

Aplikacje backendowe - wykład 2



dr Artur Bartoszewski
UTH Radom

Phpinfo()



Funkcja `phpinfo()` w języku programowania PHP służy do wyświetlania szczegółowych informacji na temat konfiguracji serwera i interpretera PHP. Jest to przydatne narzędzie do diagnozowania problemów, monitorowania ustawień PHP oraz uzyskiwania informacji o dostępnych rozszerzeniach PHP.

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

Uwaga: Funkcja `phpinfo()` jest użyteczna do celów diagnostycznych i debugowania, ale należy jej używać ostrożnie. Nie powinna być dostępna publicznie na produkcji, ponieważ wyjawia wiele szczegółów na temat konfiguracji serwera, które mogą być przydatne dla potencjalnych atakujących.

Oto kilka kluczowych punktów dotyczących funkcji `phpinfo()`:

- **Wyświetlanie informacji konfiguracyjnych:** Głównym celem funkcji `phpinfo()` jest wyświetlanie informacji na temat bieżącej konfiguracji PHP. Możesz dowiedzieć się o wersji PHP, kompilatorze, opcjach konfiguracji, rozszerzeniach i wiele innych.
- **Interfejs webowy:** Wywołanie `phpinfo()` powoduje wygenerowanie strony HTML, która zawiera szczegółowe informacje o konfiguracji PHP. Ta strona jest wygodna do przeglądania w przeglądarce internetowej.
- **Rozszerzenia PHP:** Na stronie wygenerowanej przez `phpinfo()` można znaleźć listę dostępnych rozszerzeń PHP oraz informacje o nich, takie jak wersja, status i opis.
- **Konfiguracja PHP:** Możesz uzyskać dostęp do informacji dotyczących ustawień PHP, takich jak maksymalny rozmiar przesyłanych plików, limit czasu wykonania skryptu, obsługiwane protokoły, ścieżki do plików konfiguracyjnych itp.
- **Zabezpieczenia i moduły:** `phpinfo()` wyświetla również informacje o dostępnych modułach serwera (jeśli są dostępne) oraz ustawieniach zabezpieczeń PHP, takich jak tryb bezpieczny (`safe_mode`) i dostępne opcje konfiguracyjne dotyczące bezpieczeństwa.



Język PHP



Definiowanie ciągów znaków



W języku PHP cudzysłowy (podwójne) i apostrofy (pojedyncze) różnią się w kontekście definiowania ciągów znaków (stringów) oraz przetwarzania zmiennych wewnątrz nich.

Cudzysłowy " "

W ciągach znaków zdefiniowanych w cudzysłowach można osadzać zmienne (interpolacja), co oznacza, że zmienne wewnątrz ciągu zostaną zastąpione ich aktualnymi wartościami. Na przykład:

```
<?php
    $zmienna = "świecie";
    echo "Witaj, $zmienna!"; // Wyświetli: "Witaj, świat!"
?>
```

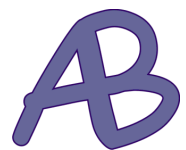
W ciągach znaków w cudzysłowach możesz osadzać specjalne znaki, takie jak znaki nowej linii (\n) i znaki unikowe (\) w celu formatowania tekstu.

Apostrofy ''

Ciągi znaków w apostrofach są traktowane dosłownie, co oznacza, że zmienne osadzone wewnątrz ciągów w apostrofach nie zostaną zinterpretowane jako zmienne, tylko wyświetlone dosłownie. Na przykład:

```
<?php
    $zmienna = "świecie";
    echo 'Witaj, $zmienna!'; // Wyświetli: "Witaj, $zmienna!"
?>
```

W ciągach znaków w apostrofach nie można osadzać znaków specjalnych ani korzystać z interpolacji zmiennych.



Formularz kontaktowy

Imię: Email: Wybierz opcję:



Przykład tworzenia formularza w języku HTML – formularz kontaktowy

```
<h1>Formularz kontaktowy</h1>
<form action="proces.php" method="post">
  <!-- Pole tekstowe -->
  <label for="imie">Imię:</label>
  <input type="text" id="imie" name="imie" required>
  <!-- Pole email -->
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <!-- Pole wyboru -->
  <label for="wybor">Wybierz opcję:</label>
  <select id="wybor" name="opcja">
    <option value="opcja1">Opcja 1</option>
    <option value="opcja2">Opcja 2</option>
    <option value="opcja3">Opcja 3</option>
  </select>
  <!-- Przycisk wysyłający formularz -->
  <input type="submit" value="Wyślij">
</form>
```

<form>: To tag formularza, który zawiera elementy formularza. Atrybut **action** określa, gdzie zostaną przesłane dane formularza, a atrybut **method** definiuje, jak zostaną przesłane (GET lub POST).

