Składnia języka PHP

1. Informacje ogólne o składni języka PHP

PHP jest narzędziem i językiem służącym do pisania dynamicznie generowanych stron internetowych. Kod PHP jest zwykle wbudowywany w kod HTML w związku z czym istnieją specjalne znaczniki ograniczające bloki PHP.

Różne sposoby oznaczania bloku kodu PHP w HTML

```
<? echo "Przykład użycia krótkich znaczników PHP<br>"; ?>
<?php echo "Przykład użycia pełnych znaczników PHP<br>"; ?>
<script language="php">
  echo "Przykład użycia instrukcji przetwarzania<br>";
</script>
<% echo "Przykład użycia znaczników w stylu ASP<br>"; %>
```

W dalszej części będziemy używać pełnych znaczników czyli:

```
<?php echo "<B>Pełne znaczniki PHP</B><br>"; ?>
```

PHP jest syntaktycznie bardzo podobny do języka C. Na przykład poszczególne instrukcje oddzielane są średnikiem

```
<?php
   echo "<I>Pierwsza instrukcja</I><br>";
   echo "<B>Druga instrukja</B><br>";
?>
```

Komentarze

Komentarze w PHP można oznaczać symbolami komentarzy pochodzącymi z C, C++ lub skryptów Uniksa.

```
<?php
   echo "Linia 1<br>"; // To jest komentarz jednowierszowy w stylu C++
   echo "Linia 2<br>"; # To jest komentarz w stylu skryptów Uniksa
   /* To jest wielowierszowy blok komentarza
      echo "Ta linia nie zostanie wykonana";
   */
?>
```

2. Typy danych

W języku PHP istnieją następujące typy danych: liczby całkowite, liczby zmiennoprzecinkowe, ciągi, tablice i obiekty. Typ zmiennej, w odróżnieniu od języka C, nie jest ustalany jawnie przez programistę lecz w oparciu o kontekst w jakim jest użyta zmienna. Należy na to zwrócić szczególną uwagę, gdyż niejawna konwersja typów może spowodować trudne do znalezienia błędy.
Aby łatwo można było zapanować nad typami, PHP posiada kilka funkcji do określania typów jak np.: `gettype()`, `settype()`, `is_integer()`, `is_array()`, itp.

Zmienne w PHP są oznaczane znakiem dolara (`$`), po którym następuje nazwa zmiennej. Wielkość liter w nazwach zmiennych jest rozróżniana. Prawidłowe nazwy zmiennych muszą zaczynać się literą lub znakiem podkreślenia, po których może nastąpić litera, liczba lub znak podkreślenia. Prawidłowymi literami w zmiennych są a-z, A-Z lub dowolne znaki ASCII z zakresu 127-255 (0x7f-0xff).

```php
<?php
   $zmienna1 = "Ala";
   $zmienna2 = "Ola";
   print( "$zmienna1, $zmienna2<br>" ); // wypisuje "Ala, Ola"

   $1variable = 123;  // nieprawidłowa nazwa zmiennej
   $_test = "test";   // prawidłowo, rozpoczyna się podkreśleniem
   $_ąęć = "test2";   // prawidłowo
?>
```

Liczby całkowite i zmiennoprzecinkowe

Liczby całkowite można podawać używając notacji dziesiętnej, ósemkowej i szesnastkowej, zaś liczby zmiennoprzecinkowe przy użyciu notacji zwykłej lub naukowej. A oto przykłady wspomnianych notacji.

```php
<?php
   $int1 = 123;           // liczba dziesiętna
   $int2 = 0173;          // ósemkowy zapis liczby 123
   $int3 = 0x7B;          // szesnastkowy zapis liczby 123
   $float1 = 123.123      // zwykły zapis liczby zminnoprzecinkowej
   $float2 = 1.23123e2;   //liczba zmiennoprzecinkowa w notacji naukowej
?>
```

Stringi (ciągi znaków)

Stringi w PHP są ograniczane apostrofami (‘) lub cudzysłowami (”), przy czym zapisy te różnią się sposobem interpretacji. Jeżeli string jest otoczony cudzysłowami, zmienne zapisane w stringu zamieniane są na ich wartości, natomiast w stringach otoczonych apostrofami zmienne nie są zastępowane.

```php
<?php
   $string1 = ”To jest pierwszy string”;
   $string2 = ”To jest drugi string”;

   // wyświetlenie zawartości stringów
   print(”$string1 <br> $string2 <br><br>”);

   // wyświetlenie nazw zmiennych
   print(‘$string1 <br> $string2 <br>’);
?>
```

