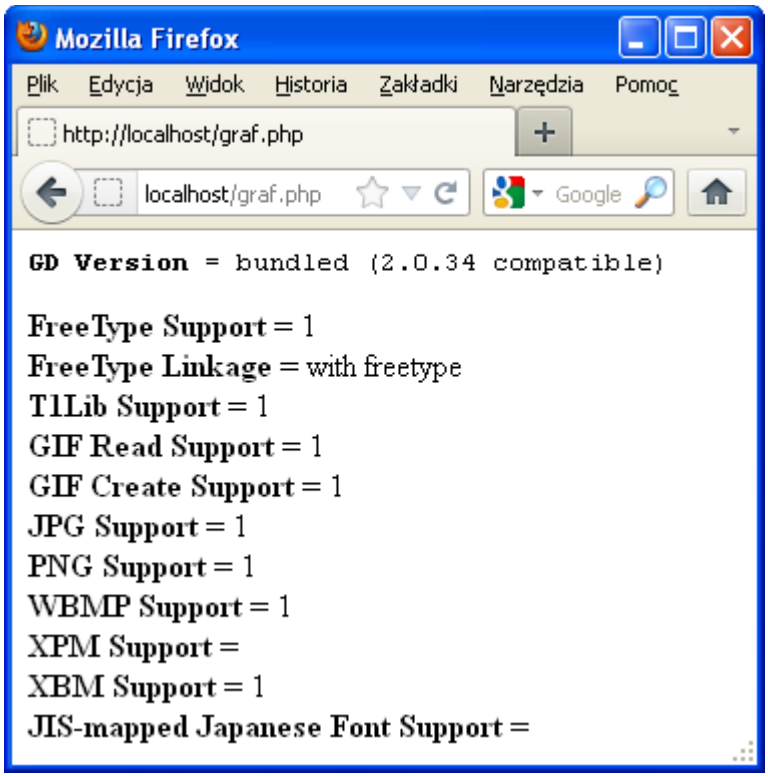
Grafika i obrazy

Biblioteka graficzna

PHP umożliwia zarówno wykonywanie różnorodnych operacji na obrazkach, jak i generowanie plików graficznych. Wraz z PHP jest w tym celu dostarczana standardowo biblioteka GD. W celu przekonania się, czy dana wersja PHP ma włączoną obsługę biblioteki GD, można użyć funkcji o nazwie gd_info. Zwraca ona tablice asocjacyjną zwracającą informacje o konfiguracji.

```php
<?php
$arr = gd_info();
foreach($arr as $key => $val){
  echo "$key = $val\n";
}
?>

