ROZDZIAŁ 5.

Grafika w PHP

Być może zastanawiasz się, czym różni się grafika w PHP od grafiki w zwykłym HTML-u. Tworząc strony WWW, na pewno opatrywałeś je ilustracjami, a znacznik IMG i jego atrybuty z pewnością nie są Ci obce. Domyślasz się także, że umieszczenie znacznika IMG w danych wypisywanych przez skrypt PHP zaowocuje ilustracją na odpowiednim miejscu strony WWW. Po co więc tematowi poświęcać cały rozdział?

Odpowiedź nie jest skomplikowana. Nie będziemy zajmować się tutaj statyczną grafiką, dołączaną do dokumentów HTML z plików z ilustracjami. Natomiast omówimy w tym rozdziale *dynamiczne* tworzenie ilustracji przez skrypt działający na serwerze.

Łatwo się domyślić, jakie to niesie za sobą możliwości. Będzie można ilustrować różne informacje grafikami na podstawie zebranych danych. Nie trzeba w tym celu tworzyć wielu ilustracji i wybierać takiej, która Ci odpowiada. Po prostu utworzysz ją sobie (a raczej użytkownikowi) w czasie rzeczywistym, czyli wtedy, gdy będzie potrzebna.

Zacznijmy od pokazania, że to jest bardzo łatwe. Pomoże nam w tym poniższe ćwiczenie.

Ćwiczenie 5.1

Utwórz program, który przygotuje i wyświetli ilustrację w formacie JPG. Postaraj się, aby była to biała kratka na czarnym tle, o rozmiarach 100x100.

Nie bardzo wiadomo, jak się zabrać do takiego zadania, a tymczasem, gdy znamy metodę, jest to bardzo łatwe. Przygotowanie i wyświetlenie ilustracji odbywa się w kilku etapach.

Należy wyświetlić nagłówek dokumentu, właściwy dla formatu grafiki. W przypadku grafiki JPG jest to:

```
Content-type: image/jpeg
```

Pomaga w tym funkcja header.

header Przesyła nagłówek, określony jako parametr.

Następnie należy utworzyć rysunek, używając funkcji createimage. Funkcja ta zwraca zmienną, która jest jego identyfikatorem.

imagecreate	Tworzy rysunek. Parametry określają jego rozmiar w pikselach, a funkcja zwraca zmienną, będącą jego identyfikatorem.

Współrzędne ilustracji są liczone od lewego górnego rogu. Pierwsza współrzędna dotyczy osi poziomej, a druga – pionowej.

Na tak utworzonej ilustracji możemy dokonywać przeróżnych operacji. W tym ćwiczeniu użyjemy trzech: definiowania kolorów, wypełnienia kolorem oraz rysowania linii.

imagecolorallocate	Definiowanie koloru dla rysunku. Kolejnymi argumentami są: zmienna będąca identyfikatorem rysunku oraz składowe koloru: R, G i B (od 0 do 255). Funkcja zwraca identyfikator koloru.
imagefill	Wypełnienie rysunku (identyfikator jest pierwszym argumentem) kolorem stanowiącym czwarty argument. Drugi i trzeci argument to współrzędne, od których rozpoczyna się wypełnianie obszaru (jako obszar rozumiane są piksele jednego koloru, a jakikolwiek inny kolor stanowi brzeg).
imageline	W rysunku, którego identyfikator jest pierwszym argumentem, rysuje linię o współrzędnych początku i końca podanych w kolejnych czterech parametrach. Kolor linii jest określony szóstym parametrem.

Wyświetlenie przygotowanej ilustracji w formacie JPG jest realizowane przez funkcję imagejpeg. Istnieją także funkcje wyświetlające ilustracje w innych formatach. Pamiętaj, że musisz zachować zgodność wysyłanego nagłówka dokumentu z formatem rysunku.

imagejpg	Wyświetlenie ilustracji w formacie JPG. Argument funkcji jest identyfikatorem ilustracji.
imagegif	Wyświetlenie ilustracji w formacie GIF. Argument funkcji jest identyfikatorem ilustracji.
imagepng	Wyświetlenie ilustracji w formacie PNG. Argument funkcji jest identyfikatorem ilustracji.

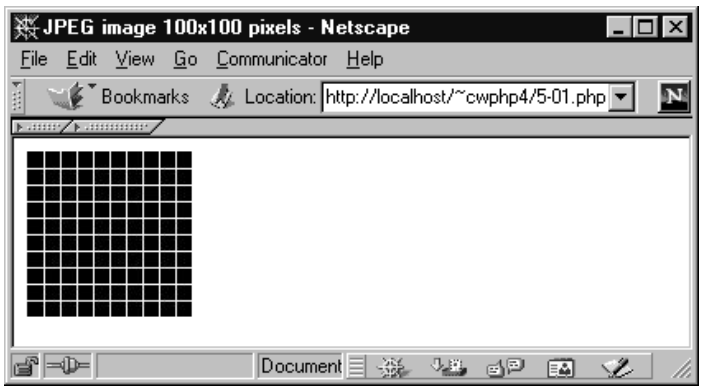
Całość programu przedstawia się następująco:

5-01.php

```
<?
  // Program tworzy ilustrację z białą kratką na czarnym tle.

  header("Content-type: image/jpeg");

  $rysunek = imagecreate (100,100);

  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);

  imagefill ($rysunek, 0, 0, $kolorczarny);

  for ($i=1; $i<10; $i++) {
```

```
    imageline ($rysunek, 10*$i, 0, 10*$i, 100, $kolorbialy);
    imageline ($rysunek, 0, 10*$i, 100, 10*$i, $kolorbialy);
  }

  imagejpeg ($rysunek);
?>
```

Efektem działania jest rysunek w formacie JPG o wymiarach 100x100, przedstawiający szachownicę.



