PRZEDMIOT	KLASA	CZAS TRWANIA	MODUŁ EGZAMINACYJNY
Tworzenie stron i aplikacji internetowych	5 TI	90 minut (2 x 45 min)	Kwalifikacja INF.03

Obsługa formularzy w PHP

Przesyłanie, odbieranie, walidacja i bezpieczne wyświetlanie danych

Cel zajęć

Celem lekcji jest zrozumienie całej drogi danych: od pola formularza HTML, przez żądanie HTTP, do tablicy $_POST w PHP, a następnie ich sprawdzenie, przetworzenie i bezpieczne wyświetlenie w dokumencie HTML.

Po zajęciach uczeń potrafi

- zbudować poprawny formularz HTML i wyjaśnić znaczenie atrybutów action, method, id, name, value i required;
- rozróżnić metody GET i POST oraz wskazać ich typowe zastosowania;
- odebrać dane formularza w PHP za pomocą tablicy $_POST;
- sprawdzić, czy żądanie zostało wysłane metodą POST i czy wymagane dane istnieją;
- zweryfikować dane liczbowe, wykonać rzutowanie typu i sprawdzić zakres wartości;
- bezpiecznie wyświetlić tekst pochodzący od użytkownika za pomocą htmlspecialchars();
- zastosować instrukcję if...else oraz pętlę for do przetwarzania danych formularza.

> **Najważniejsza zasada**
>
> Walidacja w HTML pomaga użytkownikowi, ale nie zastępuje sprawdzania danych w PHP. Każde dane przesłane do serwera mogą zostać zmodyfikowane poza formularzem, dlatego serwer powinien je zweryfikować ponownie.

> **Dlaczego kod jest obrazem?**
>
> Przykłady kodu w tej instrukcji są zapisane jako obrazy, aby zachęcić do samodzielnego przepisywania i analizowania składni zamiast mechanicznego kopiowania.

1. Budowa formularza HTML - znaczniki i atrybuty

Formularz HTML zbiera dane od użytkownika. Sam HTML nie zapisuje danych do bazy ani nie wykonuje obliczeń po stronie serwera - do tego wykorzystamy PHP.

Element	Znaczenie	Najważniejsze informacje
<form>...</form>	Kontener formularza. Obejmuje wszystkie pola, które mają zostać wysłane.	action wskazuje adres skryptu odbierającego dane; method określa metodę HTTP (GET albo POST).
<label>...</label>	Etykieta opisująca pole formularza.	Atrybut for powinien mieć tę samą wartość co id opisywanego pola. Kliknięcie etykiety aktywuje pole.
<input>	Pole wejściowe. Jest znacznikiem pustym - nie ma znacznika zamykającego.	Zachowanie zależy od type, np. text, number, radio, checkbox, submit.
<button>...</button>	Przycisk.	type="submit" wysyła formularz. Między znacznikami można umieścić tekst lub elementy HTML.
<fieldset>...</fieldset>	Grupuje logicznie powiązane pola formularza.	Przydatny np. dla zestawu przycisków radio.
<legend>...</legend>	Tytuł grupy utworzonej przez fieldset.	Poprawia czytelność i dostępność formularza.
...	Akapit tekstu.	Nie jest elementem formularza jako takim, ale może porządkować opisy.
 	Przejście do nowej linii.	W prostych ćwiczeniach może rozdzielać pola, choć w większych projektach układ lepiej wykonywać CSS-em.

Najważniejsze atrybuty pól

Atrybut	Co robi	Przykład / uwaga
type	Określa rodzaj pola input.	text - tekst; number - liczba; radio - jedna opcja z grupy.
id	Jednoznaczny identyfikator elementu na stronie.	Służy m.in. do połączenia pola z label przez for.
name	Nazwa parametru wysyłanego do serwera.	To klucz, pod którym wartość trafia do $_POST lub $_GET. Bez name dane pola nie zostaną wysłane.
value	Wartość wysyłana dla wybranego pola.	Szczególnie ważne dla radio i checkbox.
required	Wymaga uzupełnienia pola przed standardowym wysłaniem formularza.	Jest walidacją po stronie przeglądarki - można ją ominąć, więc PHP nadal musi sprawdzać dane.
min / max	Ustala minimalną i maksymalną wartość pola number.	Pomaga użytkownikowi, ale zakres trzeba także sprawdzić w PHP.
step	Określa krok wartości liczbowej.	step="0.01" pozwala wygodnie wprowadzać kwoty z groszami.
checked	Domyślnie zaznacza radio lub checkbox.	W jednej grupie radio zwykle tylko jedna opcja ma checked.
for	W label wskazuje id pola, którego	label for="wiek" łączy się z input

	dotyczy etykieta.	id="wiek".