Podobnie jak w języku C w stringach można umieszczać znaki specjalne poprzedzając je znakiem ukośnika (\).
W stringach otoczonych cudzysłowami stosuje się następujące znaki specjalne:

Sekwencja znaków	Znaczenie
\n	nowa linia
\r	powrót karetki (CR)
\t	tabulacja
\\	lewy ukośnik
\"	cudzysłów
\$	znak dolara

W stringach otoczonych apostrofami dopuszczalnymi znakami specjalnymi są lewy ukośnik (\) i apostrof (\’).

```
<?php
   $string1 = ”Kowalski”;
   $string2 = ”Nazywam się \”$string1\””;
   $string3 = ”Nazywam się \$string1”;

   print(”$string1<br>$string2<br>$string3<br>”);
   /* zostanie wypisany tekst:
      Kowalski
      Nazywam się ”Kowalski”
      Nazywam się $string1
   */
?>
```

Stringi można łączyć ze sobą przy użyciu operatora kropki (.).

```
<?php
   $string1 = ”Treść stringu 1”;
   $string2 = ”Treść stringu 2”;

   $string3 = $string1 . ” oraz ” . $string2;
   print(”$string3” . ”<br>i coś jeszcze”);
   /* zostanie wypisany tekst:
      Treść stringu 1 oraz Treść stringu 2
      i coś jeszcze
   */
?>
```

Z powodu ulotnej natury typów w PHP, zmienne mogą zmieniać swój typ w zależności od kontekstu w jakim występują.
Liczby mogą być konwertowane niejawnie na stringi, jeżeli zostaną użyte jako argument operatora operującego na stringach. Stringi zaś mogą zostać skonwertowane na liczby, jeżeli będą użyte w wyrażeniach matematycznych.
Gdy PHP konwertuje string na liczbę, korzysta z następujących zasad:

- jeżeli string zaczyna się od danych numerycznych, zostaną one skonwertowane na liczbę
- jeżeli string nie zaczyna się prawidłowymi danymi liczbowymi, wartością stringu będzie zero (0)
- jeżeli dane numeryczne zawierają jeden ze znaków `.`, `e` lub `E`, wartość będzie liczbą zmiennoprzecinkową a w przeciwnym przypadku liczbą całkowitą.

Prawidłowymi danymi numerycznymi są: opcjonalny znak, po którym następuje jedna lub więcej cyfr, opcjonalna kropka dziesiętna oraz opcjonalny znak wykładnika. Znakiem wykładnika jest „e” lub „E”, po którym następuje jedna lub więcej liczb.

```
<?php
   $Zmienna1 = 123;
   print("\$Zmienna1 = $Zmienna1, typ = ".gettype($Zmienna1)."<br>");

   $Zmienna2 = $Zmienna1 . " niejawnie skonwertowane do ciągu";
   print("\$Zmienna2 = $Zmienna2, typ = ".gettype($Zmienna2)."<br>");

   // niejawna konwersja na liczbę całkowitą
   $Zmienna3 = $Zmienna2 + 1;
   print("\$Zmienna3 = $Zmienna3, typ = ".gettype($Zmienna3)."<br>");

   // niejawna konwersja na liczbę zmiennoprzecinkową
   $Zmienna3 = $Zmienna2 * 1.1;
   print("\$Zmienna3 = $Zmienna3, typ = ".gettype($Zmienna3)."<br>");

   $NieLiczba = "abc";
   $Zmienna4 = $NieLiczba * 1; // próba konwersji na liczbę, zwracane jest 0
   print("\$Zmienna4 = $Zmienna4, typ = ".gettype($Zmienna4)."<br>");
```

```
   $Liczba = "3 studentki";
   $Zmienna5 = $Liczba + 1; // konwersja $Liczba na liczbę 3
   print("\$Zmienna5 = $Zmienna5, typ = ".gettype($Zmienna5)."<br>");
?>
```

Tablice

Tablice w PHP zachowują się zarówno tak jak tablice indeksowane (podobnie jak w C++) oraz jak tablice mieszające (asocjacyjne). Operacje na tablicy indeksowanej przedstawia poniższy przykład.

```
<?php
   // Jawne tworzenie prostej tablicy indeksowanej
   $tab[0] = "Zero";
   $tab[1] = "Jeden";
   $tab[] = "Dwa";    // jawne przypisanie do indeksu (klucza) 2
   $tab[] = "Trzy";   // jawne przypisanie do indeksu (klucza) 3
   print( "$tab[3], $tab[2], $tab[1], $tab[0]<br>" );
?>
```