Rysunek 5.1. Wynik działania skryptu – kratkowany JPG

Jeżeli w programie popełniłeś błąd, nie zobaczysz informacji o tym w oknie przeglądarki (wszak nagłówek, który został do niej posłany, to nagłówek graficzny, a nie tekstowy). Jeżeli ilustracja, którą chcesz wyświetlić, nie pojawi się, sprawdź źródło strony. Jeżeli program wygenerował jakiś komunikat o błędzie, tam na pewno go znajdziesz.

Można zadać pytanie, dlaczego do prostego, dwukolorowego rysunku zastosowaliśmy format JPG, a nie GIF. Ten drugi zapewniłby prawdopodobnie mniejszy plik i brak strat podczas kompresji. To prawda. Jest jednak pewien problem. Jeżeli masz zainstalowany pakiet PHPTriad, prawdopodobnie nie obsługuje on formatu GIF. Nie wierzysz? Zamień w poprzednim ćwiczeniu wiersz wyświetlający nagłówek na:

```
header("Content-type: image/gif");
```

oraz zmień polecenie wyświetlenia na:

```
imagegif ($rysunek);
```

Jeżeli ilustracja nie wyświetli się poprawnie, sprawdź źródło strony, a wszystko stanie się jasne. Ta wersja biblioteki GD (służącej do manipulowania rysunkami), której używasz, nie posiada obsługi formatu GIF.

Jeżeli korzystasz z publicznego serwera WWW z obsługą PHP, prawdopodobnie dostawca usług rozwiązał ten problem za Ciebie. Jeśli jednak wykonujesz ćwiczenia na własnym serwerze pod kontrolą systemu Windows, prawdopodobnie pojawi się mały kłopot.

Gdyby nie dało się tego problemu rozwiązać, byłaby to tragedia. Format PNG nie jest jeszcze wystarczająco rozpowszechniony, a JPG posiada swoje ograniczenia. W niektórych sytuacjach (tak się składa, w przypadku generowania grafiki po stronie serwera dzieje się tak najczęściej) istnieje potrzeba wykorzystania formatu GIF.

Na szczęście dawniej biblioteka GD obsługiwała format GIF. Wystarczy więc tę wersję, którą dysponujesz, zastąpić odpowiednią. Odpowiednią wersję dla Windows znajdziesz pod adresem: *http://aki.a4.pl/ksiazki/cwphp4/php_gd_gif_404.zip* (158 kB).

Ściągnięty plik musisz rozpakować, a następnie plik *php_gd.dll* skopiować do katalogu *php\extensions* swojej instalacji (a więc zapewne do *c:\apache\php\extensions*). Plik, który miałeś w tym katalogu do tej pory, na wszelki wypadek najpierw zarchiwizuj.

Od tego momentu powinieneś bez problemu tworzyć i wyświetlać także ilustracje w formacie GIF. Sprawdzimy, czy tak się stało w następnym ćwiczeniu.

Ćwiczenie 5.2

Utwórz podobną szachownicę, jak w ćwiczeniu 5.1. Grafikę wyświetl w formacie GIF, a każde pole szachownicy wypełnij losowym kolorem.

W pętli, dla każdego ze 100 pól szachownicy, będziemy losować kolor. Jeżeli chcemy, by wartość każdej ze składowych zawierała się w przedziale 0–255, wywołanie funkcji powinno mieć następującą postać:

```
imagecolorallocate ($rysunek, rand()%256, rand()%256, rand()%256);
```

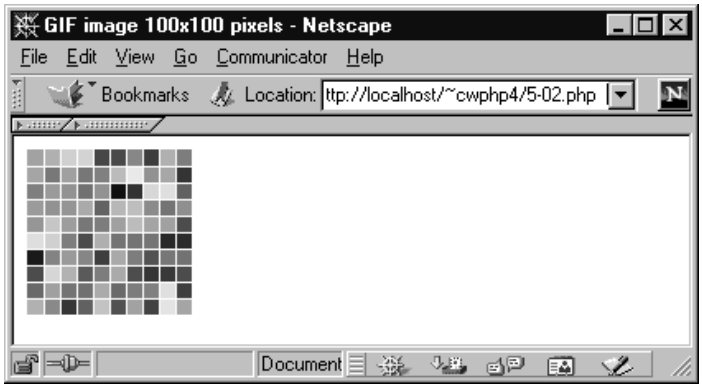
Nie należy zapominać o zainicjowaniu generatora liczb pseudolosowych funkcją srand.