Zapamiętaj: id i name mają różne zadania

id identyfikuje element w dokumencie HTML i może łączyć go z etykietą. name określa nazwę parametru wysyłanego do PHP. Dwa atrybuty często mają podobną wartość, ale nie są tym samym.

2. GET i POST - jak dane trafiają do PHP?

Po kliknięciu przycisku submit przeglądarka tworzy żądanie HTTP. Wartości pól posiadających atrybut name są kodowane i wysyłane do adresu określonego w action.

Cecha	GET ($_GET)	POST ($_POST)
Gdzie są dane?	W parametrach adresu URL, np. ?q=php&page=2.	W treści żądania HTTP. Nie są dopisywane do adresu URL.
Widoczność dla użytkownika	Parametry są widoczne w pasku adresu i mogą trafić do historii przeglądarki.	Nie są widoczne w pasku adresu, ale nadal można je podejrzeć np. w narzędziach deweloperskich.
Typowe zastosowania	Wyszukiwanie, filtry, paginacja, odnośniki do konkretnego widoku.	Logowanie, formularze zapisujące dane, większe formularze, operacje zmieniające stan aplikacji.
Rozmiar danych	Praktycznie ograniczony długością adresu URL i ograniczeniami klienta/serwera.	Limit zależy od konfiguracji serwera i PHP, np. ustawienia post_max_size.
Bezpieczeństwo	GET nie powinien przenosić sekretów w URL.	POST nie szyfruje danych. Poufność transmisji zapewnia HTTPS, nie sama metoda POST.
Powtarzalność/adresowalność	Łatwo zapisać, skopiować lub udostępnić adres z parametrami.	Dane nie są częścią adresu, dlatego nie da się odtworzyć żądania samym URL-em.

Bardzo ważne - POST nie oznacza „bezpiecznie”

POST ukrywa dane przed paskiem adresu, ale nie szyfruje ich. Dane przesyłane przez sieć chroni protokół HTTPS. Zarówno GET, jak i POST można ręcznie zmodyfikować, dlatego PHP musi sprawdzać wszystkie wartości po stronie serwera.

Schemat przepływu danych

- 1. Użytkownik wpisuje dane do pól formularza.
- 2. Przeglądarka zbiera tylko pola posiadające name.
- 3. Formularz wysyła żądanie do adresu z action metodą GET albo POST.
- 4. PHP udostępnia odebrane wartości w odpowiedniej tablicy superglobalnej.
- 5. Skrypt waliduje dane, wykonuje obliczenia i generuje odpowiedź HTML.

Dlaczego name jest kluczowe?

Jeżeli pole ma name="user_imie", to przy metodzie POST jego wartość można odczytać w PHP jako $_POST['user_imie']. Pole bez atrybutu name nie zostanie przekazane w standardowym wysłaniu formularza.

3. Przykład kompletnego formularza HTML

Poniższy dokument zawiera pełną strukturę HTML oraz formularz wysyłany metodą POST do pliku oblicz.php. Kod należy przepisać samodzielnie.

FRAGMENT DO SAMODZIELNEGO PRZEPISANIA - kod zapisany jako obraz

```html
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Formularz użytkownika</title>
</head>
<body>
    <h1>Dane użytkownika</h1>

    <form action="oblicz.php" method="POST">
        <label for="imie">Podaj imię:</label>
        <input type="text" id="imie" name="user_imie" required>
        <br><br>

        <label for="wiek">Twój wiek:</label>
        <input type="number" id="wiek" name="user_wiek"
               min="1" max="120" required>
        <br><br>

        <fieldset>
            <legend>Wybierz płeć:</legend>
            <label>
                <input type="radio" name="plec" value="K"> Kobieta
            </label>
            <label>
                <input type="radio" name="plec" value="M" checked> Mężczyzna
            </label>
        </fieldset>

        <button type="submit">Wyślij dane</button>
    </form>
</body>
</html>
```

Co warto zauważyć w tym przykładzie?