```

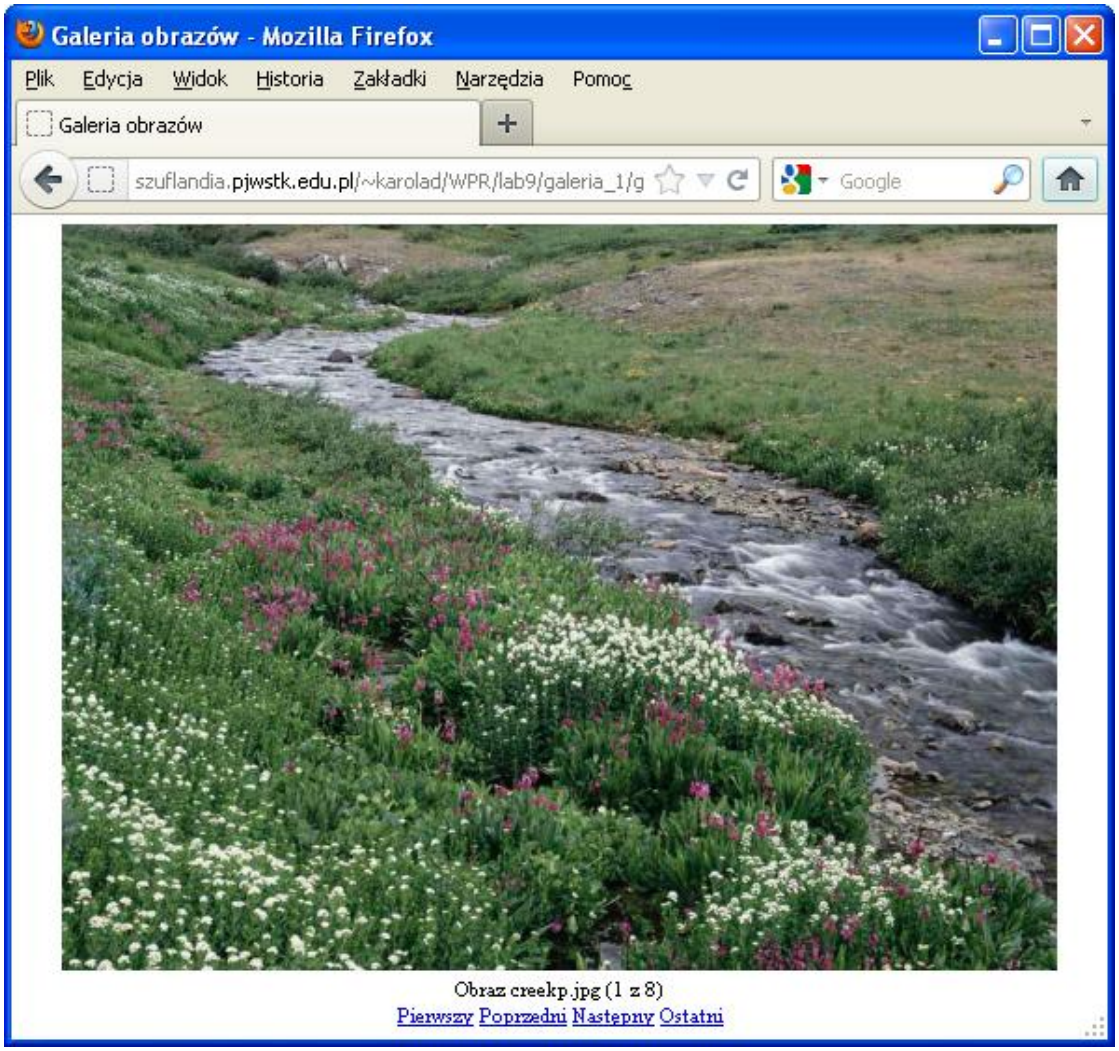


Prosta galeria

Najprostsza galeria wyświetla pojedyncze obrazy wraz z odnośnikami nawigacyjnymi. Pliki z obrazami zapiszemy w osobnym katalogu. Skrypt oczywiście uwzględni automatycznie wszystkie zmiany zawartości tego katalogu. Aby jednak nie komplikować kodu skryptu, przyjęte zostanie założenie, że katalog nie może zawierać innych plików niż pliki galerii ani podkatalogów. Będzie się w nim musiał znajdować co najmniej jeden plik graficzny.

Numer wyświetlanego obrazu będzie przekazywany do skryptu za pomocą metody GET w postaci parametru o nazwie imgid. Obrazy będą wyświetlane w kolejności alfabetycznej. Na każdej stronie galerii będą wyświetlane:

- Obraz (umieszczony na stronie za pomocą znacznika <img>),
- Nazwa pliku graficznego
- Numer obrazu
- Całkowita liczba obrazów
- Odnośnik do pierwszego, poprzedniego, następnego i ostatniego obrazu.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Galeria obrazów</title>
</head>
<body>
<?php
$imgDir = "./images";

//odczytanie parametru
if(isSet($_GET['imgid'])){
  $imgId = $_GET['imgid'];
}
else{
  $imgId = 0;
}

//odczytanie zawarto¶ci katalogu
$dir = scandir($imgDir);
array_shift($dir);
array_shift($dir);

$count = count($dir);

//sprawdzenie poprawno¶ci parametru
if($imgId < 0 || $imgId >= $count || !is_Numeric($imgId)){
  $imgId = 0;
}

