

Wydruki skryptów

Rozdział 2.

Należy pamiętać, że część z poniższych skryptów realizuje zapytania do wyszukiwarek. W trakcie korzystania z nich należy zachować odpowiednią ostrożność.

spiderviewer.php

```
<html>

<head>
<title>Podgląd robota indeksującego wyszukiwarek</title>
</head>

<body>

<form name=mainform action="" method="get">
<table border="0" width="100%" align=center>
  <tr>
    <td>Enter URL: <br>
    <input type="text" name="url" size="20"></td>
  </tr>
  <tr>
    <td>
    <input type="submit" value="Kliknij, aby zobaczyć podgląd robota indeksującego"
    name="submit">
    </td>
  </tr>
</table>
</form>
<hr>

<?php

$myurl = $_GET['url'];

if (isset($myurl)) {
  print spiderViewer($myurl);
}
?>

</body>
</html>
```

```

<?php

function spiderViewer($url) {
    $finalHTML='';
    if($url) {
        $originalHTML=get_content($url);
        if($originalHTML) {
            $finalHTML.='<table border="0" align="center" width="75%">';
            $finalHTML.='<tr><td align="center" valign="top">';
            $finalHTML.='<b>Podgląd robota indeksującego dla adresu URL: ' . $url . '</b>';
            $finalHTML.='</tr>';
            $finalHTML.='<tr><td align="left" valign="top">';
            $originalHTML=preg_replace('/<script.*?>.*?</script.*?>/sim','', $originalHTML);
            $originalHTML=preg_replace('/<object.*?>.*?</object.*?>/sim','', $originalHTML);
            $originalHTML=preg_replace('/<applet.*?>.*?</applet.*?>/sim','', $originalHTML);
            $originalHTML=preg_replace('/<style.*?>.*?</style.*?>/sim','', $originalHTML);
            $originalHTML=preg_replace('/<.*?>/sim','', $originalHTML);
            $originalHTML=preg_replace('/&[#]{0,1}.[^\s]*/sim',' ', $originalHTML);
            $stopWordsArray=explode("<br />",
file_get_contents('stopwords.txt'));

            for($tmploop=0;$tmploop<count($stopWordsArray);$tmploop++) {
                $originalHTML=preg_replace('/[\W]{1,1}' . $stopWordsArray[$tmploop] .
                ' [\W]{1,1}/sim','', $originalHTML);
            }
            $originalHTML=preg_replace('/[^A-Z0-9a-z\.\?\!\;\;\,\-\r\n ]*/sim','',
            $originalHTML);
            $originalHTML=preg_replace('/[\r\n ]{2,1000}/sim',' ', $originalHTML);
            $finalHTML.= $originalHTML . '</td></tr></table>';
        } else {
            $finalHTML='Sprawdź swój adres URL.';
        }
    } else {
        $finalHTML='Wprowadzony adres URL jest niepoprawny.';
    }
    return $finalHTML;
}

function get_content($url)
{
    $ch = curl_init();
    curl_setopt ($ch, CURLOPT_URL, $url);
    curl_setopt ($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_FAILONERROR, 0);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
    curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/4.0 (compatible;MSIE 8.0; Windows NT
    6.0)');
    curl_setopt($ch, CURLOPT_TIMEOUT, 30);
    if(preg_match('/^https:\/\/\//sim',$url)==true) {
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
    }
    ob_start();
    curl_exec ($ch);
    curl_close ($ch);
    $string = ob_get_contents();
    ob_end_clean();
    return $string;
}

```

Rozdział 3.