- DOCTYPE informuje przeglądarkę, że dokument korzysta ze współczesnego standardu HTML5.
- meta charset="UTF-8" ustawia kodowanie znaków i pozwala poprawnie wyświetlać polskie litery.
- form ma action="oblicz.php", więc po wysłaniu dane otrzyma właśnie ten plik.
- method="POST" powoduje, że wartości trafią do tablicy $_POST w PHP.
- label for="imie" jest połączony z input id="imie".
- name="user_imie" i name="user_wiek" są nazwami parametrów używanymi później w PHP.
- dwa pola radio mają ten sam name="plec" - dzięki temu tworzą jedną grupę i można wybrać tylko jedną wartość.
- wartości value="K" oraz value="M" są tym, co zostanie wysłane do serwera.
- required, min i max poprawiają obsługę formularza, ale nie zastępują walidacji w PHP.

4. Odbieranie danych w PHP

PHP udostępnia kilka tzw. tablic superglobalnych. Są one dostępne w całym skrypcie bez dodatkowego deklarowania.

Element PHP	Znaczenie
$_POST	Tablica asocjacyjna z danymi przesłanymi standardowym formularzem metodą POST. Kluczami są wartości atrybutów name.
$_GET	Tablica asocjacyjna z parametrami przesłanymi w adresie URL metodą GET.
$_SERVER	Tablica z informacjami o żądaniu i środowisku serwera. W tym ćwiczeniu używamy klucza REQUEST_METHOD.
isset(...)	Sprawdza, czy zmienna lub element tablicy istnieje i nie ma wartości null. Nie sprawdza poprawności treści.
trim(...)	Usuwa białe znaki z początku i końca tekstu, np. przypadkowe spacje.
(int)	Rzutuje wartość na liczbę całkowitą. Pomaga uzyskać przewidywalny typ danych, ale nadal trzeba sprawdzić zakres.
htmlspecialchars(...)	Koduje znaki specjalne HTML przed umieszczeniem tekstu użytkownika w treści strony. Chroni kontekst HTML przed wstrzyknięciem znacznika lub skryptu.
exit	Natychmiast kończy wykonywanie bieżącego skryptu. Przydatne, gdy wykryto błędne dane i dalsze obliczenia nie mają sensu.

Cztery etapy poprawnej obsługi danych

1. Odbierz - odczytaj wartość z $_POST, najlepiej dopiero po sprawdzeniu, czy żądanie jest typu POST.
2. Waliduj - sprawdź, czy wartość istnieje, ma oczekiwany typ i mieści się w dozwolonym zakresie.
3. Przetwórz - wykonaj obliczenia lub logikę programu dopiero na poprawnych danych.
4. Wyświetl bezpiecznie - dane tekstowe użytkownika umieszczane w HTML koduj przez htmlspecialchars().

> **Uwaga na pojęcia**
>
> isset() nie jest walidacją wartości - sprawdza tylko istnienie. Rzutowanie na int nie sprawdza automatycznie, czy liczba ma sens biznesowy. htmlspecialchars() nie „czyści bazy danych” i nie zastępuje zapytań przygotowanych SQL; jego rolą jest bezpieczne umieszczenie tekstu w HTML.

Wariant bardziej formalny - filter_input()

PHP posiada również funkcję filter_input(), która pozwala pobrać daną wejściową i zastosować filtr walidujący. To dobry kierunek w bardziej rozbudowanych aplikacjach.

PRZYKŁAD ROZSZERZONY - nie jest wymagany do podstawowego ćwiczenia

```
$wiek = filter_input(INPUT_POST, 'user_wiek', FILTER_VALIDATE_INT);
if ($wiek === false || $wiek === null) {
    echo "Wiek musi być liczbą całkowitą.";
}
```

5. Skrypt PHP - walidacja, warunek i pętla

Skrypt oblicz.php sprawdza metodę żądania, odbiera wartości, waliduje wiek i imię, bezpiecznie przygotowuje imię do wyświetlenia, a następnie używa instrukcji warunkowej i pętli.

FRAGMENT DO SAMODZIELNEGO PRZEPISANIA - kod zapisany jako obraz

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $imie = isset($_POST['user_imie']) ? trim($_POST['user_imie']) : '';
    $wiek = isset($_POST['user_wiek']) ? (int) $_POST['user_wiek'] : 0;
    $plec = isset($_POST['plec']) ? $_POST['plec'] : '';

    // Walidacja danych po stronie serwera
    if ($imie === '' || $wiek < 1 || $wiek > 120) {
        echo "Niepoprawne dane formularza.";
        exit;
    }