//ustalenie nazwy bież±cego obrazu oraz
//identyfikatorów obrazów dla odno¶ników
$imgName = $dir["$imgId"];
$first = 0;
$last = $count - 1;
if($imgId < $count - 1){
  $next = $imgId + 1;
}
else{
  $next = $count - 1;
}

if($imgId > 0){
  $prev = $imgId - 1;
}
else{
  $prev = 0;
}
?>
<div>
<div id='obraz' style='text-align:center'>
<?php
  echo "<img src=\"$imgDir/$imgName\" alt=\"$imgName\" />";
?>
</div>
<div id='opis' style='text-align:center'>
<?php
  $imgId++;
  echo "Obraz $imgName ($imgId z $count)";
?>
</div>
<div id='nawigacja' style='text-align:center'>
<?php
  echo "<a href=\"galeria.php?imgid=$first\">Pierwszy</a> ";
  echo "<a href=\"galeria.php?imgid=$prev\">Poprzedni</a> ";
  echo "<a href=\"galeria.php?imgid=$next\">Następny</a> ";
  echo "<a href=\"galeria.php?imgid=$last\">Ostatni</a> ";
?>
</div>
</div>
</body>
</html>
```

Ćwiczenie do samodzielnego wykonania

Zmodyfikuj skrypt w taki sposób, aby zachowywał się prawidłowo, gdy w katalogu przeznaczonym do przechowywania obrazów:

a) Nie ma żadnego pliku
b) Zapisane są plik o rozszerzeniach innych niż jpg, gif, png,
c) Znajdują się podkatalogi.

Galeria z miniaturami obrazów

Nieco bardziej złożonym przykładem jest galeria zawierająca miniatury obrazów. U góry strony wyświetlana jest pewna liczba miniatur. Kliknięcie każdej z nich powoduje wyświetlenie w dolnej części strony wybranego obrazu w pełnej rozdzielczości. Pod miniaturkami znajdują się z kolei odnośniki pozwalające na nawigacje pomiędzy kolejnymi stronami miniatur.

Poliki graficzne zawierające zdjęcia w pełnej rozdzielczości zapiszemy w podkatalogu o nazwie images, natomiast miniaturki w podkatalogu thumbnails.

Zastanówmy się teraz, jak ma wyglądać struktura strony zawierającej galerię. Najprościej jest użyć kilku warstw. Górna warstwa będzie przechowywać obrazy miniatur, środkowa – odnośniki do kolejnych stron miniatur, a dolna – wybrany przez użytkownika obraz w pełnej rozdzielczości. Struktura będzie miała zatem schematyczną postać:

```
<div id='miniatury' style='text-align:center'>
<!-- lista miniatur-->
</div>
<div id='nawigacja' style='text-align:center'>
<!-- odnośniki do kolejnych stron z miniaturami-->
</div>
<div id='obraz' style='text-align:center'>
<!-- obraz w pełnej rozdzielczości-->
</div>
```

Do skryptu będą przekazywane dwa argumenty:

- pid – określający numer strony z miniaturkami, która ma zostać wyświetlona w górnej części witryny.
- iid – określający numer obrazu który ma zostać wyświetlony w dolnej części witryny.

Miniatury powinny mieć takie same nazwy jak właściwe pliki z grafiką i być według nich ustawione alfabetycznie

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Galeria obrazów</title>
</head>
<body>
<?php
$imgDir = "./images";
$thumbDir = "./thumbnails";
$thOnPage = 5;

if(isSet($_GET['iid']) && isSet($_GET['pid'])){
  $iId = intval($_GET['iid']);
  $pId = intval($_GET['pid']);
}
else{
  $iId = 0;
  $pId = 0;
}

$dir = scandir($thumbDir);
array_shift($dir);
array_shift($dir);

$count = count($dir);
$pages = ceil($count / $thOnPage);

if($iId < 0 || $iId >= $count || $pId < 0 || $pId >= $pages){
  $iId = 0;
  $pId = 0;
}
?>

<div id='miniatury' style='text-align:center'>

<?php
for($i = 0; $i < $thOnPage; $i++){
  $imgNo = $pId * $thOnPage + $i;
  if($imgNo >= $count) break;
  $imgName = $dir[$imgNo];
  $imgTag = "<img src=\"$thumbDir/$imgName\" alt=\"$imgName\" />";
  $aHead = "<a href=\"./galeria.php?pid=$pId&amp;iid=$imgNo\">";
  $aFoot = "</a>";
  echo "$aHead $imgTag $aFoot";
}
?>
</div>
<div id='nawigacja' style='text-align:center'>
IdŁ do strony:
<?php
for($i = 0; $i < $pages; $i++){
  echo "<a href=\"./galeria.php?pid=$i&amp;iid=$iId\">$i</a> ";
}
?>
</div>
<div id='obraz' style='text-align:center'>
<?php
  $imgName = $dir[$iId];
  echo "<img src=\"$imgDir/$imgName\" alt=\"$imgName\" />";
?>
</div>
</body>
</html>
```