uklad1.html

```
<html>
<head>
<style>
#nawigacja {
position: absolute;
top: 10px;
left: 50%;
width: 800px;
margin-left: 400px;
text-align: left;
}

#dane {
position: absolute;
top: 150px;
left: 50%;
width: 800px;
margin-left: 400px;
text-align: left;
}

body {
text-align: center;
min-width: 600px;
}
</style>
</head>
<body>

<div id="dane">dane<!-- W tym miejscu znajduje się treść zoptymalizowana dla wyszukiwarek --></div>
<div id="nawigacja">nawigacja<!-- W tym miejscu znajdują się elementy nawigacyjne --></div>

</body>
</html>
```

uklad2.html

```
<html>
<head>

<style>
#nawigacja {
position: absolute;
top: 0px;
left: 400;
width: 200px;
margin-left: 400px;
text-align: left;
}

#dane {
position: absolute;
top: 0px;
left: 600;
width: 600px;
margin-left: 400px;
```

```

text-align: left;
}

body {
    text-align: center;
    min-width: 800px;
}
</style>
</head>
<body>

<div id="dane">
W tym miejscu znajduje się treść zoptymalizowana dla wyszukiwarek.</div>

<div id="nawigacja"> W tym miejscu znajdują się elementy nawigacyjne.</div>

</body>
</html>

```

uklad3.html

```

<html>
<head>
<style>

#gora {
position: absolute;
top: 10px;
left: 50%;
width: 800px;
margin-left: 400px;
text-align: left;
}

#lewo {
position: absolute;
top: 150px;
left: 50%;
width: 200px;
margin-left: 400px;
text-align: left;
}

#glowna {
position: absolute;
top: 150px;
left: 50%;
width: 600px;
margin-left: 200px;
text-align: left;
}

#prawo {
position: absolute;
top: 150px;
left: 50%;
width: 200px;
margin-left: 0px;
text-align: left;
}

body {
    text-align: center;

```

```

        min-width: 800px;
    }

</style>
</head>
<body>

<div id="glowna">Zoptymalizowana część główna strony</div>

<div id="lewo">Lewy panel</div>

<div id="gora">Górny panel</div>

<div id="prawo">Prawy panel</div>

</body>

</html>

```

Rozdział 4.

rankingfactors.pl

```

#!/usr/local/bin/perl
#####
# Plik: rankingfactors.pl                                #
# Opis: Skrypt wykonuje analizę kilku czynników        #
#        wpływających na pozycję strony w rankingu,    #
#        w tym analizuje:                                #
# 1) Słowa kluczowe w tytułach stron                    #
# 2) Słowa kluczowe w nazwach domen                     #
# 3) Słowa kluczowe w treści strony                     #
# 4) Słowa kluczowe w nagłówkach                       #
# 5) Słowa kluczowe w opisach meta                     #
# 6) Bliskość słów kluczowych                         #
# 7) Słowa kluczowe w linkach zewnętrznych             #
# 8) Rozmiar strony                                    #
# 9) Liczbę słów na stronę                             #
# 10) Rozmiar witryny                                  #
# i wiele innych czynników...                           #
#
# Format: perl rankingfactors.pl 10|100 słowo(a) kluczowe #
#####

use LWP::Simple;
use LWP::UserAgent;
use HTML::TokeParser;
use HTML::TreeBuilder;
use File::Path;
use Math::Round qw(:all);

my $keyphrase = "";

my @googleLinks = ( );
my @googleTitles = ( );
my @yahooLinks = ( );
my @yahooTitles = ( );
my @bingLinks = ( );
my @bingTitles = ( );

# w razie potrzeby stwórz frazę/słowo kluczowe