    // Bezpieczne wyświetlanie tekstu w HTML
    $imie_html = htmlspecialchars($imie, ENT_QUOTES, 'UTF-8');

    if ($wiek >= 18) {
        echo "<h3>Witaj $imie_html! Jesteś osobą pełnoletnią.</h3>";
    } else {
        $brakuje = 18 - $wiek;
        echo "<h3>Witaj $imie_html! Brakuje Ci $brakuje lat do pełnoletności.</h3>";
    }

    echo "<p>Twoje szczęśliwe liczby:</p><ul>";
    for ($i = 1; $i <= 3; $i++) {
        $losowa = rand(1, 100);
        echo "<li>Liczba $i: $losowa</li>";
    }
    echo "</ul>";
} else {
    echo "Otwórz najpierw formularz i wyślij dane metodą POST.";
}
?>
```

Dlaczego używamy ===?

Operator === sprawdza jednocześnie wartość i typ danych. W porównaniach kontrolnych jest bardziej jednoznaczny niż ==, które może wykonywać konwersje typów przed porównaniem.

6. Co dokładnie robi kod PHP?

Sprawdzenie metody żądania

Warunek $_SERVER['REQUEST_METHOD'] === 'POST' pozwala wykonać logikę formularza tylko wtedy, gdy bieżące żądanie rzeczywiście używa metody POST. Wejście bezpośrednio na oblicz.php z paska adresu jest standardowo żądaniem GET.

Pobieranie wartości

Operator trójargumentowy ma postać: warunek ? wartość_gdy_prawda : wartość_gdy_fałsz. W instrukcji używamy go razem z isset(), aby ustawić wartość domyślną, jeśli pole nie zostało przesłane.

> **Przykład znaczeniowy**
>
> „Jeżeli istnieje $_POST['user_imie'], pobierz tę wartość i usuń skrajne spacje; w przeciwnym razie ustaw pusty napis.”

Walidacja zakresu

Wiek po rzutowaniu na int jest sprawdzany ponownie w PHP. Dzięki temu skrypt nie ufa jedynie ograniczeniom min i max z formularza HTML, które da się ominąć.

Bezpieczne wyświetlanie tekstu

htmlspecialchars($imie, ENT_QUOTES, 'UTF-8') zamienia znaki mające szczególne znaczenie w HTML na bezpieczne odpowiedniki. Tekst użytkownika pozostaje tekstem zamiast stawać się fragmentem kodu strony.

Instrukcja if...else

Na podstawie wieku program wykonuje jedną z dwóch gałęzi. Jeżeli wiek jest co najmniej 18, wyświetla informację o pełnoletności. W przeciwnym razie oblicza, ile lat brakuje do 18.

Pętla for

Pętla wykonuje się trzy razy. Zmienna $i pełni rolę licznika, a rand(1, 100) zwraca pseudolosową liczbę całkowitą z zakresu 1-100. Wyniki są dopisywane do listy HTML.

Najczęstsze błędy uczniów

- brak name w input - wartość nie trafia do $_POST;
- inna nazwa w HTML i inny klucz w PHP, np. name="wiek" oraz $_POST['user_wiek'];
- porównywanie lub obliczanie danych bez wcześniejszego sprawdzenia, czy istnieją;
- dzielenie przez zero albo brak sprawdzenia zakresu liczb;
- wyświetlanie surowego tekstu użytkownika w HTML bez htmlspecialchars();
- mylenie id z name;
- brak średnika na końcu instrukcji PHP;
- niedomknięte cudzysłowy, nawiasy lub klamry.

7. Zadanie laboratoryjne do oceny

Uproszczony symulator spłaty - „Kalkulator Finansowy”

> **Założenie edukacyjne**
>
> To zadanie ćwiczy formularze, walidację, instrukcje warunkowe, obliczenia i pętle. Model oprocentowania jest celowo uproszczony i nie odzwierciedla rzeczywistych harmonogramów bankowych, RRSO ani rat malejących/równych.

Utwórz jeden plik index.php zawierający formularz HTML oraz kod PHP przetwarzający dane po wysłaniu formularza.

A. Część HTML - interfejs użytkownika

- formularz ma używać method="POST";
- pole tekstowe Imię i nazwisko: type="text", własne id, name="klient", label i required;
- pole Kwota kredytu: type="number", name="kwota", wartość minimalna większa od zera oraz step="0.01";
- pole Liczba rat: type="number", name="raty", min="1" i required;
- grupa radio „Rodzaj klienta”: standardowy (8%) oraz VIP (5%); oba pola mają ten sam name, np. name="typ";
- każde radio ma jednoznaczną wartość value, np. standard lub vip;
- formularz zawiera przycisk button type="submit".

B. Część PHP - dane, walidacja i obliczenia

- kod przetwarzający uruchamiaj tylko dla żądania POST;
- odbierz wartości z $_POST i zabezpiecz brakujące dane przez isset() lub równoważny mechanizm;
- rzuć kwotę na float, a liczbę rat na int;
- sprawdź, czy klient nie jest pusty, kwota jest większa od 0, liczba rat jest co najmniej 1, a typ klienta ma dozwoloną wartość;
- ustal oprocentowanie instrukcją if...else: standard = 0.08, VIP = 0.05;
- oblicz całkowitą kwotę do spłaty i wysokość jednej raty;
- wyświetl imię i nazwisko po przejściu przez htmlspecialchars();
- sformatuj kwoty do dwóch miejsc po przecinku, np. funkcją number_format();
- za pomocą pętli for wygeneruj listę rat w postaci: Rata 1: ... PLN, Rata 2: ... PLN itd.

ZESPÓŁ SZKÓŁ PONADPODSTAWOWYCH W NIEMCACH

ZASADY SAMODZIELNOŚCI - ZADANIE OCENIANE

Uczeń wykonuje rozwiązanie samodzielnie. Narzędzia AI mogą służyć wyłącznie do wyjaśniania pojęć, zadawania pytań naprowadzających oraz analizy błędów we własnym kodzie. Niedozwolone jest generowanie przez AI kompletnego rozwiązania lub gotowego kodu realizującego wymagania zadania.

8. Model obliczeń i algorytm zadania

Na potrzeby ćwiczenia stosujemy bardzo prosty model: odsetki są naliczane jednorazowo jako procent kwoty początkowej, a wszystkie raty mają tę samą wartość.

WZORY DO WYKORZYSTANIA W SKRYPCIE