Tablice asocjacyjne są to tablice, w których indeksem nie są kolejne liczby naturalne łącznie z zerem, lecz dowolny ciąg znaków – tzw. klucz. Sposób tworzenia i odwoływania się do poszczególnych elementów takiej tablicy przedstawia poniższy przykład.

```
<?php
   // Tworzenie tablicy asocjacyjnej
   $kolor["niebieski"] = "#0000FF";
   $kolor["zielony"] = "#00FF00";
   $kolor["czerwony"] = "#FF0000";
   print("Wartość szesnastkowa koloru czerwonego: {$kolor['czerwony']}<br>");
?>
```

Zamiast jawnego podawania każdej wartości tablice mogą być tworzone przy użyciu funkcji `list()` lub `array()` itp. Do operacji na tablicach można wykorzystać wiele różnych funkcji wbudowanych w PHP, których opis można znaleźć w dokumentacji.

```
<?php
   // Tworzenie tej samej co poprzednio tablicy asocjacyjnej
   $kolor = array("niebieski"  =>  "#0000FF",
                  "zielony"  =>  "#00FF00",
                  "czerwony"  =>  "#FF0000");
   print("Wartość szesnastkowa koloru zielonego: {$kolor['zielony']}<br>");
?>
```

PHP pozwala również na tworzenie tablic wielowymiarowych. Z powodu unikalnej konstrukcji tablic w PHP, można indeksować jeden wymiar tablicy wielowymiarowej liczbami a inny w sposób asocjacyjny.

```php
<?php
   // Ręczne tworzenie tablicy wielowymiarowej (indeksowanej)
   $tabww[0][0] = "Zero Zero";
   $tabww[0][1] = "Zero Jeden";
   print("Wartością \$tabww[0][1] jest {$tab[0][1]}<br>");

   // Ręczne tworzenie asocjacyjnej tablicy wielowymiarowej (mieszanej)
   $kierunek["Informatyka"][0] = "Kowalski";
   $kierunek["Informatyka"][1] = "Kwiatkowska";
   $kierunek["Informatyka"][2] = "Nowak";
   $kierunek["Ekonomia"][0] = "Sowa";
   $kierunek["Ekonomia"][1] = "Tokarz";
   $kierunek["Ekonomia"][2] = "Wójcik";
   print("\$kierunek['Ekonomia'][0] = {$kierunek['Ekonomia'][0]}<br>");
?>
```

3. ZMIENNE STAŁE I OPERATORY

Wartości zmiennych mogą być przypisywane przez wartość lub przez referencję.
Gdy przypisanie jest realizowane przez wartość, obliczona wartość wyrażenia jest przepisywana do zmiennej docelowej. Po przypisaniu zmienne są niezależne i zmiana wartości w jednej nie wpływa na wartość drugiej zmiennej.
Gdy wartości są przypisywane przez referencję, nowa zmienna staje się odwołaniem do oryginalnej zmiennej. Zmiana wprowadzona do dowolnej zmiennej powoduje zmianę drugiej. Aby wykonać przypisanie przez referencję, należy poprzedzić nazwę znakiem `&`.

```php
<?php
   $zmienna1 = "Ala";
   $zmienna2 = $zmienna1;     // przypisanie  przez wartość
   print("$zmienna1, $zmienna2<br>" );  // wypisuje "Ala, Ala"
   $zmienna2 = "Ola";
   print("$zmienna1, $zmienna2<br>" );  // wypisuje "Ala, Ola"

   $zmienna3 = &$zmienna1;    // przypisanie przez referencję
   print("$zmienna1, $zmienna3<br>" );  // wypisuje "Ala, Ala"
   $zmienna3 = "Ela";
   print("$zmienna1, $zmienna3<br>" );  // wypisuje "Ela, Ela"
?>
```

Zmienne predefiniowane

Oprócz zmiennych definiowanych przez użytkownika, w PHP istnieją zmienne tworzone przez system. Lista tych zmiennych zależy od kontekstu wykonania skryptu (na przykład, czy jest uruchamiany samodzielnie, czy poprzez serwer WWW), wersji PHP i typu serwera WWW. Ponieważ lista zmiennych jest zależna od wielu czynników, niektóre z nich mogą nie być nigdy dostępne. Aby zobaczyć wszystkie zmienne dostępne w środowisku można użyć funkcji `phpinfo()`. Wykaz predefiniowanych zmiennych wraz z opisem można znaleźć w dostępnej literaturze.