Tworzenie obrazu

W celu utworzenia nowego obrazu w pamięci należy wykorzystać funkcje ***imagecreatetruecolor***, której jako argumenty przekazuje się szerokość i wysokość obrazu. Schematyczne wywołanie ma zatem postać:

imagecreatetruecolor(szerokość, wysokość)

Funkcja zwraca identyfikator obrazu, który pozwala na dalsze operacje. Gdy obraz utworzony za pomocą ***imagecreatetruecolor*** nie będzie już potrzebny, powinien zostać usunięty z pamięci za pomocą wywołania ***imagedestroy***.

W przypadku gdy chcemy wczytać obraz z pliku graficznego, należy użyć funkcji dedykowanej dla danego formatu graficznego:

- ***imagecreatefromgif*** – dla formatu GIF,
- ***imagecreatefromjpeg*** – dla formatu JPEG,
- ***imagecreatefrompng*** – dla formatu PNG.

Każda z nich przyjmuje jako argument nazwę pliku graficznego.

Zapisywanie plików graficznych

Obraz utworzony w pamięci może być w prosty sposób zapisany do pliku. W zależności od wybranego formatu należy użyć jednej z dedykowanych funkcji:

- ***imagegif*** – dla formatu GIF
- ***imagejpeg*** – dla formatu JPEG
- ***imagepng***– dla formatu PNG

Każda z nich przyjmuje dwa argumenty – pierwszy określa obraz, natomiast drugi nazwę pliku. Jeśli nazwa pliku zostanie pominięta, obraz jest wysyłany do standardowego wyjścia. Aby na przykład zapisać na dysku plik typu ***JPEG*** o nazwie ***image.jpg*** utworzony z obrazu o wskazywanego przez zmienną ***$img***, zastosujemy konstrukcję:

imagejpeg($img, „image.jpg”);

Funkcje ***imagejpeg*** i ***imagepng*** mogą też przyjmować trzeci argument określający jakość pliku wynikowego. W przypadku ***imagejpeg*** jest to wartość od 0 do 100. W przypadku ***imagepng*** jest to wartość od 0 do 9, gdzie 9 oznacza największą kompresję.

Kolory

Wiele funkcji operujących na obrazie wymaga podania w postaci argumenty koloru. Aby jednak móc skorzystać z danego koloru, niezbędne jest wcześniejsze jego zaalokowanie, które odbywa się przez wywołanie funkcji ***imagecolorallocate*** jej wywołanie ma postać:

imagecolorallocate($obraz,czerwony,zielony,niebieski)

Gdzie ***$obraz*** to identyfikator zwrócony przez jedną z funkcji tworzących obraz natomiast pozostałe argumenty określają poszczególne składowe koloru w formacie RGB.