<label>: Etykiety opisujące pola formularza. Atrybut **for** łączy etykietę z polem formularza na podstawie atrybutu **id**.

<input>: Elementy wejściowe, takie jak pola tekstowe, pola email i wiele innych. Atrybuty **type** określają rodzaj pola, **id** identyfikuje je, a **name** jest używane do identyfikacji pól w wynikowym formularzu. Atrybut **required** oznacza, że pole jest wymagane.

<select>: Element do tworzenia listy rozwijanej (dropdown) z opcjami wyboru. Poszczególne opcje są definiowane za pomocą znacznika **<option>**.

<input type="submit">: Przycisk służący do wysłania formularza.

Skrypt, który odbierze wartości z formularza z poprzedniego slajdu.
Ten skrypt przetwarza dane z formularza i wyświetla je na ekranie

```
<h1>Dane z formularza</h1>
<?php
    $imie = $_POST["imie"];
    $email = $_POST["email"];
    $opcja = $_POST["opcja"];

    echo "<p>Imię: " . $imie . "</p>";
    echo "<p>Email: " . $email . "</p>";
    echo "<p>Wybrana opcja: " . $opcja . "</p>";
?>
```

Atrybut action



Atrybut action w elemencie <form> w języku HTML określa adres URL, na który zostaną przesłane dane z formularza po jego wysłaniu.

Atrybut action może przyjmować różne wartości:

Pusty łańcuch ("" lub puste):

Wartość domyślna. W takim przypadku dane z formularza zostaną przesłane do tej samej strony, na której znajduje się formularz. Często jest to wykorzystywane, gdy strona obsługuje przetwarzanie formularza na tym samym adresie URL, który wyświetla formularz.

```
<form action="" method="post">
```

Atrybut action



Adres URL (URL docelowy):

Możesz określić konkretne URL, do którego mają zostać przesłane dane z formularza. To jest najczęstsze zastosowanie atrybutu action. Na przykład:

```
<form action="przetwarzanie.php" method="post">
```

Adresy URL zewnętrzne:

Możesz także użyć zewnętrznego adresu URL jako wartość atrybutu action, aby dane z formularza były przesyłane na inną stronę lub serwer. Na przykład:

```
<form action="https://www.innastrona.com/odbierz_dane" method="post">
```

Przy określaniu wartości atrybutu action, ważne jest również, aby upewnić się, że adres docelowy istnieje i jest dostępny, oraz że będzie on w stanie poprawnie przetworzyć dane z formularza. W przypadku błędnych lub nieosiągalnych adresów mogą pojawić się błędy podczas próby przesłania danych z formularza.

Pole tekstowe (Text Input):

`<input type="text">`: Pozwala użytkownikowi wprowadzać krótki tekst, np. nazwę, adres e-mail itp.

Pole hasła (Password Input):

`<input type="password">`: Ukrywa wprowadzany tekst, co jest przydatne do wprowadzania haseł.

Pole wyboru (Checkbox):

`<input type="checkbox">`: Pozwala użytkownikowi wybrać jedną lub wiele opcji.

Przycisk opcji (Radio Button):

`<input type="radio">`: Pozwala użytkownikowi wybrać tylko jedną opcję spośród dostępnych.

Pole wyboru (Select):

`<select>` z elementami `<option>`: Umożliwia użytkownikowi wybór z listy rozwijanej.

Pole tekstowe wielolinijkowe (Textarea):

`<textarea>`: Pozwala użytkownikowi wprowadzać dłuższy tekst, np. komentarze.



Pole wyboru pliku (File Input):

`<input type="file">`: Pozwala użytkownikowi wybrać plik z lokalnego komputera do przesłania na serwer.

Pole daty (Date Input):

`<input type="date">`: Umożliwia wybór daty z kalendarza.

Pole czasu (Time Input):

`<input type="time">`: Umożliwia wybór czasu.

Pole liczbowe (Number Input):

`<input type="number">`: Pozwala na wprowadzenie liczb, z opcją określenia zakresu.

