PHP - podstawy

1. Podstawy składni języka PHP

- każda instrukcja (nie blok instrukcji) kończy się średnikiem
- wielkość liter ma znaczenie tylko w nazwach zmiennych
- nazwy mogą składać się z liter, cyfr i znaku podkreślenia (_)
- nazwy nie mogą zaczynać się od cyfry i nie mogą zawierać spacji

2. Dodawanie skryptu PHP w kodzie HTML

```
<?php
 // tu wstaw kod PHP
?>
```

3. Komentarz jednolinijkowy

```
// ta linia zostanie zignorowana
# ta linia również zostanie zignorowana
```

4. Komentarz wielolinijkowy

```
/* te linie
    zostaną
    zignorowane */
```

5. Blok instrukcji

```
{
 instrukcja1;
 instrukcja2;
}
```

6. Umieszczanie tekstu na stronie

```
echo "<h1>Tytuł</h1>";          // umieszcza tekst na stronie
echo "<h1>", "Tytuł", "</h1>";  // poszczególne napisy można oddzielić przecinkiem
print "<b>napis</b>";           // alternatywny sposób umieszczania tekstu
```

7. Wartości

```
5           // liczba całkowita (typ integer)
-5          // ujemna liczba całkowita (typ integer)
3.14        // liczba rzeczywista (typ float)
"napis"     // tekst (typ string)
'$napis$'   // tekst, inny zapis, można umieszczać znaki specjalne (typ string)
true        // wartość logiczna, prawda (typ boolean)
false       // wartość logiczna, fałsz (typ boolean)
null        // wartość "pusta" (typ specjalny)
```

8. Zmienne (definicja za pomocą znaku dolara $ i znaku równości =)

```
$kod  = "PHP";    // definicja zmiennej o nazwie $kod i przypisanie do niej napisu "PHP"
$liczba = 10;     // zmienna $liczba przechowuje liczbę całkowitą 10
$pi = 3.14;       // zmienna $pi przechowuje liczbę rzeczywistą 3,14
$log = true;      // zmienna $log przechowuje wartość logiczną true (prawda)
```

9. Stałe

```
define("PI", 3.1415);             // definicja stałej o nazwie "PI" i wartości 3,1415
define("EMAIL", "poczta@a.pl");   // definicja stałej "EMAIL" o wartości "poczta@a.pl"
```

10. Operatory arytmetyczne

```
$a = 1 + 2;        // dodawanie, zmienna $a przechowuje wartość 3
$b = 5 – 3;        // odejmowanie, zmienna $b przechowuje wartość 2
$c = 2 * 5;        // mnożenie, zmienna $c przechowuje wartość 10
$d = 4 / 2;        // dzielenie, zmienna $d przechowuje wartość 2
$e = 5 % 2;        // reszta z dzielenia, zmienna $e przechowuje wartość 1
$f = (2 + 3) * 4;  // grupowanie (nawias), najpierw dodaj, potem pomnóż, wynik 20
$g = 2 ** 4;       // potęgowanie: 2^4, zmienna $g przechowuje wartość 16
```

11. Operator konkatenacji (łączenia, sklejania napisów) – kropka

```
$a = "Tekst" . "<br>";    // zmienna $a przechowuje napis "Tekst<br>"
```

12. Inkrementacja (zwiększanie o 1)

```
$a++;       // zwraca $a, a następnie
            // zwiększa $a o 1

// przykład:
$b = 5;
echo $b++;  // wypisze: 5
echo $b;    // wypisze: 6

++$c;       // zwiększa $c o 1,
            // a następnie zwraca $c

// przykład:
$d = 5;
echo ++$d;  // wypisze: 6
echo $d;    // wypisze: 6
```

13. Dekrementacja (zmniejszanie o 1)

```
$a--;       // zwraca $a, a następnie
            // zmniejsza $a o 1

// przykład:
$b = 5;
echo $b--;  // wypisze: 5
echo $b;    // wypisze: 4

--$c;       // zmniejsza $c o 1,
            // a następnie zwraca $c

// przykład:
$d = 5;
echo --$d;  // wypisze: 4
echo $d;    // wypisze: 4
```