```
$calkowita_kwota = $kwota + ($kwota * $oprocentowanie);
$rata = $calkowita_kwota / $ilosc_rat;
```

Przed wykonaniem dzielenia upewnij się, że $ilosc_rat jest większa od zera. W formularzu ustaw min="1", ale sprawdzenie musi istnieć również w PHP.

Proponowana kolejność pracy

5. Zbuduj sam formularz HTML i sprawdź, czy wszystkie pola mają name.
6. Dodaj warunek sprawdzający metodę POST.
7. Odbierz wartości i przypisz je do zmiennych.
8. Wykonaj walidację danych oraz zakończ skrypt, jeśli są błędne.
9. Instrukcją if...else ustal oprocentowanie na podstawie typu klienta.
10. Oblicz całkowitą kwotę i pojedynczą ratę.
11. Bezpiecznie wyświetl nazwę klienta oraz podsumowanie.
12. Pętlą for wygeneruj wszystkie raty.
13. Przetestuj program dla kilku różnych zestawów danych.

Minimalny zestaw testów

Test	Przykładowe dane	Oczekiwane zachowanie
1	Klient standard, kwota 1000, 10 rat	Program liczy 8% i wyświetla 10 rat.
2	Klient VIP, kwota 1000, 10 rat	Program liczy 5% i wyświetla niższą ratę niż dla klienta standard.
3	Liczba rat = 0	Program nie wykonuje dzielenia; wyświetla komunikat o błędnych danych.
4	Kwota ujemna	Program odrzuca wartość.
5	Puste imię i nazwisko	Program nie przechodzi do obliczeń.
6	Nietypowy tekst w nazwie klienta	Tekst zostaje wyświetlony jako tekst, a nie jako kod HTML.

Wyzwanie dodatkowe

Dodaj pole wyboru okresu promocyjnego albo prowizji i zmień obliczenia tak, aby nowy parametr był jawnie pokazany w podsumowaniu. Pamiętaj: każdą nową daną z formularza trzeba walidować również w PHP.

9. Kryteria oceniania - maksymalnie 20 punktów

Kryterium	0 pkt	1-2 pkt	3-4 pkt
Struktura HTML i formularz	brak działania	część pól lub atrybutów jest błędna	poprawny formularz, label, typy pól, name, wymagane ograniczenia
POST i odbiór danych w PHP	brak odbioru	częściowe odczytanie lub błędy kluczy	poprawne sprawdzenie POST oraz bezpieczne pobranie wymaganych wartości
Walidacja i instrukcje warunkowe	brak	częściowa walidacja	sprawdzenie danych po stronie PHP, zakresów oraz poprawne przypisanie oprocentowania
Obliczenia	brak lub błędne	częściowo poprawne	poprawna kwota całkowita, rata, zabezpieczenie przed błędnymi wartościami
Pętla i prezentacja wyniku	brak	pętla lub wynik częściowo działa	poprawny harmonogram rat, czytelny wynik, tekst użytkownika bezpiecznie wyświetlany

Skala przyjęta na potrzeby tych zajęć

Punkty	Procent	Ocena
20	100%	celujący (6)
18-19	90-95%	bardzo dobry (5)
17	85%	dobry (4)
16	80%	dostateczny (3)
15	75%	dopuszczający (2) - próg zaliczenia zadania
0-14	0-70%	niedostateczny (1)

Ważne rozróżnienie

W egzaminie zawodowym część praktyczną uznaje się za zdaną po uzyskaniu co najmniej 75% punktów. Powyższa szkolna skala ocen 1-6 jest przyjęta na potrzeby tego ćwiczenia przez nauczyciela; nie jest oficjalną skalą ocen CKE.