Pole e-mail (Email Input):

`<input type="email">`: Ułatwia wprowadzanie adresu e-mail i może sprawdzać jego poprawność.

Pole URL (URL Input):

`<input type="url">`: Ułatwia wprowadzanie adresu URL i może sprawdzać jego poprawność.

Pole wyszukiwania (Search Input):

`<input type="search">`: Stosowane do wprowadzania danych do wyszukiwania.

Pole telefonu (Tel Input):

`<input type="tel">`: Ułatwia wprowadzanie numeru telefonu i może sprawdzać jego poprawność.

Pole koloru (Color Input):

`<input type="color">`: Pozwala na wybór koloru przy użyciu przeglądarkowego selektora kolorów.

```
<h2>Formularz HTML z różnymi typami pól dialogowych</h2>
<form action="submit.php" method="post">
  <label for="username">Imię:</label>
  <input type="text" id="username" name="username" placeholder="Twoje imię" required><br><br>

  <label for="password">Hasło:</label>
  <input type="password" id="password" name="password" placeholder="Hasło" required><br><br>

  <label>Akceptuję regulamin:</label>
  <input type="checkbox" name="agree" value="yes"><br><br>

  <label>Płeć:</label><br>
  <input type="radio" name="gender" value="male" id="male"><label for="male">Mężczyzna</label><br>
  <input type="radio" name="gender" value="female" id="female"><label for="female">Kobieta</label><br><br>

  <label for="country">Kraj zamieszkania:</label>
  <select id="country" name="country">
    <option value="usa">Stany Zjednoczone</option>
    <option value="canada">Kanada</option>
    <option value="uk">Wielka Brytania</option>
  </select><br><br>
  <input type="submit" value="Wyślij">
</form>
```

Wynik:

Formularz HTML z różnymi typami pól dialogowych

Imię:

Hasło:

Akceptuję regulamin: ☐

Płeć:

☐ Mężczyzna

☐ Kobieta

Kraj zamieszkania:


```
<h2>Formularz HTML z różnymi typami pól dialogowych cz. 2</h2>
<form action="submit.php" method="post">
  <label for="comment">Komentarz:</label><br>
  <textarea id="comment" name="comment" rows="4" cols="50"
    placeholder="Napisz swój komentarz..."></textarea><br><br>

  <label for="profile-picture">Załącz zdjęcie profilowe:</label>
  <input type="file" id="profile-picture" name="profile-picture"><br><br>

  <label for="birthdate">Data urodzenia:</label>
  <input type="date" id="birthdate" name="birthdate"><br><br>

  <label for="meeting-time">Godzina spotkania:</label>
  <input type="time" id="meeting-time" name="meeting-time"><br><br>

  <label for="quantity">Ilość produktów:</label>
  <input type="number" id="quantity" name="quantity" min="1" max="10"><br><br>


  <input type="submit" value="Wyślij">
</form>
```


Wynik:

Formularz HTML z różnymi typami pól dialogowych cz. 2

Komentarz:

Załącz zdjęcie profilowe: Nie wybrano pliku

Data urodzenia: 

Godzina spotkania: 

Ilość produktów:

```
<h2>Formularz HTML z różnymi typami pól dialogowych cz. 3</h2>
<form action="submit.php" method="post">

    <label for="email">Adres e-mail:</label>
    <input type="email" id="email" name="email" placeholder="Twój adres e-mail"><br><br>

    <label for="website">Adres strony internetowej:</label>
    <input type="url" id="website" name="website" placeholder="Adres strony internetowej"><br><br>

    <label for="search">Wyszukaj produkt:</label>
    <input type="search" id="search" name="search" placeholder="Szukaj produktu..."><br><br>

    <label for="phone">Numer telefonu:</label>
    <input type="tel" id="phone" name="phone" placeholder="Twój numer telefonu"><br><br>

    <label for="background-color">Wybierz kolor tła:</label>
    <input type="color" id="background-color" name="background-color"><br><br>

    <input type="submit" value="Wyślij">
```

Wynik:

Formularz HTML z różnymi typami pól dialogowych cz. 3

Adres e-mail:

Adres strony internetowej:

Wyszukaj produkt:

Numer telefonu:

Wybierz kolor tła:

Przykład: Poniższy formularz pozwoli wczytać 2 liczby które zostaną przesłane do skryptu *wynik.php*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title></title>
  </head>
  <body>
    <h1>Dodawacz</h1>
    <form action="wynik.php">
      <input name="dana_1" type="number" /><br />
      <label for="dana_2">Druga liczba:</label>
      <input name="dana_2" type="number" /><br />
      <label for="dana_1">Pierwsza liczba:</label>
      <input type="submit" value="Oblicz" />
    </form>
  </body>
</html>
```

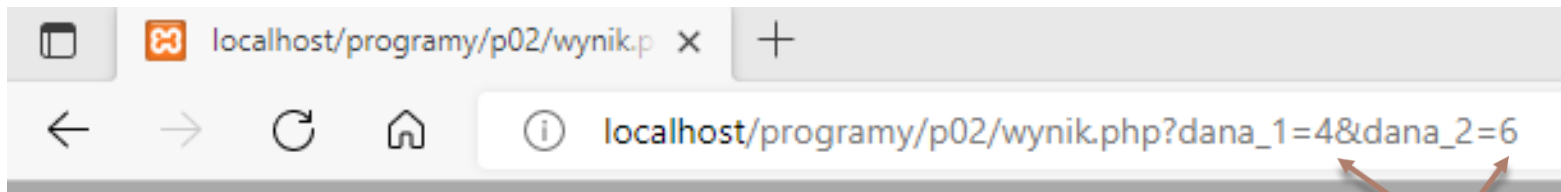
Skrypt otrzymuje dane i wykonuje ich dodawanie

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title></title>
  </head>
  <body>
    <?php
      $x1 = $_GET['dana_1'];
      $x2 = $_GET['dana_2'];
      $wynik = $x1 + $x2;
      echo '<h1>'.$x1.' + '.$x2.' = '.$wynik.'</h1>';
    ?>
  </body>
</html>
```

Metody przesyłania wartości



Domyślnie dane przesyłane są przez formularza metodą GET. Nazwy zmiennych ich wartości doklejane są do adresu wywoływanego skryptu.



Przesłano liczby 4 i 6

Metody przesyłania wartości



Wybranie metody w formularzu polega na dodaniu atrybutu *method* z wartością *POST* lub *GET* do znacznika `<form>`

```
<form action="index.php" method="get">
```

```
<form action="index.php" method="post">
```

Aby odebrać dane przesłane należy odwołać się do nazwy zmiennej przez użycie tablicy:

- dla metody POST `$_POST["zmienna"]`,
- dla metody GET jest to tablica `$_GET["zmienna"]`.

GET

Dane wysyłane metodą GET dołączane są do adresu URL i stanowią jego część. Z taką metodą spotykasz się m.in. w formularzach stosowanych w wyszukiwarkach, dzięki czemu widać, jakie pola zostały przesłane i jaką mają wartość.

Zalety:

- nie potrzeba formularza aby zmodyfikować zawartość tych zmiennych.
- łatwo = dodać taką stronę do "ulubionych" - zakładka będzie kierować wprost do strony z wynikami wyszukiwania.

Wady:

- Informacje umieszczone w adresie są umieszczane w logach serwera, więc nie nadaje się do przesyłania ważnych danych, jak np. haseł, numerów kont, informacji autoryzacyjnych.
- adres URL nie może być dowolnie długi i bezpiecznie jest przyjąć, że nie powinien być dłuższy niż 1024 znaki, choć niektóre przeglądarki obsługują poprawnie adresy o długości ponad 2 tys. znaków.

POST

Post przesyła dane na standardowe wejście skryptu, więc nie posługuje się w tym celu adresem URL. Przesyłanych danych nie widać w adresie i użytkownik też ich nie zobaczy. Dlatego ten sposób przesyłania danych stosowany jest do haseł, przy autoryzacji, czy przesyłaniu innych danych, które nie powinny zostać ujawnione w prosty sposób lub zapisane do logów systemowych serwera.