14. Operatory przypisania

```
$a = 10;  // do zmiennej $a zostanie przypisana wartość 10
$a += 2;  // do zmiennej $a zostanie dodane 2, wynik 12
$a -= 4;  // od zmiennej $a zostanie odjęte 4, wynik 8
$a *= 2;  // zmienna $a zostanie pomnożona przez 2, wynik 16
$a /= 4;  // zmienna $a zostanie podzielona przez 4, wynik 4
$a %= 3;  // zmienna $a przyjmie wartość reszty z dzielenia $a przez 3, wynik 1

$a .= "koniec";  // do zmiennej $a na końcu zostanie dodany napis "koniec"
```

15. Przypisywanie długich napisów do zmiennej - heredoc

```
$napis = "NAPIS";
$tekst = <<<"TX"
Bardzo długi $napis
TX;
echo $tekst; //wypisze: Bardzo długi NAPIS
```

16. Przypisywanie długich napisów do zmiennej - nowdoc

```
$napis = "NAPIS";
$tekst = <<<'TX'
Bardzo długi $napis
TX;
echo $tekst;  //wypisze: Bardzo długi $napis
```

17. Operatory porównania

```
$a == $b;  // równy, prawda jeśli $a jest równe $b
$a != $b;  // różny, prawda jeśli $a jest różne od $b
$a >  $b;  // większy, prawda jeśli $a jest większe od $b
$a <  $b;  // mniejszy, prawda jeśli $a jest mniejsze od $b
$a >= $b;  // większy lub równy, prawda jeśli $a jest większe lub równe od $b
$a <= $b;  // mniejszy lub równy, prawda jeśli $a jest mniejsze lub równe od $b

$a === $b; // identyczny, prawda jeśli $a równe $b i są tego samego typu
$a !== $b; // nieidentyczny, prawda jeśli $a nie jest równe $b lub nie są tego
           // samego typu
```

18. Operatory logiczne

```
$a && $b;           // AND (i - koniunkcja), zwróci prawdę gdy $a i $b są prawdą
$a || $b;           // OR (lub - alternatywa), zwróci prawdę gdy $a lub $b jest prawdą
! $a;               // NOT (nie - negacja), zwróci prawdę, jeśli $a nie jest prawdą
                    // NOT (nie - negacja), zwróci fałsz, jeśli $a jest prawdą
```

19. Operator tłumienia błędów (@)

```
$x = (5 / 0);       // wyświetli BŁĄD, dzielenie przez 0 !!!
$x = @(5 / 0);      // nadal błąd, ale bez wyświetlanie komunikatu
```

20. Tablice indeksowane numerycznie (typ array)

```
// definicja pustej tablicy (2 sposoby):
$pusta = [];
$pusta2 = array();

// definicja tablicy:
$auta = ["Audi", "Opel", "Kia"];

// dostęp do elementów tablicy:
echo $auta[0];  // wypisze: Audi
echo $auta[1];  // wypisze: Opel
echo $auta[2];  // wypisze: Kia



// dodanie elementu na końcu tablicy:
$auta[] = "Ford"

// sposób alternatywny definicji tablicy:
$auta = array("Audi", "Opel", "Kia");

// zliczanie elementów tablicy:
echo count($auta);   // wypisze: 3
```

21. Tablice asocjacyjne (typ array)

```
// definicja tablicy:
$osoba = ["imie"     => "Jan",
          "nazwisko" => "Kowalski",
          "email"    => "poczta@a.pl"];

// dostęp do elementów tablicy:
echo $osoba["imie"];  //wypisze: Jan
echo $osoba["nazwisko"]; //wypisze: Kowalski
echo $osoba["email"]; //wypisze: poczta@a.pl

// dodanie elementu do tablicy:
$osoba["wiek"] = 18;

// sposób alternatywny definicji tablicy:
$osoba = array("imie"     => "Jan,
               "nazwisko  => "Kowalski",
               "email"    => "poczta@a.pl");
```