5-02.php

```
<?
  // Program tworzy ilustrację z białą kratką. Każde pole
  // jest wypełnione losowym kolorem.

  header("Content-type: image/gif");
  $rysunek = imagecreate (100,100);

  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);
  imagefill ($rysunek, 0, 0, $kolorczarny);

  for ($i=1; $i<10; $i++) {
    imageline ($rysunek, 10*$i, 0, 10*$i, 100, $kolorbialy);
    imageline ($rysunek, 0, 10*$i, 100, 10*$i, $kolorbialy);
  }

  srand(time());
  for ($x=0; $x<10; $x++) {
    for ($y=0; $y<10; $y++) {
      $kolorlosowy = imagecolorallocate
          ($rysunek, rand()%256, rand()%256, rand()%256);
      imagefill ($rysunek, 5+$x*10, 5+$y*10, $kolorlosowy);
    }
  }
  imagegif ($rysunek);
?>
```

Wypełnienie każdego pola realizujemy funkcją imagefill. Kolor jest oczywiście tym wybranym losowo w każdym przebiegu pętli. Jako miejsce, od którego należy rozpocząć wypełnianie, wybieramy leżący mniej więcej w środku każdego z pól punkt o współrzędnych: (5+$x*10, 5+$y*10), gdzie $x i $y są odpowiednimi numerami w pionie i poziomie danego pola (liczonymi od 0).

Efekt działania programu przedstawia poniższa ilustracja, która z pewnością traci na efektowności z powodu braku kolorów.



Rysunek 5.2. Szachownica wypełniona losowymi kolorami

Zauważ, że w nagłówku został podany odpowiedni typ pliku:

```
header("Content-type: image/gif");
```

i wykorzystano właściwą funkcję wyświetlenia ilustracji (imagegif).

Tworząc własne grafiki przy pomocy modułu GD, możesz kontrolować kolor każdego piksela (pamiętając o ograniczeniach każdego z formatów graficznych). Spróbujemy tego dokonać w następnym ćwiczeniu.

Ćwiczenie 5.3

Napisz program, który w rysunku w formacie GIF o wymiarach 100x100 wylosuje i zakoloruje 300 punktów czerwonych i 300 czarnych (punkty w kolejnych losowaniach mogą się powtarzać).

Wykorzystamy funkcję, która określa kolor pojedynczego piksela.

imagesetpixel	Dla rysunku określonego przez pierwszy argument definiuje kolor piksela o współrzędnych podanych przez drugi i trzeci argument. Czwarty argument określa kolor.

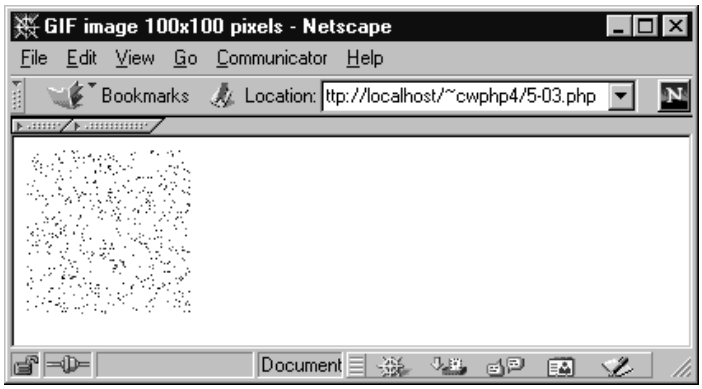
Nie musimy martwić się o powtórzenia wylosowanych punktów. W związku z tym w pętli losujemy współrzędne czerwonego i czarnego punktu.

5-03.php

```
<?
  // Program losuje 300 punktów czerwonych i czarnych i zaznacza je.

  header("Content-type: image/gif");

  $rysunek = imagecreate (100,100);

  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczerwony = imagecolorallocate ($rysunek, 255, 0, 0);
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);
```

```
  imagefill ($rysunek, 0, 0, $kolorbialy);
  srand(time());

  for ($i=1; $i<=300; $i++) {
    imagesetpixel ($rysunek, rand()%100-1, rand()%100-1, $kolorczerwony);
    imagesetpixel ($rysunek, rand()%100-1, rand()%100-1, $kolorczarny);
  }
  imagegif ($rysunek);
?>
```

Działanie programu pokazuje poniższa ilustracja (choć trudno będzie dostrzec różnicę pomiędzy czerwonymi i czarnymi punktami).



Rysunek 5.3. Losowe czerwone i czarne piksele

Istnieje wiele funkcji, które modyfikują rysunki. Warto zapoznać się z niektórymi z nich.

Ćwiczenie 5.4

Napisz program, który w rysunku w formacie GIF o wymiarach 100x100 narysuje pięć okręgów czerwonych i pięć czarnych w losowych miejscach.

Nie ma funkcji rysującej okrąg, ale skorzystamy z bardziej ogólnej, która rysuje wycinek elipsy.