- ✓ POST nie ma ograniczenia długości danych, więc można przysyłać nie tylko długie pola formularza, ale też całe pliki binarne.
- ✓ Metoda ta musi być więc używana gdy długość danych w formularzu przekroczy 1000 znaków i wszędzie tam, gdzie dane z formularza nie powinny być widoczne w adresie strony.

```
<form action="wynik.php" method="get">
  <label for="dana_1">Pierwsza liczba:</label>
  <input name="dana_1" type="number" /><br>
  <label for="dana_2">Druga liczba:</label>
  <input name="dana_2" type="number" /><br>
  <input type="submit" value="Oblicz">
</form>
```

...wynik.php?dana_1=10&dana_2=20

```
<?php
$x1 = $_GET['dana_1'];
$x2 = $_GET['dana_2'];
$wynik = $x1 + $x2;
echo '<h1>'.$x1.' + '.$x2.' = '.$wynik.'</h1>';
?>
```

```
<form action="wynik.php" method="post">
  <label for="dana_1">Pierwsza liczba:</label>
  <input name="dana_1" type="number" /><br>
  <label for="dana_2">Druga liczba:</label>
  <input name="dana_2" type="number" /><br>
  <input type="submit" value="Oblicz">
</form>
```

...wynik.php

```
<?php
$x1 = $_POST['dana_1'];
$x2 = $_POST['dana_2'];
$wynik = $x1 + $x2;
echo '<h1>'.$x1.' + '.$x2.' = '.$wynik.'</h1>';
?>
```



Sprawdzanie czy przekazano dane z formularza



Aby sprawdzić, czy formularz został już wysłany w skrypcie PHP, możesz wykorzystać zmienną superglobalną **\$_SERVER** w połączeniu z metodą HTTP, która została użyta w formularzu (czyli GET lub POST).

Oto przykład dla metody POST:

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Formularz został wysłany metodą POST
        // Tutaj możesz przetwarzać dane z formularza
    } else {
        // Formularz nie został jeszcze wysłany
    }
?>
```

Walidacja pól obowiązkowych:

Jeśli określone pola formularza są obowiązkowe, możesz sprawdzić, czy zostały one wypełnione. Na przykład:

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $imie = $_POST["imie"];
        $email = $_POST["email"];
        $opcja = $_POST["opcja"];
        if (empty($imie) || empty($email) || empty($opcja)) {
            echo "Wszystkie pola formularza są obowiązkowe.";
        } else {
            // Kontynuuj przetwarzanie danych
        }
    }
?>
```

Walidacja pól obowiązkowych:

Jeśli określone pola formularza są obowiązkowe, możesz sprawdzić, czy zostały one wypełnione. Na przykład:

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $imie = $_POST["imie"];
        $email = $_POST["email"];
        $opcja = $_POST["opcja"];
        if (empty($imie) || empty($email) || empty($opcja)) {
            echo "Wszystkie pola formularza są obowiązkowe.";
        } else {
            // Kontynuuj przetwarzanie danych
        }
    }
?>
```


Walidacja innych typów danych:

W zależności od rodzaju danych, które oczekujesz od użytkownika, można użyć różnych metod walidacji. Na przykład, użyj funkcji **is_numeric** do sprawdzenia, czy wartość jest liczbą:

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $numer = $_POST["numer"];

        if (!is_numeric($numer)) {
            echo "To nie jest poprawna liczba.";
        } else {
            // Kontynuuj przetwarzanie danych
        }
    }
?>
```

Funkcje przydatne do sprawdzania i walidacji danych w PHP, a także do zapewnienia poprawności typów zmiennych przed ich użyciem w kodzie:

1. `is_array($var)`: Sprawdza, czy zmienna jest tablicą.
2. `is_bool($var)`: Sprawdza, czy zmienna jest typu boolean.
3. `is_callable($var, $syntax_only = false, $callable_name = "")`: Sprawdza, czy zmienna jest wywoływalna, czyli funkcją lub metodą.
4. `is_double($var)`: Jest przestarzałą funkcją i zaleca się używanie `is_float()` zamiast niej.
5. `is_float($var)`: Sprawdza, czy zmienna jest typu zmiennoprzecinkowego (float).
6. `is_int($var)`: Sprawdza, czy zmienna jest typu całkowitego (int).
7. `is_integer($var)`: Jest przestarzałą funkcją i zaleca się używanie `is_int()` zamiast niej.
8. `is_long($var)`: Jest przestarzałą funkcją i zaleca się używanie `is_int()` zamiast niej.
9. `is_null($var)`: Sprawdza, czy zmienna jest pusta (null).
10. `is_numeric($var)`: Sprawdza, czy zmienna jest numeryczna.