Przykładowo uzyskanie indeksu koloru czerwonego dla obrazu wskazywanego przez zmienną ***$img*** będzie wymagało wywołania w postaci:

$czerwony=imagecolorallocate($img,255,0,0);

Jeśli chcemy wypełnić wybranym kolorem pewien obszar obrazu, można skorzystać z funkcji ***imagefill***, której wywołanie ma postać:

imagefill($obraz,wspx,wspy,kolor)

Gdzie ***$obraz*** określa obraz, ***wspx*** i ***wspy*** – współrzędne punktu, od którego rozpocznie się procedura wypełniania , a kolor – kolor wypełnienia.

Informacje o obrazie

Jeśli chcemy otrzymać informacje o rozmiarach obrazu, możemy skorzystać z funkcji ***imagesx*** i ***imagesy***. Pierwsza z nich zwraca szerokość, a druga wysokość obrazu. W oby przypadkach należy jako argument podać identyfikator obrazu.

Równie przydatną funkcją jest ***getimagesize***, która zwraca wiele przydatnych informacji, a nie tylko rozmiary obrazu. Jej wywołanie ma postać:

getimagesize(nazwa_pliku)

Gdzie ***nazwa_pliku*** określa nazwę pliku, z którego dane chcemy odczytać. Wartością zwracaną przez ***getimagesize*** jest tablica o następujących wartościach.

- Indeks ***0*** – szerokość obrazu w pikselach,
- Indeks ***1*** – wysokość obrazu w pikselach
- Indeks ***2*** – określenie typu pliku,
- Indeks ***3*** – ciąg znaków zapisany w postaci ***heigh="wyskość" width="szerokość",*** który określa rozmiary obrazu
- Klucz ***mime*** – ciąg znaków określający typ ***mime*** pliku
- Klucz ***channels*** – liczba kanałów (3 dla RGB, 4 dla CMYK)
- Klucz ***bits*** – liczba bitów, na których zapisywany jest kolor.

Generowanie grafiki

Na utworzonym lub wczytanym obrazie można za pomocą odpowiednich funkcji wykonywać różne operacje graficzne.

Rysowanie linii

Do rysowania linii służy funkcja ***imageline***, której wywołanie ma postać:

imageline($obraz, xp, yp, xk, yk, kolor)

gdzie:

- ***$obraz*** – to określenie obrazu,
- ***xp*** – współrzędna x początku linii,
- ***yp*** – współrządna y początku linii,
- ***xk*** – współrzędna x końca linii,
- ***yk*** – współrządna y końca linii,
- ***kolor*** – określenie koloru.

```php
<?php
$img = imagecreatetruecolor(100, 30);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zielony = imagecolorallocate($img, 0, 255, 0);
$czerwony = imagecolorallocate($img, 255, 0, 0);

imagefill($img, 0, 0, $bialy);

imageline($img, 0, 0, 99, 29, $czerwony);
imageline($img, 0, 29, 99, 0, $zielony);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie prostokątów

Do rysowania prostokątów służą funkcje ***imagerectangle*** i ***imagefilledrectangle,*** których wywołanie ma postać:

imagerectangle *($image , $x1 , $y1 , $x2 , $y2 , $color)*

imagefilledrectangle *($image , $x1 , $y1 , $x2 , $y2 , $color)*

gdzie:

obraz - Zasób obrazu, zwrócony przez jedną z funkcji tworzących obrazy, taką jak imagecreatetruecolor().

x1 - lewa górna współrzędna x
y1 - lewa górna współrzędna y 0,0 to lewy górny róg obrazka
x2 - prawa dolna współrzędna x
y2 - prawa dolna współrzędna y
color - indentyfikator koloru utworzony za pomocą funkcji imagecolorallocate()

```php
<?php
$img = imagecreatetruecolor(100, 100);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zielony = imagecolorallocate($img, 0, 255, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagefilledrectangle($img, 10, 10, 50, 50, $niebieski);
imagefilledrectangle($img, 50, 50, 90, 90, $niebieski);
imagerectangle($img, 10, 50, 50, 90, $zielony);
imagerectangle($img, 50, 10, 90, 50, $zielony);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie wielokątów