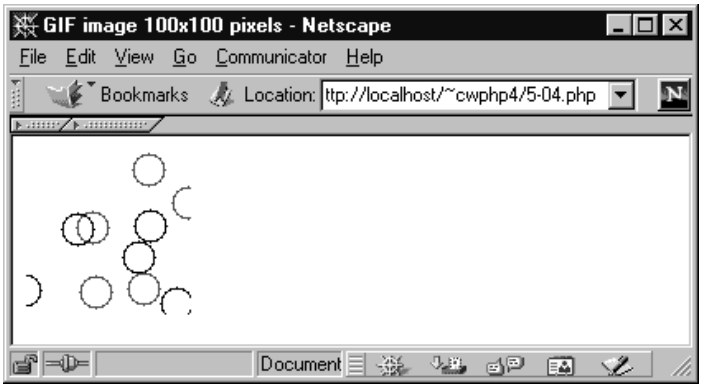
imagearc	Rysuje wycinek elipsy. Kolejnymi argumentami są: identyfikator rysunku, współrzędna x środka łuku, współrzędna y środka łuku, szerokość łuku, wysokość łuku, kąt początku, kąt końca, kolor.

Jeżeli w wywołaniu funkcji szerokość i wysokość będą sobie równe, a łuk będzie pełen (od 0 do 360 stopni), to rysowany łuk okaże się okręgiem.

5-04.php

```
<?
  // Program tworzy ilustrację z pięcioma czerwonymi i pięcioma
  // czarnymi okręgami.

  header("Content-type: image/gif");

  $rysunek = imagecreate (100,100);
```

```
  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczerwony = imagecolorallocate ($rysunek, 255, 0, 0);
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);

  imagefill ($rysunek, 0, 0, $kolorbialy);
  srand(time());

  for ($i=1; $i<=5; $i++) {
    imagearc ($rysunek, rand()%100-1, rand()%100-1,
              20, 20, 0, 360, $kolorczerwony);
    imagearc ($rysunek, rand()%100-1, rand()%100-1,
              20, 20, 0, 360, $kolorczarny);
  }
  imagegif ($rysunek);
?>
```



Rysunek 5.4. Losowe okręgi

Do rysunku można dołączać inne (albo takie, które są dostępne, albo tworzone w czasie działania skryptu).

Ćwiczenie 5.5

Napisz program, który wyświetli formularz pozwalający wpisać liczbę. Po jej wpisaniu wyświetli graficzną prezentację wpisanej liczby.

Liczbę będziemy konstruować z poszczególnych cyfr. W katalogu *cyfry* przykładów znajdziesz wszystkie cyfry w formacie GIF o rozmiarze 15x20 pikseli.

Wykorzystamy trzy nowe funkcje.

imagecreatefromgif — Tworzy rysunek na podstawie już istniejącego. Argumentem jest nazwa pliku. Funkcja zwraca identyfikator utworzonego rysunku.

imagecopyresized — Kopiuje do rysunku określonego przez pierwszy argument inny – określony przez drugi. Następne argumenty określają: współrzędne x i y miejsca, w którym należy rysunek umieścić, współrzędne x i y oraz rozmiar dx i dy w rysunku docelowym i wielkość ddx i ddy w rysunku źródłowego (pozwalające na przeskalowanie).

imagedestroy	Likwiduje identyfikator rysunku, jednocześnie zwalniając pamięć z nim związaną.

5-05.php

```
<?
  // Program wyświetla formularz, pozwalający wpisać liczbę. Jeżeli liczba
  // jest wpisana, wyświetla ją w postaci graficznej

  function printliczba ($numer) {
  // funkcja wyświetla liczbę w postaci grafiki, korzystając z grafik - cyfr,
  // zawartych w katalogu cyfry.
    $liczbacyfr=1; $l = $numer;
    while ($l >= 10) {
      $liczbacyfr++;
      $l=floor($l/10);
    }

    header("Content-type: image/gif");
    $rysunek = imagecreate (15*$liczbacyfr,20);
    for ($i=$liczbacyfr; $i>=1; $i--) {
      $cyfra = floor($numer/pow(10,$i-1));
      $numer = $numer % pow(10,$i-1);
      $rysunekcyfra = imagecreatefromgif ("cyfry/$cyfra.gif");
      imagecopyresized
        ($rysunek,$rysunekcyfra,($liczbacyfr-$i)*15,0,0,0,15,20,15,20);
      imagedestroy ($rysunekcyfra);
    }
    imagegif ($rysunek);
  }

  if ($liczba>0) { # jest wpisana jakaś wartość w formularzu
     printliczba ($liczba);
  } else { # nie ma wpisanych danych, wyświetlamy formularz
    print '<HTML>';
    print '  <HEAD>';
    print '    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; ';
    print '    charset=iso-8859-2">';
    print '    <TITLE>Liczba graficznie</TITLE>';
    print '  </HEAD>/n';
    print '  <BODY>/n';
    print '    <FORM ACTION="5-05.php" METHOD=GET>/n';
    print '    <INPUT TYPE="text" NAME="liczba">/n';
    print '    <INPUT TYPE="submit" VALUE="Wyślij">/n';
    print '    </FORM>/n';
    print '  </BODY>/n';
    print '</HTML>/n';
  }
?>
```

Kluczową rolę odgrywa oczywiście funkcja printliczba, wyświetlającą graficzną prezentację liczby, która jest jej parametrem. Warto zapoznać się z jej działaniem.

Na początku funkcja określa liczbę cyfr liczby. Czyni to poprzez zliczanie, ile raz można ją całkowicie podzielić przez 10. Wykorzystana funkcja floor podaje całkowitą część liczby, stanowiącej jej parametr.