```

```

foreach $argnum (1 .. $#ARGV) {
    $keyphrase = $keyphrase . "$ARGV[$argnum] ";
}
my $numres = $ARGV[0];
$keyphrase =~ s/^\s+//;
$keyphrase =~ s/\s+$//;
$keyphrase =~ s/'//g;
$keyphrase =~ s/"//g;

print "\nUruchamianie..";
# czyszczenie plików tymczasowych
rmtree( './serptemp', {keep_root => 1} );

print "\n.. czyszczenie zakończone ";
# inicjalizacja zmiennych
initializeKeyVariables($keyphrase, \@googleLinks,
                        \@googleTitles, \@yahooLinks,
                        \@yahooTitles, \@bingLinks,
                        \@bingTitles);

# zapisywanie wszystkich docelowych linków znalezionych na stronach z wynikami wyszukiwania
print "\n..pobieranie stron z wynikami wyszukiwania";
getSERPResults($#googleLinks, \@googleLinks, "google");
getSERPResults($#yahooLinks, \@yahooLinks, "yahoo");
getSERPResults($#bingLinks, \@bingLinks, "bing");
print "\n..pobrano strony z wynikami wyszukiwania";

#-----Analiza tytułów-----
# pobieranie rzeczywistych tytułów
my @googleRealTitles = ( );
my @yahooRealTitles = ( );
my @bingRealTitles = ( );
getRealTitles($#googleLinks, \@googleRealTitles, "google");
getRealTitles($#yahooLinks, \@yahooRealTitles, "yahoo");
getRealTitles($#bingLinks, \@bingRealTitles, "bing");
print "\n..pobrano rzeczywiste tytuły";

# pobranie rzeczywistych tytułów z tytułami na stronach z wynikami wyszukiwania
my @googleTitleComp = ( );
my @yahooTitleComp = ( );
my @bingTitleComp = ( );
my $percentMatchTitlesGoogle = compareArrays($#googleTitles, \@googleRealTitles,
\@googleTitles,
\@googleTitleComp);
my $percentMatchTitlesYahoo = compareArrays($#yahooTitles, \@yahooRealTitles,
\@yahooTitles, \@yahooTitleComp);
my $percentMatchTitlesBing = compareArrays($#bingTitles, \@bingRealTitles,
\@bingTitles, \@bingTitleComp);
print "\n..zakończono częściowe porównanie tytułów";

# szukanie dopasowań słów kluczowych
my @googleKeywordTitleMatch = ( );
my @yahooKeywordTitleMatch = ( );
my @bingKeywordTitleMatch = ( );
getKeywordsTitleMatch($keyphrase, \@googleRealTitles, $#googleRealTitles,
\@googleKeywordTitleMatch );

getKeywordsTitleMatch($keyphrase, \@yahooRealTitles, $#yahooRealTitles,
\@yahooKeywordTitleMatch);
getKeywordsTitleMatch($keyphrase, \@bingRealTitles, $#bingRealTitles,
\@bingKeywordTitleMatch);
print "\n..zakończono porównywanie słów kluczowych tytułów";