22. Operator warunkowy

```
warunek ? wartość1 : wartość2;
// jeżeli warunek jest prawdziwy
// operator zwraca wartość1
// jeżeli warunek jest fałszywy
// operator zwraca wartość2

// przykład:
$x = 10;
$wynik = ($x < 0) ? "ujemna" : "dodatnia";
echo "Wartość zmiennej x jest $wynik";
// wypisze: Wartość zmiennej x jest dodatnia
```

23. Zmiana typu zmiennej

```
// tymczasowa zmiana typu zmiennej:
(int) $x;    // rzutowanie do typu integer
(float) $x;  // rzutowanie do typu float
(string) $x; // rzutowanie do typu string
(bool) $x;   // rzutowanie do typu boolean

// trwała zmiana typu zmiennej:
settype($x, "int");     // zmiana na integer
settype($x, "float");   // zmiana na float
settype($x, "string");  // zmiana na string
settype($x, "bool");    // zmiana na boolean
```

24. Instrukcja warunkowa if

```
if (wyrażenie) {
  // kod do wykonania, jeżeli
  // wyrażenie jest prawdziwe
}

// przykład:
$x = 5;
if ($x == 5) {   // wykonaj,
  echo "Pięć";   // jeżeli $x równe 5
}
// wypisze: Pięć
```

25. Instrukcja warunkowa if … else

```
if (wyrażenie) {
  // kod do wykonania, jeżeli
  // wyrażenie jest prawdziwe
}
else {
  // kod do wykonania, jeżeli
  // wyrażenie nie jest prawdziwe
}

// przykład:
$x = 5;
if ($x == 5) {   // wykonaj,
  echo "Pięć";   // jeżeli $x równe 5
}
else {                // wykonaj, jeżeli
  echo "Inna liczba"; // $x różne od 5
}
// wypisze: Inna liczba
```

26. Instrukcja warunkowa if … elseif … else

```
if (wyrażenie1) {
  // instrukcje do wykonania, jeżeli wyrażenie1 jest prawdziwe
}
elseif (wyrażenie2) {
  // instrukcje do wykonania, jeżeli wyrażenie2 jest prawdziwe,
  // i jednocześnie wyrażenie1 jest fałszywe
}
else {
  // instrukcje do wykonania, jeżeli wyrażenie1 i wyrażenie2 nie są prawdziwe
}

// przykład:
$x = 10;
if ($x == 5) {          // wykonaj, jeżeli $x równe 5
  echo "Pięć";
}
elseif ($x == 10) {     // wykonaj, jeżeli $x równe 10 i jednocześnie różne od 5
  echo "Dziesięć"
}
else {                  // wykonaj, jeżeli $x różne od 5 i od 10
  echo "Inna liczba";
}
// wypisze: Dziesięć
```

27. Instrukcja wyboru switch

```
switch ($zmienna) {

  case  wartość1:
    // instrukcje do wykonania, jeśli
    // $zmienna równa wartość1
    break;

  case wartość2:
    // instrukcje do wykonania, jeśli
    // $zmienna równa wartość2
    break;
}

// przykład:
$a = 1;
switch ($a) {
  case 0:
    echo "Zero";
    break;
  case 1:
    echo "Jeden";
    break;
}
// wypisze: Jeden
```

28. Instrukcja wyboru switch … default

```
switch ($zmienna) {

  case wartość1:
    // instrukcje do wykonania, jeśli
    // $zmienna równa wartość1
    break;

  case wartość2:
    // instrukcje do wykonania, jeśli
    // $zmienna równa wartość2
    break;

  default:
   // wykonaj, jeśli $zmienna będzie
   // różna od wartości1 i wartości2
}

// przykład:
$a = "arbuz"
switch ($a) {
  case "pomidor":
     echo "Pomidor";
     break;
  case "cebula":
     echo "Cebula";
     break;
  default:
     echo "Inne warzywo";
}
// wypisze: Inne warzywo
```