Następnie tworzymy ilustrację o odpowiednim rozmiarze. Każda cyfra ma rozmiary 15x20, więc znając liczbę cyfr, tworzymy rysunek o rozmiarach 15*$liczbacyfr x 20.

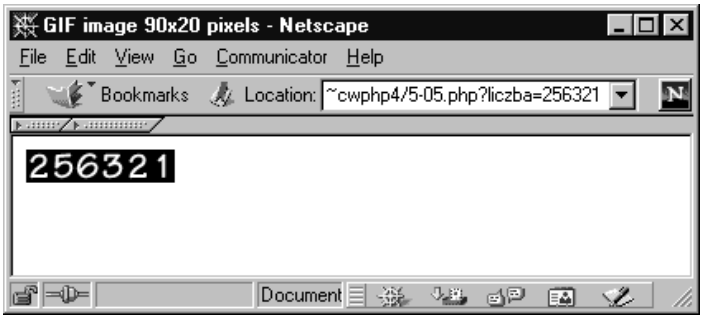
Dalej w pętli dla każdej cyfry naszej liczby przeprowadzamy pewną operację. Zwróć uwagę, jak określamy kolejną cyfrę liczby.

```
$cyfra = floor($numer/pow(10,$i-1));
$numer = $numer % pow(10,$i-1);
```

Pierwszą cyfrę liczby określamy, dzieląc ją całkowicie przez liczbę, która określa jej rząd (największą potęgę liczby 10, mniejszą od niej). Do dalszych działań odcinamy po prostu pierwszą cyfrę.

Warto zauważyć, że w pętli pojawia się inna zmienna identyfikująca rysunek: $rysunekcyfra. Wynika to z faktu, że aby skopiować jedną ilustrację do drugiej, musisz określić oba rysunki.

Po umieszczeniu ilustracji we właściwym miejscu (w docelowej grafice) warto ją usunąć z pamięci funkcją imagedestroy (mimo, że w tym przypadku program działałby tak samo bez tej funkcji).



Rysunek 5.5. Liczba 256321 wyświetlona graficznie

Do graficznej prezentacji danych nieoceniona jest funkcja, która rysuje wypełniony prostokąt.

imagefilledrectangle Rysuje wypełniony prostokąt. Argumentami są: identyfikator rysunku, współrzędne x i y lewego górnego rogu, współrzędne x i y prawego dolnego rogu oraz kolor.

Ćwiczenie 5.6

Napisz skrypt, który wypełni tablicę dziesięcioma losowymi liczbami z zakresu 0–9. Następnie zilustruj wylosowane wartości na wykresie słupkowym.

5-06.php

```
<?
  // Program tworzy wykres słupkowy na podstawie dziewięcioelementowej
  // tablicy, zapełnionej losowymi wartościami. Wykres jest podpisany.

  srand(time());
  for ($i=0; $i<10; $i++) { $liczby[$i] = rand()%10; }

  header("Content-type: image/gif");

  $rysunek = imagecreate (100,100);
  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
```

```
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);
  imagefill ($rysunek, 0, 0, $kolorbialy);

  for ($i=0; $i<10; $i++) {
    $kolorslupka = imagecolorallocate ($rysunek, 25*$i, 25*$i, 0);
    imagefilledrectangle
      ($rysunek, $i*10+3, 90-$liczby[$i]*10, $i*10+7, 90, $kolorslupka);
    imagestring ($rysunek, 1, 3+$i*10, 91, $i, $kolorczarny);
  }

  imagegif ($rysunek);
?>
```

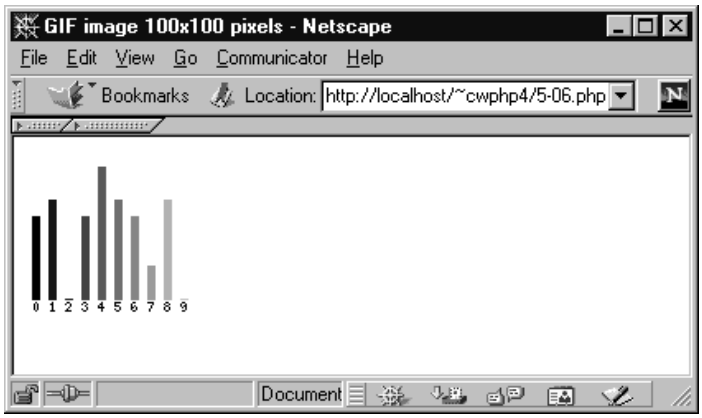
W pętli, dla każdego elementu tablicy określamy kolor słupka (sprytnie zmieniając dwie składowe koloru, powodujemy, że kolor będzie się zmieniał od czarnego do żółtego).

Następnie rysujemy słupek. Współrzędną x lewego górnego rogu określamy w zależności od wartości odpowiedniej komórki tablicy. Dzięki temu słupek będzie miał wysokość uzależnioną od tej wartości.

Oprócz tego każdy ze słupków numerujemy. Biblioteka GD pozwala na dodanie tekstu do utworzonej grafiki, warto więc skorzystać z tej możliwości. Pomocna okaże się funkcja imagestring.

imagestring	Umieszcza tekst na rysunku. Identyfikator tekstu jest pierwszym argumentem. Następne to: numer czcionki, współrzędne x i y, od których rozpoczyna się pisanie, tekst do wypisania i jego kolor.

