PHP I - zajęcia 8 – łańcuchy znaków i wyrażenia regularne

1. **Odnajdywanie pozycji łańcucha znaków**
   - ★ strpos($tablca, “szukany ciąg znaków”, [offset]) - zwraca numeryczną pozycję szukanego ciągu znaków. Jako wynik podaje pierwsze wystąpienie. Jako Offset możemy podać indeks od którego funkcja ma rozpocząć wyszukiwanie.
   - ★ strrpos() - działa jak strpos() przy czym zwraca ostatnie wystąpienie.
2. **Zmiana fragmentów łańcucha znaków**
   - ★ str_replace($tablica_szukanych_slow, “zamiennik”, ”przeszukiwany ciąg znaków”) - funkcja szuka ciągów znaków podanych w pierwszym parametrze (może to być jedno słowo) a następnie zastępuje je ciągiem znaków podanym w 2gim parametrze, trzeci parametr to ciąg znaków który jest przeszukiwany.
   - ★ substr_replace(“ciag znakow”, “zamiennik”,start, [długość]) - funkcja zamienia część łańcucha znaków podanego w pierwszym parametrze na ciąg znaków podany w parametrze 2gim.Parametr start odpowiada za offset w łańcuchu, jeżeli jest wartością większą równą zero jej pozycja liczy się od początku łańcucha, jeżeli jest ujemna liczy się od końca łańcucha. Opcjonalny parametr długość określa po jakiej ilości znaków funkcja powinna zakończyć przeszukiwanie. Jeżeli wynosi 0 zamiennik zostanie wstawiony do łańcucha bez nadpisania istniejącej wartości. Tak jak w str_replace() jako pierwszy parametr można podać tablicę.
3. **Wyrażenia regularne wstęp i ograniczniki**
   - ★ Wyrażenia regularne zapewniają nam dopasowanie wzorców w łańcuchach znaków. Np. Wyrażenie /[a-z]/ będzie pasowało do każdego łańcucha znaków zawierającego małą literę alfabetu, nie ma znaczenia czy łańcuch zawierać będzie jeden znak czy poszukiwany znak jest jednym z wielu w dłuższym łańcuchu.
   - ★ / - jest to najczęściej używany ogranicznik, poszukiwania słowa “sklep” należałoby opisać w następujący sposób: /sklep/
   - ★ \/ - jeżeli jest konieczne użycie znaku “/” ukośnika należy go poprzedzić backslashem “\” np. /http:\/\//
   - ★ # - Jeżeli w danym wyrażeniu dany ogranicznik występuje wiele razy można posłużyć się innym np.. #http://#
   - ★ i- modyfikator wzorca. Zapewnia brak rozróżnienia w wielkości liter. Jest najczęściej używanym. Np. /sklep/i
4. **Zbiory i klasy znaków**
   - ★ . - może zastąpić dowolny znak np. /.łot/ zwróci zarówno “młot”, “płot”, “błot” jak i “&łot”
   - ★ [a-z] - zastępuje znaki będące literami alfabetu np. /[a-z]łot/ => “młot” , “płot”,”błot”. By nie rozróżniać wielkości liter możemy zastosować modyfikator wzorca lub zapisać [a-zA-Z]
   - ★ [abcd...e] - wyliczenie znaków - klasa znaków np. /[mp]łot/ => “młot”, ”płot”
   - ★ [^a-z] - wykluczenie – akcent przeciągły, znacznik ^ określa jakie znaki nie mogą być użyte
   - ★ Istnieje klika predefiniowanych klas znaków. Zewnętrzna para nawiasów klamrowych ogranicza klasę a wewnętrzna para wchodzi w skład jej nazwy np.. /[[:alpha]1-5]/:
     - ○ [[:alnum]] - znaki alfanumeryczne
     - ○ [[:alpha]] - znaki alfabetu
     - ○ [[:ascii]] - znaki ASCII

- [[:lower]] - małe litery
- [[:upper]] - wielkie litery
- [[:word]] - znaki tworzące słowa w tym litery, cyfry i znaki podkreślenia
- [[:digit]] - liczby dziesiętne
- [[:xdigit]] - liczby szesnastkowe
- [[:punct]] - znaki przystankowe
- [[:blank]] - tabulatory i spacje
- [[:space]] - znaki odstępu
- [[:cntrl]] - znaki kontrolne
- [[:print]] - wszystkie możliwe do wyświetlenia znaki
- [[:graph]] - wszystkie możliwe do wyświetlenia znaki po za spacjami