Zasięg zmiennych

Zmienne globalne w PHP mają taki sam zasięg, który rozciąga się również na dołączane pliki. Wewnątrz funkcji definiowanych przez użytkownika zmienne mają zasięg lokalny. Zmienne globalne muszą być deklarowane jako globalne, aby mogły być wykorzystywane wewnątrz funkcji. PHP posiada również zmienne statyczne, które deklarowane wewnątrz funkcji zapewniają utrzymywanie swojej wartości pomiędzy kolejnymi wywołaniami funkcji.

```php
<?php
   $Globalna1 = "Zmienna globalna";

   // Dołączamy inny plik z kodem PHP, gdzie powyższa zmienna będzie dostępna
   include("plik.php");

   function Pisz()
   {
      /*
      Poniższa instrukcja wypisze tylko <br>, ponieważ zmienna $ Globalna1
      wewnątrz funkcji jest poza zasięgiem.
      */
      print("$Globalna1<br>");
   }
   Pisz();

   function Pisz2()
   {
      global $Globalna1;
      /*
      Poniższa instrukcja wypisze wartość zmiennej, ponieważ została
      zadeklarowana jako globalna.
      */
      print("$Globalna1<br>");
   }
   Pisz2();

   function FunStatyczna()
   {
      static $ZmiennaStatyczna = 0;
      print("$ZmiennaStatyczna<br>");
      $ZmiennaStatyczna++;
   }
   // Poniższe wywołania spowodują wypisanie wartosci 0, a następnie 1
   FunStatyczna();
   FunStatyczna();
?>

      --- Zawartość pliku plik.php ---
<?php
   print("$Globalna1<br>");
?>
```

Stałe

PHP posiada kilka predefiniowanych stałych oraz pozwala na definiowanie własnych. Aby zdefiniować nową stałą używa się funkcji `define()`. Stałe w PHP (w odróżnieniu od są makr w stylu C) nie mogą być wyrażeniami, lecz muszą być wartościami skalarnymi.

```php
<?php
   define("MojString", "To jest mój string");
   define("Liczba", 1);

   print("Zdefiniowaliśmy liczbę: " . Liczba . "<br>");
   print("oraz string '" . MojString . "'<br>");
?>
```

Operatory

Język PHP posiada zestaw takich samych operatorów jak C i C++. Warto jednak wspomnieć o dodatkowych operatorach, które w C nie występują.
Operator wykonania oznaczany przez znak ` (znajdujący się na jednym klawiszu ze znakiem ~) jest podobny do operatora dostępnego we wielu językach programowania powłoki. Wyrażenie otoczone znakami ` jest wykonywane na serwerze a zwracana wartość przekazywana do zmiennej.
Operator porównania === (`$a === $b`) zwracający wartość `True`, jeżeli `$a` jest równe `$b` i są one tych samych typów.
PHP posiada również **operator kontroli błędów** `@`. Gdy operator ten jest umieszczony przed wyrażeniem, nie są generowane komunikaty błędów powodowanych przez to wyrażenie.

```
<?php
   $Liczba1 = 1;
   $Liczba2 = 2;

   $Wynik  = ($Liczba1 == $Liczba2 ) ? "Wartości są równe" :
                  "Wartości są różne";
   print("$Wynik<br>");    // wypisze "Wartości są różne"

   $Wynik = (1 == "1") ? "Wartości są równe" :
               "Wartości są różne";
   print("$Wynik<br>");    // wypisze "Wartości są równe"

   $Wynik = (1 === "1") ? "Wartości są identyczne" :
                "Wartości nie są identyczne";
   print("$Wynik<br>");    // wypisze "Wartości nie są identyczne"

   /*
      Poniższy fragment powoduje przypisanie do $Listing zawartości bieżącego
      katalogu serwera, a następnie konwersję znaków nowej linii na znaczniki
      <br> i wypisanie wyników.
   */
   $Listing = `ls -l`;
   $Lista = nl2br($Listing);
   print("<br>Zawartość katalogu:<br><b>$Lista</b><br>");
?>
```

