T: Przechowywanie danych w PHP. Pliki jednorodne.

1. Istnieją dwa sposoby przechowywania danych - w pliku jednorodnym oraz w bazie danych. Plik jednorodny może mieć wiele różnych formatów, lecz zazwyczaj terminem tym oznacza się prosty plik tekstowy. Temat dotyczy przechowywania danych w plikach tekstowych.

2. Zapisywanie danych w pliku następuje w trzech etapach. Są to:
   a) otwarcie pliku. Jeżeli dany plik nie istnieje, należy go utworzyć,
   b) zapisanie danych w pliku,
   c) zamknięcie pliku.

3. Aby otworzyć plik w PHP, stosuje się funkcję **fopen()**. Można otworzyć plik w następujących trybach: tylko do odczytu, tylko do zapisu lub do obu tych celów. Przy zapisywaniu danych w pliku można nadpisać istniejące dane bądź dodać nowe na jego końcu.

4. Rodzaje dostępu dla fopen():
   'r' Otwiera tylko do odczytu; umieszcza wskaźnik pliku na jego początku;
   'r+' Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku;
   'w' Otwiera tylko do zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje to próbuje go utworzyć;
   'w+' Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje to próbuje go utworzyć;
   'a' Otwiera tylko do zapisu; umieszcza wskaźnik pliku na jego końcu.
   Jeśli plik nie istnieje to próbuje go utworzyć;
   'a+' Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje to próbuje go utworzyć;
   'x' Tworzy i otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie fopen() nie powiedzie się, zwróci FALSE i wygeneruje błąd na poziomie E_WARNING. Jeśli plik nie istnieje, spróbuje go utworzyć.
   'x+' Tworzy i otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie fopen() nie powiedzie się, zwróci FALSE i wygeneruje błąd na poziomie E_WARNING. Jeśli plik nie istnieje, spróbuje go utworzyć.

5. Dla przenośności zalecane jest użycie flagi **'b'** do otwierania plików za pomocą fopen().

6. Stosowanie funkcji **fopen()**:
   wykorzystując wbudowaną w PHP zmienną **$DOCUMENT_ROOT,** która wskazuje na podstawowy element drzewa katalogów serwera WWW; podobnie jak w przypadku nadawania zmiennym formy krótkich nazw, na początku skryptu należy umieścić następujący wiersz:
   $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
   $plik = fopen(„$DOCUMENT_ROOT/ .. /dane/ceny.txt”,'w');

7. Można określić bezwzględną ścieżkę dostępu do pliku, będąca ścieżką do katalogu głównego. W środowisku Uniksa stosuje się ukośniki (/), natomiast w środowisku Windows można używać lewych (\) lub prawych ukośników (/), które muszą zostać oznaczone jako znaki specjalne, aby funkcja fopen właściwie je zinterpretowała. W tym celu należy po prostu dodać przed każdym symbolem jeszcze jeden lewy ukośnik, jak pokazano w poniższym przykładzie:
   $plik = fopen(" .. \\ .. \\zamowienia\\zamowienia.txt", 'w');

8. Przykłady użycia fopen():

```
<?php
$plik = fopen("/home/www/plik.txt", "r");
$plik = fopen("/home/www/plik.gif", "wb");
$plik = fopen("http://www.przyklad.com/", "r");
$plik = fopen("ftp://uzytkownik:haslo@przyklad.com/plik.txt", "w");
?>
```

9. Prosty przykład radzenia sobie z błędami otwarcia pliku:

```
@$plik = fopen( "$DOCUMENT_ROOT/../dane/ceny.txt",'ab');
if (!$plik) {
echo „<p><strong> Nie można dopisać cen. </strong></p></body></html>”;
exit;
}
```

@ - to operator tłumienia błędów.

10. Zapisywanie danych do pliku: **fwrite()** (zapis do pliku) lub **fputs()** (umieszczenie ciągu w pliku); działają tak samo.

fwrite($plik, $ciag_znakow);
lub:
fwrite($plik, $ciag_znakow, strlen($ciag_znakow));

11. Zamknięcie pliku: **fclose($plik);**

12. Przykład prostego kodu zapisującego dane do pliku:

```
<?php
        $a = "Witryna";
        $b = "dynamiczna";
        $txt = $a." ".$b."\n";
        $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
        $plik=fopen("dane123.txt", 'ab');
        //lub:
        //$plik=fopen("$DOCUMENT_ROOT/../dane123.txt", 'ab');
        flock($plik, LOCK_EX); //blokowanie zapisu - plik nie może być dzielony
        if (!$plik) {
                echo "<p><strong>Błąd!</strong></p></body></html>";
        exit; }
        fwrite($plik, $txt, strlen($txt));
        flock($plik, LOCK_UN); //zwolnienie istniejącej blokady
        fclose($plik);
?>
```

13. Napisać skrypt PHP, który zapisze do pliku dane pobrane z formularza. W jednym wierszu pliku powinno być: imię, nazwisko, wiek, pesel, aktualna data i godzina. Kliknięcie przycisku 'wyślij' dopisze kolejne wiersze do pliku.

14. Odczyt danych z pliku; przykład:

```
<?php
        $plik=fopen("dane123.txt", 'rb');
        if (!$plik){
                echo "<p><strong>Brak wpisów.</strong></p>" ;
                exit;
                }
        while(!feof($plik)) {
                $wpis = fgets($plik, 999);
                echo $wpis."<br />";
        }
    fclose($plik) ;
?>
```

15. Funkcja **feof()** używa wskaźnika pliku jako swojego jedynego parametru. Zwraca ona wartość true, jeżeli wskaźnik pliku znajduje się na jego końcu.

16. Funkcja **gets()** jest stosowana do odczytywania pliku wiersz po wierszu. W powyższym przypadku będzie odczytywała dane, dopóki nie trafi na znak nowego wiersza (\n), na EOF lub przeczyta 998 bajtów pliku.

17. Inną odmianą funkcji fgets() jest funkcja fgetcsv(), przy jej stosowaniu plik jest odczytywany nie wiersz po wierszu, lecz od znaku podziału do znaku podziału, np. $zamowienie = fgetcsv($plik, 100, "\t"); Polecenie odczyta wiersz z pliku i podzieli ją tam, gdzie natrafi na znak tabulacji (\t). Wyniki zwracane są w postaci tablicy ($zamowienie).
    Drugi parametr (w naszym przykładzie 100) powinien mieć większą wartość niż długość (wyrażoną w liczbie znaków) najdłuższego wiersza odczytywanego pliku.

18. Aby odczytać cały plik od razu, możemy użyć **readfile()**:
    readfile("$DOCUMENT_ROOT/ .. /dane123.txt");
    Dane z pliku zostaną wyświetlone w oknie przeglądarki.

19. Funkcja fgetc() służy do odczytywania pliku znak po znaku.
    Przykład:

```
while (!feof($plik)) {
    $znak = fgetc($plik);
    if (!feof($plik)) {
        echo ($znak==" \n" ? "<br />" : $znak);
        }
        }
```

20. Istnieje wiele przydatnych funkcji plikowych: file_exists() - sprawdza czy plik istnieje, filesize() - sprawdza wielkość pliku w bajtach, unlink() - usuwa plik, ... , i wiele innych.