5. **Powtarzalność**
   - ★ * - oznacza że wzorzec może powtórzyć się 0 bądź więcej razy
   - ★ + - oznacza że wzorzec może powtórzyć się 1 bądź więcej razy
   - ★ ?- oznacza że wzorzec może powtórzyć się 0 bądź 1 raz
   - ★ Symbole powinny wystąpić bezpośrednio po części wyrażenia której dotyczą np.:
     - /[[:alnum]+]/
6. **Podwyrażenia i podwyrażenia policzalne**
   - ★ Wyrażenia możemy rozdzielić stosując nawiasy tworząc podwyrażenia w celu np. Znalezienia “co najmniej jednego z tych łańcuchów, a następnie dokładnie tego” np..
     - /(bardzo)*dużo/ => “dużo”, “bardzo dużo”, “bardzo bardzo dużo” itd.
   - ★ Możemy określić ilość wystąpień danego znaku umieszczając liczbę bądź zakres w nawiasach klamrowych
     - {2} - dokładnie 2 wystąpienia
     - {2,4} - od 2 do 4 wystąpień
     - {2, } - dwa wystąpienia i więcej (zakres otwarty)
     - Np. /(bardzo){1,3}
7. **Kotwiczenie na początku lub końcu łańcucha znaków i rozgałęzienia**
   - ★ ^ - symbol jest stosowany na początku wyrażenia regularnego w celu wskazania że musi się ono pojawić na początku szukanego łańcucha znaków. Np. /^janek/
   - ★ $ - symbol jest stosowany na końcu wyrażenia regularnego w celu wskazania że musi się ono pojawić na końcu szukanego łańcucha znaków. Np. /com$/
   - ★ | - symbol jest stosowany pomiędzy wyrażeniami by wybrać pasujące np. /com|edu|net/
8. **Odnajdywanie fragmentów łańcuchów za pomocą wyrażeń regularnych**
   - ★ preg_mach(wzorzec, “szukaj”, $tablicaDopasowania[,int flagi=0[,int przesunięcie =0]]]); - funkcja ta przeszukuje łańcuch znaków podawany w drugim parametrze (“szukaj”) w celu odnalezienia fragmentów pasujących do wyrażenia regularnego podanego w łańcuchu w pierwszym parametrze (wzorzec). Jeżeli uda się odnaleźć całe wyrażenie regularne , to dopasowany fragment zostanie zapisany jako $tablicaDopasowania[0], natomiast w kolejnych elementach tej tablicy zostaną zapisane fragmenty łańcucha “szukaj” do których zostały dopasowane kolejne podwyrażenia. Jedyną możliwą opcją dla parametru flags jest stała PREG_OFFSET_CAPTURE. W tym wypadku tablica dopasowań przyjmie inną formę. Każdy jej element będzie tablicą zawierającą dopasowane podwyrażenia oraz pozycję łańcucha szukaj na którym podany fragment został znaleziony. Parametr przesuniecie pozwala na rozpoczęcie dopasowywania od określonego miejsca w łańcuchu “szukaj”

- ★ Funkcja ta zwraca 1 jeżeli udało się znaleźć dopasowanie, 0 gdy się nie udało i false gdy nastąpił błąd.

9. **Zmiana fragmentów łańcuchów za pomocą wyrażeń regularnych**
   - ★ preg_replace(wzorzec, “zamiennik”, “szukaj”[, int limit=-1[, int liczba]) - funkcja ta poszukuje fragmentów łańcucha szukaj pasujących do wyrażenia regularnego wzorzec i zamienia je na łańcuch znaków “zamiennik”, Parametr limit określa maksymalną ilość zmian jakie zostaną dokonane. Domyślna liczba to –1 co oznacza ze nie zostanie ona ograniczona. Jeśli parametr liczba zostanie użyty to zostanie w nim zapisana całkowita liczba dokonanych zmian.
10. **Rozdzielanie łańcuchów znaków za pomocą wyrażeń regularnych.**
   - ★ preg_split(wzorzec, “szukaj”[, int limit=-1[, int liczba]) - funkcja ta dzieli łańcuch znakowy “szukaj” na fragmenty według wyrażenia regularnego wzorzec, po czym zwraca te fragmenty w tablicy. Parametr limit ogranicza liczbę fragmentów jakie zostaną zapisane w tablicy wynikowej. Wartość -1 oznacza że ilość nie zostanie niczym ograniczona.
   - ★ Parametr flags może przyjąć jedną lub więcej z poniższych stałych, należy je wtedy łączyć operatorem alternatywy bitowej “|”
     - PREG_SPLIT_NO_EMPTY – zwracane będą jedynie niepuste fragmenty
     - PREG_SPLIT_DELIM_CAPTURE – zwracane będą tez fragmenty użyte do podziału łańcucha wejściowego na części
     - PREG_SPLIT_OFFSET_CAPTURE - będą zwracane również informacje o miejscach występowania poszczególnych fragmentów łańcucha wejściowego podobnie jak w użyciu funkcji preg_match()
11. *Ćwiczenie 1 - użyj wyrażeń regularnych zamiast funkcji łańcuchowych do wykonania ćwiczenia 3 z zajęć 7.*
12. *Ćwiczenie 2 – dokonaj sprawdzenia poprawności adresu e-mail klienta*