29. Pętla for

```
for (wyrażenie1; warunek; wyrażenie2) {
      // instrukcje pętli
}
//   wyrażenie1 - instrukcje do wykonania przed rozpoczęciem pętli
//   warunek    - jeżeli prawdziwy kolejne przejście pętli zostanie wykonane
//   wyrażenie2 - instrukcje do wykonania po każdym przejściu (iteracji) pętli

// przykład:
for ($i = 0; $i < 10; $i++) {
  echo $i." ";
}
// wypisze: 0 1 2 3 4 5 6 7 8 9
```

30. Pętla while

```
while (wyrażenie) {
   // instrukcje do wykonania
   // w każdym przebiegu pętli,
   // jeżeli wyrażenie jest prawdziwe
}

// przykład:
$i = 0;
while ($i < 10) {
  echo $i." ";
  $i++;
}
// wypisze: 0 1 2 3 4 5 6 7 8 9
```

31. Pętla do … while

```
do { // początek pętli
  // instrukcje pętli
} while (wyrażenie); //jeżeli wyrażenie jest
                     //prawdziwe skocz na
                     //na początek pętli

// przykład:
$i = 0;
do {
  echo $i." ";
  $i++;
} while ($i < 10);
// wypisze: 0 1 2 3 4 5 6 7 8 9
```

32. Instrukcja break

```
for ($i = 0; $i < 10; $i++) {
  if ($i == 5) {  // jeżeli $i równe 5
    break;        // przerwij pętle
  }
  echo $i." ";
}
// wypisze: 0 1 2 3 4
```

33. Instrukcja continue

```
for ($i = 0; $i < 10; $i++) {
  if ($i == 5) {  // jeżeli $i równe 5
    continue;     // skocz do następnego
                  // przebiegu pętli
  }
  echo $i." ";
}
// wypisze: 0 1 2 3 4 6 7 8 9
```

34. Pętla foreach – przeglądanie tablic numerowanych

```
foreach ($tablica as $element) {
   // instrukcje do wykonania
   // na każdym elemencie tablicy
   // Sposób działania:
   // za każdym przejściu pętli, pobierany
   // jest kolejny element tablicy $tablica
   // i przypisywany do zmiennej $element
}

// przykład:
$lata = [1999, 2001, 2002];
foreach ($lata as $rok) {
  echo $rok." ";
}
// wypisze: 1999 2001 2002
```

35. Pętla foreach – przeglądanie tablic asocjacyjnych

```
foreach ($tablica as $klucz => $wartosc) {
   // instrukcje do wykonania
   // na każdym elemencie tablicy
}

// przykład:
$osoba = ["imię"     => "Jan",
          "nazwisko" => "Nowak",
          "wiek"     => 20];
foreach ($osoba as $nazwa => $wartosc) {
  echo $nazwa.": ".$wartosc.", ";
}
// wypisze:
// imię: Jan, nazwisko: Nowak, wiek: 20,
```

36. Funkcje

```
// definicja prostej funkcji:
function hello() {
    echo "Cześć";
}

// wywołanie funkcji:
hello();   // wypisze: Cześć
```

37. Przekazywanie argumentów do funkcji

```
// definicja funkcji dodającej
// dwie liczby:
function dodaj($a, $b) {
    echo $a + $b; // wypisz sumę $a i $b
}

// wywołanie funkcji:
dodaj(4, 2);   // wypisze: 6
dodaj(2, 1);   // wypisze: 3
```

38. Zwracanie wartości przez funkcję - return

```
// definicja funkcji dzielącej
// jedną liczbę przez drugą
function podziel($a, $b) {
    return $a / $b; // zwróć iloraz $a i $b
}

// wywołanie funkcji:
echo podziel(8, 2);   // wypisze: 4
```

39. Argumenty funkcji przekazywane przez referencję

```
// definicja funkcji z argumentem
// przekazywanym przez referencję
function inkrementujA(&$arg) {
    $arg++;
}

// definicja funkcji z argumentem
// przekazywanym przez wartość
function inkrementujB($arg) {
    $arg++;
}

$liczba = 1;
inkrementujB($liczba);
echo $liczba; // wypisze: 1

inkrementujA($liczba);
echo $liczba; // wypisze: 2
```