Poniższa ilustracja pokazuje działanie programu.



Rysunek 5.6.Prezentacja danych w postaci wykresu słupkowego

Możesz wykorzystać cechę formatu GIF i stworzyć obrazek z przeplotem (ang. *interleaced*). Spowoduje to, że będzie on wyświetlany użytkownikowi ze stopniowo rosnącą rozdzielczością. Na początku użytkownik zobaczy rysunek rozmazany, a w miarę wczytywania, będzie stawała się coraz bardziej wyraźna. Dzięki temu użytkownik mniej się będzie niecierpliwił podczas wczytywania rysunku.

Stosowanie przeplotu ma sens szczególnie w przypadku ilustracji o dużych rozmiarach. Wtedy użytkownik na pewno to doceni.

Ćwiczenie 5.7

Napisz skrypt, który utworzy dużą ilustrację (o rozmiarach 800x600). Zapełnij ją wieloma różnokolorowymi kwadratami. Narysuj też linię i wypisz jakiś tekst. Wyświetl ilustrację z przeplotem.

Za wyświetlanie ilustracji z przeplotem jest odpowiedzialna funkcja imageinterlace.

imageinterlace — Wskazuje, czy rysunek określony pierwszym parametrem ma być wyświetlony z przeplotem. Jeżeli tak, drugi argument powinien mieć wartość TRUE.

5-07.php

```
<?
  // Program wyświetla utworzoną przez siebie ilustrację z przeplotem.

  srand(time());

  header("Content-type: image/gif");

  $rysunek = imagecreate (800,600);
  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);
  imagefill ($rysunek, 0, 0, $kolorbialy);

  for ($i=0; $i<250; $i++) {
    $kolorprostokata = imagecolorallocate ($rysunek, rand()%256,
                       rand()%256, rand()%256);
    imagefilledrectangle ($rysunek, rand()%800, rand()%600, rand()%800,
                          rand()%600, $kolorprostokata);

  }
  imagestring ($rysunek, 4, 310, 91, "To jest napis", $kolorczarny);
  imageline ($rysunek, 0, 0, 800, 600, $kolorczarny);
  imageinterlace ($rysunek, TRUE);
  imagegif ($rysunek);
?>
```

Celowo utworzyliśmy tak duży rysunek, aby efekt był widoczny, nawet jeżeli korzystasz z serwera WWW na własnym komputerze (dzięki czemu uzyskujesz błyskawiczny transfer). Podczas wczytywania ilustracji przeplot można zauważyć zwłaszcza na narysowanej linii i na tekście. Jednak całkiem dobrze będzie on widoczny dla każdego, kto będzie ściągał grafikę przez Internet.

Niezwykle przydatną cechą formatu GIF jest to, że pewne jego fragmenty można uczynić przezroczystymi. Dzięki temu istnieje możliwość zamieszczenia na stronie rysunku, który sprawia wrażenie, jakby posiadał kształt inny, niż prostokątny.

Biblioteka GD pozwala wprowadzić przezroczystość w rysunku w bardzo łatwy sposób. Wystarczy zdefiniować jeden z kolorów jako przezroczysty – za pomocą funkcji imagecolortransparent.

imagecolortransparent — Dla rysunku określonego pierwszym argumentem ustala kolor, który będzie przezroczysty (stanowi on drugi argument).

Ćwiczenie 5.8

Napisz skrypt, który utworzy ilustrację z czerwonym kółkiem na białym tle. Spraw, by biały kolor było przezroczysty. Wyświetl ilustrację na stronie HTML, która posiada tło innego niż biały koloru.

5-08.php

```
<?
  // Wyświetla czerwone kółko na przezroczystym tle.

  header("Content-type: image/gif");

  $rysunek = imagecreate (200,200);

  $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
  $kolorczerwony = imagecolorallocate ($rysunek, 255, 0, 0);

  imagefill ($rysunek, 0, 0, $kolorbialy);
  imagearc ($rysunek, 100, 100, 70, 70, 0, 360, $kolorczerwony);
  imagefill ($rysunek, 100, 100, $kolorczerwony);

  imagecolortransparent ($rysunek, $kolorbialy);

  imagegif ($rysunek);
?>
```

Skrypt utworzy czerwone kółko na białym tle. Nie zauważymy przezroczystości, uruchamiając go. Wykorzystajmy jednak wynik działania skryptu wyświetlając go na szarym tle w dokumencie HTML.