Do rysowania prostokątów służą funkcje ***imagepolygon*** i ***imagefilledpolygon***, których wywołanie ma postać:

imagepolygon($obraz, $punkty, ile, kolor)

imagefilledpolygon($obraz, $punkty, ile, kolor)

gdzie:

- ***$obraz*** – to określenie obrazu,
- ***$punkty*** – tablica zawierająca współrzędne kolejnych punktów,
- ***Ile*** – liczba wierzchołków,
- ***Kolor*** – określenie koloru.

```php
<?php
$img = imagecreatetruecolor(320, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$zolty = imagecolorallocate($img, 255, 255, 0);

imagefill($img, 0, 0, $bialy);

$tab = array(80, 100, 120, 20, 200, 20, 240, 100, 200, 180, 120, 180);

imagefilledpolygon($img, $tab, 6, $zolty);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie elips

Do rysowania elips służą funkcje ***imageelipse*** i ***imagefilledelipse***, których wywołanie ma postać:

imageelipse($obraz, xc, yc, xw, xh, kolor)

imagefilledelipse($obraz, xc, yc, xw, xh, kolor)

gdzie:

- ***$obraz*** – to określenie obrazu,
- ***xc*** – współrzędna x środka elipsy
- ***yc*** – współrzędna y środka elipsy
- ***xw*** – szerokość elipsy
- ***xh*** – wysokość elipsy
- ***kolor*** – określenie koloru.

```php
<?php
$img = imagecreatetruecolor(320, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$czarny = imagecolorallocate($img, 0, 0, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagefilledellipse($img, 100, 100, 100, 30, $niebieski);
imageellipse($img, 220, 100, 80, 80, $czarny);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```

Rysowanie wycinków elips

Do rysowania wycinków elips służą funkcje ***imagearc*** i ***imagefilledarc***, których wywołanie ma postać:

imagearc($obraz, xc, yc, xw, xh, k1, k2, kolor)

imagefilledarc($obraz, xc, yc, xw, xh, k1, k2, kolor, styl)

gdzie:

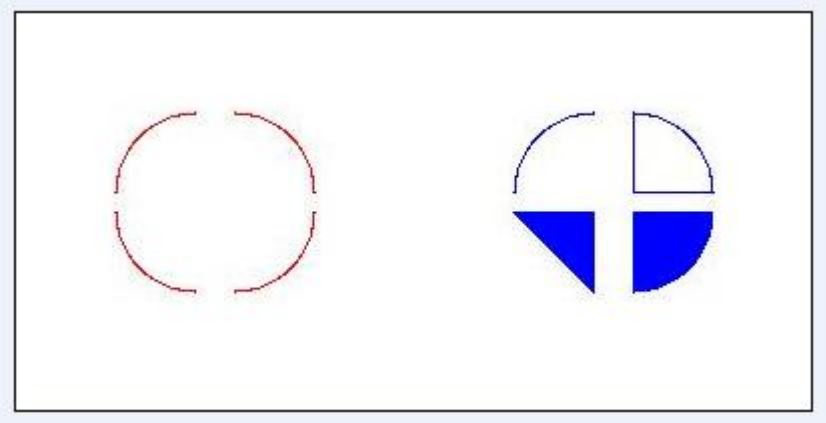
- ***$obraz*** – to określenie obrazu,

- ***xc*** – współrzędna x środka elipsy
- ***yc*** – współrzędna y środka elipsy
- ***xw*** – szerokość elipsy
- ***xh*** – wysokość elipsy
- ***k1*** – kąt określający linię początkową
- ***k2*** – kąt określający linię końcową
- ***kolor*** – określenie koloru
- ***Styl*** – styl wypełnienia.