Dodatkowe zasady punktowania

- literówka w name, która uniemożliwia odebranie konkretnej wartości w PHP, może obniżyć punktację za daną część;
- samo required w HTML nie jest pełną walidacją - liczy się również sprawdzenie w PHP;
- CSS nie jest obowiązkowy, chyba że nauczyciel rozszerzy wymagania; estetyka nie może zastąpić poprawności działania;
- kod powinien mieć logiczne wcięcia i czytelne nazwy zmiennych;
- program powinien reagować na błędne dane bez ostrzeżeń PHP i bez dzielenia przez zero.

10. Ściąga - znaczniki, atrybuty i funkcje użyte na lekcji

Nazwa	Kategoria	Do czego służy
<form>	HTML	tworzy formularz i określa action oraz method
<label>	HTML	opisuje pole; for łączy etykietę z id pola
<input>	HTML	tworzy pole wejściowe; typ zależy od type
<button>	HTML	tworzy przycisk; type="submit" wysyła formularz
<fieldset> / <legend>	HTML	grupuje i podpisuje powiązane pola
name	atrybut HTML	ustala nazwę parametru wysyłanego do PHP
id	atrybut HTML	identyfikuje element w dokumencie
value	atrybut HTML	ustala wartość wysyłaną np. przez radio
required	atrybut HTML	wymaga wartości w standardowej walidacji przeglądarki
min / max / step	atrybut HTML	ogranicza i ułatwia wpisywanie liczb po stronie klienta
$_POST	PHP	dostęp do parametrów standardowego formularza wysłanego POST
$_GET	PHP	dostęp do parametrów zapytania z URL
$_SERVER	PHP	informacje o żądaniu i serwerze
isset()	PHP	sprawdza, czy wartość istnieje i nie jest null
trim()	PHP	usuwa białe znaki z początku i końca tekstu
htmlspecialchars()	PHP	koduje znaki specjalne przed wyświetleniem tekstu w HTML
filter_input()	PHP	pobiera i może walidować dane wejściowe
number_format()	PHP	formatuje liczbę do wyświetlenia, np. z 2 miejscami po przecinku
if...else	PHP	wykonuje różne instrukcje zależnie od warunku
for	PHP	powtarza blok kodu określoną liczbę razy
rand()	PHP	zwraca pseudolosową liczbę całkowitą z podanego zakresu
exit	PHP	kończy wykonywanie skryptu

Lista kontrolna przed oddaniem

- Czy każde pole, które ma trafić do PHP, ma atrybut name?
- Czy klucze w $_POST są identyczne z wartościami name w HTML?

- Czy kod PHP uruchamia obliczenia tylko po wysłaniu formularza?
- Czy dane liczbowe są sprawdzane także po stronie PHP?
- Czy liczba rat nie może spowodować dzielenia przez zero?
- Czy tekst użytkownika jest wyświetlany przez htmlspecialchars()?
- Czy pętla generuje dokładnie tyle rat, ile podał użytkownik?
- Czy program został przetestowany na danych poprawnych i błędnych?

Źródła merytoryczne

Opracowanie oparto na dokumentacji PHP Manual dotyczącej formularzy, tablic $_POST, funkcji htmlspecialchars() i filter_input() oraz na aktualnych informacjach CKE dotyczących progu zaliczenia części praktycznej egzaminu zawodowego. Materiał został uproszczony dydaktycznie do poziomu ćwiczenia INF.03.