5-08.htm

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Rysunek z przezroczystym tłem</TITLE>
  </HEAD>

  <BODY BGCOLOR="#AAAAAA">
    <IMG SRC="5-08.php">
  </BODY>

</HTML>
```

Można zauważyć, że do utworzonej za pomocą skryptu PHP ilustracji odwołujemy się podobnie, jak do zwykłej ilustracji.

Wynikiem jest czerwone kółko na szarym tle, co świadczy o tym, że utworzony przez skrypt rysunek ma przezroczyste tło.

Bardzo cenną możliwością oferowaną przez bibliotekę jest odczytanie rozmiarów rysunku. Może to się przydać na przykład przy pisaniu skryptu automatycznie obsługującego galerię zdjęć, z których każde może być innej wielkości.

Ćwiczenie 5.9

Dla istniejącego rysunku wyświetl go w dokumencie HTML, automatycznie określając jego szerokość i wysokość, a także nadając atrybuty WIDTH i HEIGHT.

Rozmiar rysunku można określić na dwa sposoby. Pierwszy z nich wymaga najpierw powołania określającej go zmiennej. Tworząc nową ilustrację, posługiwaliśmy się funkcją createimage. W przypadku rysunku już istniejącego należy zastosować funkcję createimagefromgif.

createimagefromgif — Funkcja zwraca zmienną określającą – rysunek zawarty w pliku, którego nazwa stanowi argument funkcji.

Wielkość ilustracji określa się przy użyciu dwóch funkcji.

imagesizex — Funkcja zwraca szerokość rysunku, określonego przez identyfikator – parametr.

imagesizey — Funkcja zwraca wysokość rysunku, określonego przez identyfikator – parametr.

Inną metodą jest wykorzystanie bardzo pożytecznej funkcji getimagesize.

getimagesize — Funkcja zwraca tablicę, zawierającą cztery pola: szerokość rysunku, jego wysokość, typ oraz łańcuch znaków do wykorzystania w znaczniku IMG.

W przykładzie stosujemy obie metody.

5-09.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Odczytanie rozmiaru ilustracji</TITLE>
  </HEAD>

  <BODY>
    <? // Aplikacja daje nam możliwość odczytu, kasowania, dodawania
       // i poprawy danych w bazie.

      $nazwarysunku = "bannery/1.gif";
      $rysunek = imagecreatefromgif ($nazwarysunku);

      print 'Szerokość rysunku: <B>'.imagesx($rysunek);
      print '</B>, wysokość rysunku: <B>'.imagesy($rysunek).'</B><BR>';

      $wh = getimagesize ($nazwarysunku);
      print "<IMG SRC=\"$nazwarysunku\" $wh[3]>";

    ?>

  </BODY>
</HTML>
```

Skrypt wypisuje kod wyświetlający rysunek (za pomocą znacznika <IMG>). Wartości atrybutów WIDTH i HEIGHT dla ówcześnie odczytanego rysunku są określane odpowiednimi funk-

cjami. W naszym przypadku posługujemy się bannerem, który znajduje się w pliku z przykładami.

Wyświetlamy też tekstową informację o szerokości i wysokości rysunku.

PHP daje możliwość przeskalowania rysunku (odpowiada za to poznana już funkcja imgcopyfromgif). Należy jednak zdawać sobie sprawę, że jakość tego przeskalowania jest niższa, niż gdyby zostało dokonane za pomocą dobrego programu graficznego. Być może chciałbyś wykorzystywać w swoim serwisie jedną ilustrację w kilku różnych rozmiarach i masz zamiar przechowywać na serwerze tylko największą, a mniejsze tworzyć przy użyciu skryptu PHP. Warto się zastanowić, czy nie lepiej zamieścić ją jednak na serwerze w każdej z potrzebnych wielkości, po przekształceniu za pomocą programu graficznego. Zapewni to znacznie wyższą jakość ilustracji, przyspieszy wyświetlanie i odciąży serwer.

Ćwiczenie 5.10

Napisz skrypt, który przeskalowuje ilustrację. Porównaj jakość takiej ilustracji z przeskalowaną za pomocą programu graficznego.

Spróbujemy zmienić rozmiar jednej z okładek książek. Oryginał znajduje się w katalogu *img* przykładów. Tam także odnajdziesz ilustrację zmniejszoną za pomocą programu Corel PHOTO-PAINT.

Poniższy skrypt odczytuje już istniejącą ilustrację i kopiuje ją do tej nowo utworzonej (po pomniejszeniu).

5-10.php

```
<?
  // Program przeskalowuje ilustrację z 181x236 na 72x93.

  header("Content-type: image/gif");

  $rysunek = imagecreate (72,93);
  $rysunekprzeksztalcany = imagecreatefromgif ("img/ph6bib.gif");
  imagecopyresized
    ($rysunek,$rysunekprzeksztalcany,0,0,0,0,72,93,181,236);
  imagedestroy ($rysunekprzeksztalcany);
  imagegif ($rysunek);
?>
```

Utworzymy także dokument HTML, który wyświetli wszystkie ilustracje: wyjściową, pomniejszoną za pomocą PHP i przy zastosowaniu programu Corel PHOTO-PAINT.

