcie społeczności: Fora, grupy dyskusyjne, czy chaty z zespołem wsparcia technicznego.

Wykonaj zadanie:

Zapoznaj się z informacjami dostępnymi w sieci Internet na temat oprogramowania do tworzenia dokumentacji technicznej **Doxygen**. Sporządź krótką notatkę w zeszycie nt. tego programu.

Temat: Przykład realizacji prostej aplikacji.

UWAGA!

W razie problemów kieruj pytania do nauczyciela na adres greszata@zs9elektronik.pl.

Materiał tworzony między innymi na podstawie podręcznika wydawnictwa WSiP dla kwalifikacji zawodu technik informatyk INF.03 p.t. "Tworzenie stron i aplikacji internetowych oraz baz danych i administrowanie nimi" autorstwa Agnieszki i Tomasza Klekot.