40. Funkcje - zmienne globalne

```
$liczba1 = 10;
$liczba2 = 20;

function wyswietl() {

  global $liczba1;

  echo $liczba1; // wypisze: 10
  echo $liczba2; // błąd, $liczba2 nie
                 // jest widoczna w funkcji
}
```

41. Funkcje - zmienne statyczne

```
function inkrementuj() {

  static $liczba1 = 0;
  $liczba2 = 0;

  $liczba1++;
  $liczba2++;

  echo $liczba1." ".$liczba2;
}

inkrementuj();  // wypisze: 1 1
inkrementuj();  // wypisze: 2 1
inkrementuj();  // wypisze: 3 1
```

42. Funkcje - argumenty domyślne

```
function dodaj($x, $y = 1) {
  return $x + $y;
}

echo dodaj(2, 4);  // wypisze: 6
echo dodaj(10);    // wypisze: 11
```

43. Funkcje - zwracanie tablicy

```
function funkcja() {
  // zwróć tablicę trzyelementową:
  return array(1, 2, 3);
}
// przypisz elementy tablicy zwracanej
// przez funkcje() do zmiennych:
// $a = 1, $b = 2, $c = 3
list($a, $b, $c) = funkcja();
```

44. Definicja klasy i obiektu (instancji klasy)

```
// definicja klasy:

class Osoba {
  $imie;       // właściwość
  $nazwisko;   // właściwość

  function wypisz_imie() { // metoda

    //dostęp do składowych z wnętrza klasy:
    echo $this->imie;

  }
}

// definicja (tworzenie) obiektu:
$osoba1 = new Osoba();

// dostęp do właściwości:
$osoba1->imie = "Piotr";

// dostęp do metody:
$osoba1->wypisz_imie(); // wypisze: "Piotr"
```

45. Klasy - modyfikatory dostępu

```
class Osoba {

  //dostęp bez ograniczeń (opcja domyślna):
  public $imie;

  //dostęp tylko z wnętrza danej klasy:
  private $pesel;

  //dostęp tylko z wnętrza danej klasy
  //i z wnętrza klas potomnych:
  protected $wiek;

  public function wypisz_pesel() {

    echo $this->pesel;

  }
}

$osoba1 = new Osoba();

// ok, właściwość imie jest publiczna:
$osoba1->imie = "Piotr";

// błąd, brak dostępu do właściwości pesel
$osoba1->pesel = "01234567890";

// ok, dostęp do właściwości pesel
// przez metodę publiczną:
$osoba1->wypisz_pesel();
```

46. Klasy - konstruktor i destruktor

```
class Klasa {

  // definicja konstruktora, który jest
  // wywoływany podczas tworzenia obiektu
  function __construct() {

    echo "Tworzenie obiektu;

  }

  // definicja destruktora, który jest
  // wywoływany podczas usuwania obiektu
  function __destruct() {

    echo "Usuwanie obiektu";

  }
}

$obiekt = new Klasa();
// wyświetli: Tworzenie obiektu

unset($obiekt);
// wyświetli: "Usuwanie obiektu"
```

47. Dziedziczenie

```
// klasa bazowa:
class Osoba {

  public function funkcjaA() {

    echo "Osoba";

  }
}
// klasa potomna (dziedziczy
// po klasie bazowej):
class Uczen extends Osoba {

  public function funkcjaB() {

    echo "Uczeń";

  }
}


$uczen1 = new Uczen();

$uczen1->funkcjaA(); // wypisze: Osoba
$uczen1->funkcjaB(); // wypisze: Uczeń
```

48. Klasy - stałe

```
class Klasa {

    // definicja stałej:
    const pi = 3.14;
}

// dostęp do stałej:
echo Klasa::pi;       // wyświetli: 3.14

// błąd: próba zmiany stałej
Klasa::pi = 10;
```