Argument styl może składać z następujących stałych:

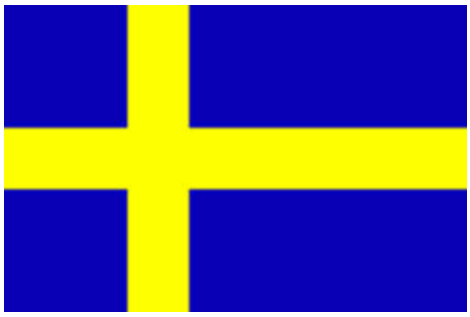
- ***IMG_ARC_PIE*** – wycinek, którego końce są połączone łukiem, standardowo wypełniony zadanym kolorem.
- ***IMG_ARC_CHORD*** – wycinek, którego końce połączone są linią prostą, standardowo wypełniony zadanym kolorem.
- ***IMG_ARC_NOFILL***– wycinek nie będzie wypełniony kolorem.
- ***IMG_ARC_EDGED*** – w połączeniu z ***IMG_ARC_NOFILL*** powoduje, że zostanie wykreślony pełny kontur wycinka.

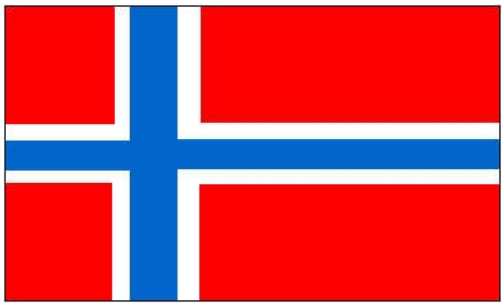
```php
<?php
$img = imagecreatetruecolor(400, 200);
$bialy = imagecolorallocate($img, 255, 255, 255);
$czerwony = imagecolorallocate($img, 255, 0, 0);
$niebieski = imagecolorallocate($img, 0, 0, 255);

imagefill($img, 0, 0, $bialy);

imagearc($img, 110, 100, 80, 80, 0, 90, $czerwony);
imagearc($img, 90, 100, 80, 80, 90, 180, $czerwony);
imagearc($img, 90, 90, 80, 80, 180, 270, $czerwony);
imagearc($img, 110, 90, 80, 80, 270, 360, $czerwony);

imagefilledarc($img, 310, 100, 80, 80, 0, 90, $niebieski,
  IMG_ARC_PIE);
imagefilledarc($img, 290, 100, 80, 80, 90, 180, $niebieski,
  IMG_ARC_CHORD);
imagefilledarc($img, 290, 90, 80, 80, 180, 270, $niebieski,
  IMG_ARC_NOFILL);
imagefilledarc($img, 310, 90, 80, 80, 270, 360, $niebieski,
  IMG_ARC_NOFILL | IMG_ARC_EDGED);

imagejpeg($img, "obraz1.jpg");
imagedestroy($img);
?>
```



Ćwiczenie do samodzielnego wykonania

Wykorzystując powyższe funkcje, stwórz skrypty generujące następujące obrazy:





Nakładanie filtrów

Wśród funkcji przetwarzających obrazy znajduje się imagefilter, która nakłada na nie jeden z dostępnych filtrów. Jej wywołanie ma postać:

imagefilter($obraz, filtr)

Filtry dostępne dla funkcji imagefilter:

IMG_FILTER_NEGATE: Zwraca wszystkie kolory obrazka

IMG_FILTER_GRAYSCALE: Konwertuje obrazek do skali szarości

IMG_FILTER_BRIGHTNESS: Zmienia jasność obrazu. Użyj arg1 do ustawienia poziomu jasności

IMG_FILTER_CONTRAST: Zmienia kontrast obrazka. Użyj arg1 do ustawienia poziomu kontrastu

IMG_FILTER_COLORIZE: Tak jak IMG_FILTER_GRAYSCALE, oczekuje określenia koloru Użyj arg1 , arg2 i arg3 w formie red , blue , green i arg4 dla kanału alpha . Zakres kolorów od 0 do 255.

IMG_FILTER_EDGEDETECT: Podkreśla krawędzie.

IMG_FILTER_EMBOSS: Uwypukla obraz

IMG_FILTER_GAUSSIAN_BLUR: Rozmazuje obraz metodą Gaussa

IMG_FILTER_SELECTIVE_BLUR: Rozmazuje obraz.