4. PROGRAMOWANIE PRZEPŁYWU STEROWANIA

PHP posiada standardowe instrukcje programowania przepływu sterowania takie jak `if` oraz pętle `while` i `for`. Składnia tych instrukcji jest identyczna ze składnią odpowiadających im instrukcji w języku C i C++, w związku z czym ograniczymy się tutaj do ich krótkiego opisu oraz stosownego przykładu.
Oprócz standardowych instrukcji PHP posiada dodatkowo dwie funkcje dołączania plików z kodem źródłowym: `include()` i `require()`, które zostaną omówione dokładniej w dalszej części.

if, else, elseif

Instrukcja `if` organizuje przepływ sterowania poprzez tworzenie rozgałęzień w programie na podstawie wyrażeń logicznych.

```
<?php
   if (1 < 2)
      print("To zostanie wydrukowane.<br>");
   else
      print("To nie zostanie wydrukowane.<br>");

   $Zmienna = 2;
   if ($Zmienna == 1)
   {
      print("\$Zmienna == 1<br>");
   }
   elseif ($Zmienna == 2)
   {
      print("\$Zmienna == 2<br>");
   }
   elseif ($Zmienna == 3)
   {
      print("\$Zmienna == 3<br>");
   }
   else
   {
      print("\$Zmienna nie jest równa 1, 2 ani 3<br>");
   }
?>
```

switch

Instrukcja `switch` upraszcza tworzenie wielokrotnych warunków. Jest ona często używana zamiast skomplikowanych konstrukcji `if...elseif...else` zawierających wiele wystąpień `elseif`. Składnia i implementacja tej instrukcji jest identyczna jak w C. Korzystnym ulepszeniem w PHP jest możliwość używania ciągów jako wyrażeń instrukcji `switch`.
Programiści Delphi i Pascala mają zwykle kłopoty z zapamiętaniem, że w konstrukcji `switch` w PHP występują instrukcje `break`. Czasami opuszczenie tej instrukcji jest wygodne. Poniższe przykłady ilustrują najczęstsze zastosowania instrukcji `switch`.

```
<?php

   // Najprostsza instrukcja switch
   $i = 2;
   switch ($i)
   {
      case 0:
         print("zero<br>");
         break;
      case 1:
         print("jeden<br>");
         break;
      case 2:
         print("dwa<br>");
         break;
      default:
         print("Nie jest to zero, jeden ani dwa<br>");
      break;
   }
```

```
   // Switch z użyciem ciągu
   $kolor= "niebieski";
   switch($kolor)
   {
      case "czerwony":
         print("#FF0000<br>");
         break;
      case "zielony":
         print("#00FF00<br>");
         break;
      case "niebieski":
         print("#0000FF<br>");
         break;
      default:
         print("inny<br>");
      break;
   }

   // przydatny przykład opuszczenia instrukcji break
   $kolor = "Czerwony";
   switch($kolor)
   {
      case "czerwony":
      case "Czerwony":
         // Poniższa instrukcja zostanie wykonana, jeżeli $kolor
         // będzie miał wartość "Czerwony" lub "czerwony"
         print("#FF0000<br>");
         break;
      case "zielony":
      case "Zielony":
         print("#00FF00<br>");
         break;
      case "niebieski":
      case "Niebieski":
         print("#0000FF<br>");
         break;
      default:
         print("inny<br>");
      break;
   }
?>
```

while

Jest to najprostszy typ pętli w PHP, która zachowuje się identycznie jak w C i innych językach wysokiego poziomu.

do .. while

Pętla ta jest podobna do `while`, jednak w pętli `do..while` warunek pętli jest sprawdzany po pierwszym przebiegu pętli. Gwarantuje to, że ciało pętli zostanie wykonane co najmniej raz.

```
<?php
   print("Liczenie w górę przy użyciu pętli <b>while</b>.<br>");
   $i = 0;
   // wypisuje liczby od 0 do 9
   while ($i < 10)
   {
      print("$i<br>");
      $i++;
   }

   print("Liczenie w dół przy użyciu pętli <b>do..while</b>.<br>");
   // wypisuje liczby od 10 do 1
   do
   {
      print("$i<br>");
      $i--;
   } while($i > 0);
?>
```

for

Pętla `for` jest najbardziej złożoną pętlą w PHP, jednak jak w poprzednich pętlach jej składnia jest identyczna ze składnią pętli `for` w języku C i wygląda następująco:

```
for (wyr1; wyr2; wyr3) instrukcja
```

```
<?php
   // Wypisuje liczby od 0 do 9
   for($i = 0; $i < 10; $i++)
   {
      print("$i<br>");
   }
?>
```

foreach

Wyrażenie `foreach` jest wygodnym sposobem na przeglądanie tablic. Podobne konstrukcje znajdują się w VBScript, Perl i innych językach. PHP posiada dwa warianty składni:

```
foreach (tablica as zmienna_wartosc) instrukcja
foreach (tablica as zmienna_klucz => zmienna_wartosc) instrukcja
```