5-10.htm

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Rysunek przeskalowany przez PHP i programem CorelDraw</TITLE>
  </HEAD>
  <BODY BGCOLOR="#AAAAAA">
    <TABLE>
    <TR><TD COLSPAN=2 ALIGN=CENTER><IMG SRC="img/ph6bib.gif"></TD></TR>
    <TR><TD COLSPAN=2 ALIGN=CENTER>Rysunek oryginalny<P></TD></TR>
    <TR><TD ALIGN=CENTER><IMG SRC="5-10.php"></TD>
        <TD ALIGN=CENTER><IMG SRC="img/ph6bib-corel.gif"></TD></TR>
    <TR><TD ALIGN=CENTER>PHP</TD><TD ALIGN=CENTER>Corel</TD></TR>
```

```
    </TABLE>
  </BODY>
</HTML>
```

Nietrudno zauważyć, że jakość prawej ilustracji jest znacznie lepsza, niż lewej.

Nie znaczy to, że w ogóle nie powinno się zmieniać rozmiarów za pomocą PHP. Warto jednak podchodzić do tego bardzo rozważnie.

Funkcja imagegif (a także imagejpg) zapewnia jeszcze jedną ciekawą możliwość. Rysunek – zamiast zostać wyświetlony – może zostać zapisany w odpowiednim formacie. Wystarczy podać mu drugi dokument – nazwę pliku.

Ćwiczenie 5.11

Wykorzystaj program z ćwiczenia 5.6 i zapisz utworzony przez niego wykres w pliku tekstowym.

Wykorzystamy możliwość, jaką daje nam funkcja imagegif.

5-11.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Zapis rysunku do pliku</TITLE>
  </HEAD>
  <BODY>

  <?
    // Program tworzy wykres słupkowy na podstawie dziewięcioelementowej
    // tablicy, zapełnionej losowymi wartościami. Wykres jest podpisany.
    // Utworzona ilustracja zostaje zapisana do pliku rys.gif.

    srand(time());
    for ($i=0; $i<10; $i++) {
      $liczby[$i] = rand()%10;
    }

    $rysunek = imagecreate (100,100);
    $kolorbialy = imagecolorallocate ($rysunek, 255, 255, 255);
    $kolorczarny = imagecolorallocate ($rysunek, 0, 0, 0);
    imagefill ($rysunek, 0, 0, $kolorbialy);

    for ($i=0; $i<10; $i++) {
      $kolorslupka = imagecolorallocate ($rysunek, 25*$i, 25*$i, 0);
      imagefilledrectangle
        ($rysunek, $i*10+3, 90-$liczby[$i]*10, $i*10+7, 90, $kolorslupka);
      imagestring ($rysunek, 1, 3+$i*10, 91, $i, $kolorczarny);
    }

    imagegif ($rysunek, 'img/rys.gif');
  ?>

  <P>Rysunek został zapisany do pliku.</P>

  </BODY>
</HTML>
```

W przypadku, gdy serwer działa w środowisku Windows, nie pojawią się żadne problemy. Sprawdź utworzony plik, wczytaj go do programu graficznego, by zobaczyć, że jest to rzeczywiście ilustracja.

Jeżeli serwer działa w systemie Linux, musisz pamiętać, że tworzenie pliku graficznego wiąże się z identycznymi zastrzeżeniami, jak w przypadku innych. Zostało to wyjaśnione w rozdziale 4. Katalog, w którym tworzysz plik, musi posiadać odpowiednie prawa – do zapisu, dlatego dobrze jest utworzyć do przechowywania inny katalog, niż ten, w którym przechowujesz dokumenty PHP.

Rysunki tworzone po stronie serwera mogą być bardzo cennym sposobem prezentacji różnego rodzaju dynamicznych danych na stronie. Musisz jednak pamiętać, że ich tworzenie obciąża serwer, dlatego powinieneś rozważyć, jak często dane ulegają zmianie. Jeżeli na przykład będziesz prezentował wykresy cen waluty, zmieniające się raz dziennie, nie ma sensu każdorazowo tworzyć ilustracji na podstawie danych z bazy. Można, korzystając z możliwości biblioteki GD, wygenerować je raz na dzień i wyświetlać użytkownikom przygotowany plik. Jeżeli jednak ilustracje są przygotowywane na podstawie danych wynikłych z komunikacji z użytkownikiem, nie ma wyjścia, trzeba tworzyć ilustrację za każdym razem.

Warto również pamiętać, że dobra ilustracja utworzona przez profesjonalnego grafika w dobrym programie graficznym na pewno będzie lepsza, niż przygotowana za pomocą skryptu w PHP. Należałoby więc raczej odrzucić (chyba że nie ma innej możliwości) pomysły typu: tworzenie za pomocą PHP przycisków graficznych z określonym tekstem czy generowanie losowego tła. Wykorzystaj bibliotekę GD i eksploatuj swój serwer tam, gdzie jest to uzasadnione.

Jeżeli znajdziesz „gotowce” w postaci procedur tworzących ilustracje, napisanych w języku C lub w Perlu, z wykorzystaniem biblioteki GD (która jest dla nich także dostępna), bez problemu poradzisz sobie z ich przekonwertowaniem do PHP. Metodologia działania, a nawet nazwy funkcji są bardzo podobne. W programie napisanym w języku C bibliotekę GD zidentyfikujesz po wierszu:

```
#include "gd.h"
```

a w programie napisanym w Perlu po wierszu:

```
use GD;
```

Katalog skryptów przeznaczonych do generowania grafiki – z wykorzystaniem biblioteki GD – znajduje się na stronie *http://php.resourceindex.com/Complete_Scripts/Images_and_Graphs/*. Katalog ciągle się rozwija, ale tymczasem nie jest zbyt bogaty. Można tam jednak odszukać na przykład darmowy skrypt, który tworzy kod kreskowy na podstawie podanej liczby.