MG_FILTER_MEAN_REMOVAL: Tworzy efekt szkicu

IMG_FILTER_SMOOTH: Wygładza obraz. Użyj arg1 do ustawienia poziomu wygładzenia

IMG_FILTER_PIXELATE: Dodaje od obrazu efekt pikselowania, Użyj arg1 do ustalenia wielkości bloku i arg2 do efektu pikselowania

Skalowanie

W celu przeskalowania obrazu do zadanych rozmiarów można użyć funkcji ***imagecopyresized*** lub ***imagecopyresampled***. W rzeczywistości pobierają one określony parametrami wycinek z obrazu źródłowego i wstawiają go w miejsce obrazu docelowego. Obie funkcje przyjmują identyczne zestawy argumentów, a ich wywołanie mają postać:

imagecopyresized($obraz_docelowy, $obraz_źródłowy, xd, yd, xs, ys, wd, hd, ws, hs);

imagecopyresampled ($obraz_docelowy, $obraz_źródłowy, xd, yd, xs, ys, wd, hd, ws, hs);

gdzie:

- ***$obraz docelowy*** – określa obraz docelowy,
- ***$obraz_źródłowy*** – określa obraz źródłowy,
- ***xd*** – współrzędna x lewego górnego rogu kopiowanego obszaru w obrazie docelowym
- ***yd*** – współrzędna y lewego górnego rogu kopiowanego obszaru w obrazie docelowym
- ***xs*** – współrzędna x lewego górnego rogu kopiowanego obszaru w obrazie źródłowym

- ***ys*** – współrzędna y lewego górnego rogu kopiowanego obszaru w obrazie źródłowym
- ***wd*** – szerokość kopiowanego obszaru w obrazie docelowym
- ***hd*** – wysokość kopiowanego obszaru w obrazie docelowym
- ***ws*** – szerokość kopiowanego obszaru w obrazie źródłowym
- ***hs***– wysokość kopiowanego obszaru w obrazie źródłowym

```php
<?php
if(!$img_src = imagecreatefromjpeg("obraz1.jpg"))
  exit("Nie udało się wczytać pliku obraz1.jpg.\n");

if(!$img_dest = imagecreatetruecolor(400, 300))
  exit("Nie udało się utworzyć nowego obrazu.\n");

if(!imagecopyresampled($img_dest, $img_src, 1, 1, 320, 240, 400, 300, 160, 120))
  exit("Nie udała się operacja skalowania.\n");

if(!imagejpeg($img_dest, "obraz2.jpg"))
  exit("Wyst±pił b³±d podczas zapisu pliku obraz2.jpg.\n");

echo "Operacja skalowania zakończona sukcesem!\n";

imagedestroy($img_src);
imagedestroy($img_dest);
?>
```

Obracanie

Do obracania obrazów służy funkcja ***imagerotate***, której wywołanie ma postać:

imagerotate($obraz_źródłowy, kąt_obrotu, kolor_tła, ignoruj przezroczyste);

Obraca ona obraz wskazywany przez ***$obraz_źródłowy*** o kąt ***kąt_obrotu***, wypełniając ewentualne powstałe przy tym puste obszary kolorem ***kolor_tła***. Ustawienie argumentu ***ignoruj_przezroczyste*** na wartość inną niż 0 powoduje, że fragmenty z oznaczoną przezroczystością będą ignorowane. Funkcja zwraca przetworzony obraz. Aby zatem obrócić obraz wskazany przez zmienną ***$img*** o 45 stopni, należałoby wykonać instrukcje:

$bialy=imagecolorallocate($img, 255, 255, 255);

$obraz=imagerotate($img, 45, $bialy);

Ćwiczenia do samodzielnego wykonania

Ćwiczenie 1

Napisz skrypt, który będzie wykonywał skalowanie obrazu do zadanej rozdzielczości. Nazwa pliku oraz rozdzielczość powinny być podawane podczas działania skryptu.