Pierwsza postać pętli przebiega po podanej tablicy i w każdym przebiegu wartość bieżącego elementu tablicy jest przypisywana do zmiennej (`zmienna_wartosc`) a wskaźnik bieżącego elementu tablicy jest przesuwany.
Druga postać realizuje to samo, ale dodatkowo do zmiennej (`zmienna_klucz`) jest przypisywany klucz bieżącej pozycji. Jest to bardzo przydatna konstrukcja jeżeli chcemy wypisać zawartość całej tablicy asocjacyjnej, która jak wiadomo zamiast indeksów liczbowych posiada klucze. A oto przykłady wykorzystania obu konstrukcji.

```php
<?php
   $tab = array("Czerwony", "Zielony", "Niebieski");

   foreach($tab as $wartosc)
   {
      print("Bieżąca wartość to $wartosc<br>");
   }

   $tabKol = array("Czerwony" => "#FF0000",
                   "Zielony" => "#00FF00",
                   "Niebieski" => "#0000FF");
   foreach($tabKol as $klucz => $wartosc)
   {
      print("Wartość szesnastkowa $klucz to $wartosc<br>");
   }
?>
```

break i continue

PHP posiada również znane z C instrukcje `break` i `continue`, które pozwalają na dodatkowe sterowanie pętlami. Obie te instrukcje pozwalają na podanie im parametru numerycznego, który określa ilość zagłębionych pętli, które należy przerwać lub rozpocząć od początku.
Wyrażenie `break` kończy wykonanie bieżącej konstrukcji sterującej (pętli lub wyrażenia `switch`). Wyrażenie `continue` jest używane jedynie w pętlach. Powoduje ono opuszczenie pozostałych instrukcji ciała pętli i rozpoczęcie nowej iteracji.
Najczęściej instrukcje `break` i `continue` są stosowane w zagnieżdżonych pętlach. W przypadku pętli prostych, wyrażenia warunkowe są wystarczające do realizacji tych zadań.

```php
<?php
   $tab= array(4, 5, 15, 12, 7, 3, 20, 11, 31);
   $Max = 17;
   /*
   Sprawdzamy, czy istnieje w tablicy wartość większa od bieżącej wartości
   maksymalnej.
   */
   foreach($tab as $Wartosc)
   {
      /*
      Wyrażenie będzie prawdziwe, gdy osiągnięta zostanie wartość 20.
      Ponieważ wykonujemy instrukcję break, nie sprawdzamy wartości, które są
      w tablicy po wartości 20
      */
      if ($Wartosc > $Max)
      {
         $Max = $Wartosc;
         break; // możemy napisać też 'break 1;'
      }
   }
   // wypisuje "Bieżącym maksimum jest 20"
   print("Bieżącym maksimum jest $Max<br>");

   // wypisuje liczby nieparzyste od 0 do 20
   $i = 0;
   for($i = 0; $i < 20; $i++)
   {
      if(($i % 2) == 0)
         continue; // opcjonalnie 'continue 1;'
      print("$i<br>");
   }
?>
```