49. Klasy - składniki statyczne

```
class Klasa {
    // definicja zmiennej statycznej:
    static $liczba = 1;
    // definicja metody statycznej:
    static function funkcja() {
      // dostęp do właściwości statycznych:
      echo self::$liczba;
    }
}
echo Klasa::$liczba; // wypisze: 1
Klasa::funkcja();  // wypisze: 1
```

50. Otwieranie i zamykanie pliku

```
// próba otworzenia pliku w danym trybie:
$plik = fopen("nazwa_pliku", "TRYB");

// operacje na pliku

// zamknięcie pliku:
fclose($plik);
```

Gdzie **TRYB**:

TRYB	Odczyt	Zapis	Położenie wskaźnika	Usuwa zawartość pliku
"r"	Tak	Nie	początek pliku	Nie
"r+"	Tak	Tak	początek pliku	Nie
"w"	Nie	Tak	początek pliku	Tak *
"w+"	Tak	Tak	początek pliku	Tak *
"a"	Nie	Tak	koniec pliku	Nie *
"a+"	Tak	Tak	koniec pliku	Nie *

* - jeżeli plik nie istnieje, zostanie utworzony nowy

51. Czytanie zawartości pliku

```
$plik = fopen("nazwa_pliku", "r");

// sprawdzenie długości pliku
$dlugosc_pliku = filesize("nazwa_pliku");

// odczyt całej zawartości pliku do
// zmiennej $zawartość
$zawartosc = fread($plik, $dlugosc_pliku);

// wypisanie zawartości pliku
echo $zawartosc;

fclose($plik);
```

52. Zapisywanie danych do pliku

```
$plik = fopen("nazwa_pliku", "w");

$zawartosc = "Przykładowa treść";

// zapis zwartości zmiennej $zawartosc
// do pliku:
fwrite($plik, $zawartosc);


fclose($plik);
```

53. Obsługa plików - podstawowe funkcje

fopen("nazwa_pliku", "TRYB") - otwiera plik o nazwie "nazwa_pliku" w danym trybie. Zwraca uchwyt pliku, lub false w przypadku błędu.

fclose($uchwyt_pliku) - zamyka plik określony przez $uchwyt_pliku. Zwraca true, jeżeli operacja się powiedzie, lub false w przypadku błędu.

fread($uchwyt_pliku, $dlugosc) - czyta z pliku określonego przez $uchwyt_pliku. Maksymalnie zostanie odczytane $dlugosc bajtów. Zwraca odczytaną zawartość lub false w przypadku błędu.

fgets($uchwyt_pliku) - czyta jeden wiersz z pliku określonego przez $uchwyt_pliku. Zwraca odczytaną zawartość lub false w przypadku błędu.

fgetc($uchwyt_pliku) - czyta jeden znak z pliku określonego przez $uchwyt_pliku. Zwraca odczytaną zawartość lub false w przypadku błędu.

file($uchwyt_pliku) - czyta z pliku określonego przez $uchwyt_pliku. Zwraca tablicę, której każdy element odpowiada linii w danym pliku. W przypadku błędu zwraca false.

fwrite($uchwyt_pliku, $napis) - zapisuje treść $napis do pliku określonego przez $uchwyt_pliku. W przypadku błędu zwraca false.

file_exists("nazwa_pliku") - zwraca wartość true jeżeli plik istnieje.

filesize("nazwa_pliku") - zwraca rozmiar pliku w bajtach.

feof($uchwyt_pliku) - sprawdza, czy wskaźnik pliku jest na końcu pliku.