- 11. *is_object(\$var)*: Sprawdza, czy zmienna jest obiektem.
- 12. *is_real(\$var)*: Jest przestarzałą funkcją i zaleca się używanie *is_float()* zamiast niej.
- 13. *is_resource(\$var)*: Sprawdza, czy zmienna jest zasobem (np. otwartym plikiem).
- 14. *is_scalar(\$var)*: Sprawdza, czy zmienna jest wartością skalarną (czyli bool, int, float lub string).
- 15. *is_string(\$var)*: Sprawdza, czy zmienna jest ciągiem znaków (string).
- 16. *is_uploaded_file(\$filename)*: Sprawdza, czy plik został przesłany na serwer przez formularz.
- 17. *is_writable(\$filename)*: Sprawdza, czy plik jest zapisywalny.

Sprawdzanie, czy zmienna istnieje: isset()



Walidacja formatu email:

Funkcja `isset()` w języku PHP służy do sprawdzania, czy zmienna istnieje i czy ma przypisaną wartość. Funkcja ta zwraca `true`, jeśli zmienna istnieje i ma wartość różną od `null`, oraz `false`, jeśli zmienna nie istnieje lub ma przypisaną wartość `null`.

```
<?php
    $zmienna = "To jest przykładowa zmienna";

    if (isset($zmienna)) {
        echo "Zmienna istnieje i ma przypisaną wartość: " . $zmienna;
    } else {
        echo "Zmienna nie istnieje lub ma wartość null.";
    }
?>
```

Funkcja `isset()` jest często wykorzystywana do sprawdzania, czy użytkownik przesłał dane formularza lub czy określona zmienna jest dostępna, zanim zostanie użyta w kodzie

Usuwanie zmiennych: unset()



Usuwanie zmiennych:

unset() usuwa zmienną, która zostaje przekazana jako argument funkcji. Oznacza to, że zmienna przestaje istnieć i nie można do niej później uzyskać dostępu. Oto przykład:

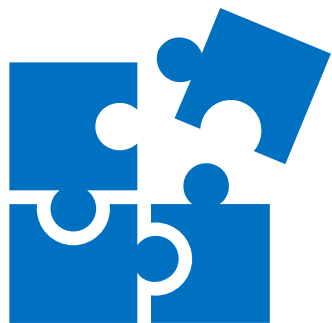
```
<?php
    $zmienna = "Jestem zmienną.";
    unset($zmienna); // Usunięcie zmiennej $zmienna
    echo $zmienna; // Spowoduje błąd, ponieważ zmienna nie istnieje
?>
```

Zwalnianie zasobów: W przypadku obszarów zasobów, takich jak połączenia z bazą danych, otwarte pliki lub innych zasobów, unset() może być używane do zwolnienia tych zasobów. To jest ważne, aby zwolnić zasoby i uniknąć wycieków pamięci oraz konfliktów z innymi częściami skryptu. Oto przykład:

```
<?php
    $plik = fopen("plik.txt", "r");
    // Wykonaj jakieś operacje na pliku
    fclose($plik); // Zamknięcie pliku
    unset($plik); // Usunięcie zmiennej, która trzyma uchwyt do pliku
?>
```



Przykład do wykonania

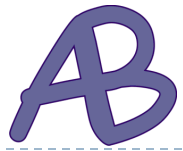


Równanie kwadratowe

Przykład



```
<?php
if (isset($_POST['a'])) {
    $a = $_POST['a'];
    $b = $_POST['b'];
    $c = $_POST['c'];
    if (is_numeric($a) && is_numeric($b) && is_numeric($c)) {
        $delta = $b * $b - 4 * $a * $c;
        echo "delta=" . round($delta,3) . "<br>";
        if ($delta > 0) {
            $x1 = (-$b - sqrt($delta)) / (2 * $a);
            $x2 = (-$b + sqrt($delta)) / (2 * $a);
            echo "x1=" . round($x1, 3) . "<br>";
            echo "x2=" . round($x2, 3) . "<br>";
        } elseif ($delta == 0) {
            $x0 = (-$b) / (2 * $a);
            echo "x0=" . round($x0,3) . "<br>";
        } else {
            echo "Brak rozwiązań<br>";
        }
    } else {
        echo "brak danych";
    }
} else {
    echo "cofnij się do formularza";
}
?>
```



Literatura

W prezentacji użyto przykładów z książki:

- Żygłowicz Jerzy - PHP - Kompendium wiedzy, Helion

- <https://www.php.net>