include i require

PHP posiada dwa mechanizmy dołączania plików zewnętrznych: `include()` i `require()`. Wyrażenie `include()` jest zwykłą funkcją PHP, natomiast `require()` jest konstrukcją językową, która posiada kilka ograniczeń. W obu przypadkach po dołączeniu pliku PHP przechodzi do trybu HTML na początku dołączanego pliku. Na końcu pliku analizator wraca do trybu PHP. Oznacza to, że dowolny kod zawarty w pliku dołączanym musi być otoczony prawidłowymi znacznikami PHP.
Funkcja `include()` jest wykonywana za każdym jej wywołaniem i może znajdować się wewnątrz pętli lub instrukcji warunkowych. Pozwala to warunkowo włączać pliki, lub włączać grupy plików przy pomocy odpowiednio skonstruowanej pętli. Funkcja `include()` pozwala również, aby dołączany plik zwracał wartość, którą można następnie przypisać do zmiennej. Przetwarzanie pliku w instrukcji `include()` kończy się, gdy zostanie napotkana instrukcja `return`.
Wyrażenie `require()` różni się tym od `include()`, że nie wchodzi w skład konstrukcji sterujących. Oznacza to, że pliki nie mogą być warunkowo dołączane za pomocą `require()`. Wyrażenie to jest wykonywane raz, jeżeli znajduje się w pętli lub nawet, jeżeli znajduje się w instrukcji warunkowej, której warunek ma wartość `False`. Inną różnicą jest to, że pliki dołączane za pomocą `require()` nie mogą zwracać wartości. Próba zwrócenia wartości w wyrażeniu `require()` powoduje błąd składni.

5. FUNKCJE

PHP pozwala na tworzenie funkcji definiowanych przez użytkownika. Funkcje nie muszą być deklarowane przed ich użyciem w kodzie PHP4. Funkcje w PHP mogą posiadać następujące cechy: zmienne nazwy funkcji, zmienna liczba argumentów, argumenty domyślne i argumenty przekazywane przez referencję. PHP pozwala na wykonywanie dowolnego kodu w ciele funkcji, włączając w to wywołania innych funkcji. Zdolność ta pozwala również na tworzenie funkcji rekurencyjnych. PHP nie pozwala na przeciążanie funkcji, nie ma również mechanizmu usuwania lub przedefiniowania wcześniej zdefiniowanych funkcji.
Domyślnie argumenty są przekazywane przez wartość. Aby przekazać argument przez referencję, należy poprzedzić nazwę zmiennej znakiem `&`. Używając argumentów domyślnych, muszą być one umieszczone po wszystkich argumentach obowiązkowych. Poniższe przykłady pokazują użycie funkcji w PHP.

```
<?php
   // prosta funkcja
   function ReturnSum( $a, $b )
   {
      return $a + $b;
   }

   // przekazanie argumentu przez referencję
   function DolaczString(&$StringBazowy, $StringDodany)
   {
      // ponieważ jest to przekazane przez referencję, wartość
      // $StringBazowy może być zmieniona poza tą funkcją
      $StringBazowy.= $StringDodany;
   }

   // wartości domyślne
   /*
      Funkcja ta może być wywołana przy użyciu jednej z postaci:
      StworzLink("href", "text");
      StworzLink("href", "text", "target");
   */
```

```
   function StworzLink($aHREF, $aText, $aTarg = "")
   {
      if ($aTarg == "")
      {
         print("<a href=\"$aHREF\">$aText</a>");
      }
      else
      {
         print("<a href=\"$aHREF\" target=\"$aTarg\">$aText</a>");
      }
   }

   print("ReturnSum(3, 5): " . ReturnSum(3, 5) . "<br>");

   $Tekst = "Ala";
   DolaczString($Tekst, " ma kota");
   print("$Tekst<br>");    // wypisuje "Ala ma kota"

   StworzLink("http://portal.wsiz.rzeszo.pl",   "Zobaczmy   naszą   ulubioną
              stronę");
   print("<br>");
   StworzLink("http://portal.wsiz.rzeszo.pl", "Zobaczmy naszą ulubioną stronę
              w nowym oknie", "_blank");
?>
```

6. KLASY I PROGRAMOWANIE OBIEKTOWE

PHP posiada zdolność tworzenia klas za pomocą składni podobnej jak w C++. PHP posiada również bardzo prostą implementację programowania obiektowego, która jest jednak wystarczająca dla większości aplikacji WWW. Dostępne jest dziedziczenie jednobazowe, nie ma dziedziczenia wielobazowego. Istnieją konstruktory klas, ale nie ma destruktorów. PHP posiada (i wymaga używania) wskaźnik `$this`, który jest stosowany do odwoływania się do metod i zmiennych obiektu.

7. ZADANIA

1. Stwórz stronę www, na której zostanie umieszczona tabelka wypełniona zdjęciami używając do tego celu PHP. Nazwy plików graficznych umieść w tablicy asocjacyjnej. Do stworzenia tabeli i jej wypełnienia użyj pętli.
2. Zmodyfikuj zdanie 1, dodając funkcję umożliwiającą określenie liczby wierszy lub kolumn dla tabeli, w której mają zostać umieszczone rysunki.