54. Obsługa formularzy - metoda GET - parametry i wartości przekazywane są JAWNIE za pomocą adresu URL do TEJ SAMEJ STRONY Z FORMULARZEM

```
<form method="get">
  <input type="text" name="pole1">
  <input type="submit" name="submit" value="Wyślij zapytanie">
</form>

<?php
   // zmienna $_GET jest tablicą zawierającą wszystkie dane
   // z formularza wypełnionego przez użytkownika

   // funkcja isset sprawdza czy zmienna $_GET["submit"] istnieje:
   if (isset($_GET["submit"])) {
     // przypisanie do zmiennej $pole1 wartość pola formularza o nazwie "pole1"
     $pole1 = $_GET["pole1"];
     // wypisanie zawartości zmiennej $pole1:
     echo $pole1;
   }
 ?>
```

55. Obsługa formularzy - metoda POST - parametry i wartości przekazywane są NIEJAWNIE za pomocą adresu URL do TEJ SAMEJ STRONY Z FORMULARZEM

```
<form method="post">
  <input type="text" name="pole1">
  <input type="submit" name="submit" value="Wyślij zapytanie">
</form>

<?php
   // zmienna $_POST jest tablicą zawierającą wszystkie dane
   // z formularza wypełnionego przez użytkownika

   if (isset($_POST["submit"])) {
     $pole1 = $_POST["pole1"];
     echo $pole1;
   }
 ?>
```

56. Obsługa formularzy - przekazywanie parametrów i wartości do INNEJ STRONY

```
<!-- początek formularza na strona1.php -->
<form method="post" action="strona2.php">
  <input type="text" name="pole1">
  <input type="submit" name="submit" value="Wyślij zapytanie">
</form>
<!-- koniec formularza na strona1.php -->


<!-- początek skryptu na strona2.php -->
<?php
   if (isset($_POST["submit"])) {
     $pole1 = $_POST["pole1"];
     echo $pole1;
   }
 ?>
<!-- koniec skryptu na strona2.php -->
```

57. Nawiązywanie połączenia z bazą danych

```
// funkcja mysqli_connect nawiązuje połączenie z bazą danych, zwraca wartość false
// w przypadku niepowodzenia lub identyfikator połączenia z bazą danych,
// przyjmuje następujące parametry:
// "host" - określa nazwę lub adres IP serwera
// "użytkownik" - określa nazwę użytkownika
// "hasło" - określa hasło użytkownika
// "baza" - określa nazwę bazy danych

$id_polaczenia = mysqli_connect("host", "użytkownik", "hasło", "baza");
if ($id_polaczenia == false) {
   echo "Błąd podczas próby połączenia z serwerem bazy danych";
   exit;
}
```

58. Kończenie połączenia z bazą danych

```
// funkcja mysqli_close kończy połączenie z bazą danych, zwraca wartość false
// w  przypadku niepowodzenia, przyjmuje następujące parametry:
// $id_polaczenia - identyfikator połączenia z bazą danych

$rezultat = mysqli_close($id_polaczenia);
if ($rezultat == false) {
   echo "Błąd podczas zamykania połączenia z serwerem bazy danych";
   exit;
}
```

59. Wysyłanie zapytania do bazy danych

```
// funkcja mysqli_query wysyła zapytanie do bazy danych, zwraca wartość false
// w  przypadku niepowodzenia lub wynik zapytania, przyjmuje następujące parametry:
// $id_polaczenia - identyfikator połączenia z bazą danych
// $zapytanie - treść zapytania
```

```
$zapytanie = "select tytul, rok, dlugosc from film";
$rezultat = mysqli_query($id_polaczenia, $zapytanie);
if ($rezultat == false) {
   echo "Błąd podczas przetwarzania zapytania";
   exit;
}
```

60. Odczytywanie danych zwróconych przez zapytanie

```
// funkcja mysqli_num_rows zwraca liczbę wierszy w wyniku zapytania
// funkcja mysqli_fetch_array zwraca kolejny wiersz z wyniku zapytania
// w postaci tablicy numerycznej i asocjacyjnej


$zapytanie = "select tytul, rok, dlugosc from film";
$rezultat = mysqli_query($id_polaczenia, $zapytanie);
$ile_wierszy = mysqli_num_rows($rezultat);
    for($i = 0; $i < $ile_wierszy; $i++) {
      $wiersz = mysqli_fetch_array($rezultat);
      echo "$wiersz[0] $wiersz[1] $wiersz[2] <br>";
      echo $wiersz['tytul'] . " " . $wiersz['rok'] . " " . $wiersz['dlugosc'] . "<br>";
    }
```

61.