```

```

# sprawdzanie, czy słowa kluczowe znalezione w tytule znajdują się w treści strony
my @googleKeywordTitlePageCopy = ( );
my @yahooKeywordTitlePageCopy = ( );
my @bingKeywordTitlePageCopy = ( );
compareTitlePageCopy($#googleRealTitles, \@googleRealTitles,
\@googleKeywordTitlePageCopy, "google");
compareTitlePageCopy($#yahooRealTitles, \@yahooRealTitles,
\@yahooKeywordTitlePageCopy, "yahoo");
compareTitlePageCopy($#bingRealTitles, \@bingRealTitles,
\@bingKeywordTitlePageCopy, "bing");
print "\n..zakończono porównywanie tytułu i treści strony";

#-----Analiza nazwy domeny-----

# dopasowanie dokładne
my @googleDomainKeywordExactMatch = ( );
my @yahooDomainKeywordExactMatch = ( );
my @bingDomainKeywordExactMatch = ( );
my $percentDomainKeywordExactMatchGoogle = keywordDomainExactMatch($keyphrase,
\@googleLinks, $#googleLinks,
\@googleDomainKeywordExactMatch);
my $percentDomainKeywordExactMatchYahoo = keywordDomainExactMatch($keyphrase,
\@yahooLinks, $#yahooLinks,
\@yahooDomainKeywordExactMatch);
my $percentDomainKeywordExactMatchBing = keywordDomainExactMatch($keyphrase,
\@bingLinks, $#bingLinks,
\@bingDomainKeywordExactMatch);
print "\n..zakończono analizę dokładnych dopasowań słów kluczowych w nazwie domeny";

# dopasowanie częściowe
my @googleDomainKeywordPartialMatch = ( );
my @yahooDomainKeywordPartialMatch = ( );
my @bingDomainKeywordPartialMatch = ( );
$percentDomainKeywordPartialMatchGoogle = keywordDomainPartialMatch($keyphrase,
\@googleLinks, $#googleLinks,
\@googleDomainKeywordPartialMatch);
$percentDomainKeywordPartialMatchYahoo = keywordDomainPartialMatch($keyphrase,
\@yahooLinks, $#yahooLinks,
\@yahooDomainKeywordPartialMatch);
$percentDomainKeywordPartialMatchBing = keywordDomainPartialMatch($keyphrase,
\@bingLinks, $#bingLinks,
\@bingDomainKeywordPartialMatch);
print "\n..zakończono analizę częściowych dopasowań słów kluczowych w nazwie domeny";

#-----Analiza treści strony-----
my @googleKeywordDensity = ( );
my @yahooKeywordDensity = ( );
my @bingKeywordDensity = ( );
my $googleAvgDensity = keywordDensity($#googleLinks, $keyphrase,
\@googleKeywordDensity, "google");
my $yahooAvgDensity = keywordDensity($#yahooLinks, $keyphrase,
\@yahooKeywordDensity, "yahoo");
my $bingAvgDensity = keywordDensity($#bingLinks, $keyphrase,
\@bingKeywordDensity, "bing");

#-----Analiza znaczników opisowych META-----
my @googleDescriptionMetaExact = ( );
my @yahooDescriptionMetaExact = ( );
my @bingDescriptionMetaExact = ( );

checkExactDescriptionMeta($#googleLinks, \@googleDescriptionMetaExact,
$keyphrase, "google");

```

```

checkExactDescriptionMeta($#yahooLinks, \@yahooDescriptionMetaExact,
$keyphrase, "yahoo");
checkExactDescriptionMeta($#bingLinks, \@bingDescriptionMetaExact,
$keyphrase, "bing");

my @googleDescriptionMetaPartial = ( );
my @yahooDescriptionMetaPartial = ( );
my @bingDescriptionMetaPartial = ( );

checkPartialDescriptionMeta($#googleLinks, \@googleDescriptionMetaPartial,
$keyphrase, "google");
checkPartialDescriptionMeta($#yahooLinks, \@yahooDescriptionMetaPartial,
$keyphrase, "yahoo");
checkPartialDescriptionMeta($#bingLinks, \@bingDescriptionMetaPartial,
$keyphrase, "bing");
print "\n..zakończono analizę opisów META";

#-----Analiza znaczników nagłówków-----
my @googleNumberOfHeaderTags = ( );
my @yahooNumberOfHeaderTags = ( );
my @bingNumberOfHeaderTags = ( );
my @googleHeaderTagsKeywords = ( );
my @yahooHeaderTagsKeywords = ( );
my @bingHeaderTagsKeywords = ( );

checkHeaderTags($#googleLinks, \@googleNumberOfHeaderTags,
\@googleHeaderTagsKeywords, "google", $keyphrase);
checkHeaderTags($#yahooLinks, \@yahooNumberOfHeaderTags,
\@yahooHeaderTagsKeywords, "yahoo", $keyphrase);
checkHeaderTags($#bingLinks, \@bingNumberOfHeaderTags,
\@bingHeaderTagsKeywords, "bing", $keyphrase);
print "\n..zakończono analizę znaczników nagłówków";

#-----Analiza zagęszczenia słów kluczowych-----
my @googleKeywordPositions = ( );
my @yahooKeywordPositions = ( );
my @bingKeywordPositions = ( );
my @googleKeywordPositionsList = ( );
my @yahooKeywordPositionsList = ( );
my @bingKeywordPositionsList = ( );
analizeKeywordPositions($#googleLinks, \@googleKeywordPositions,
\@googleKeywordPositionsList, "google",
$keyphrase);
analizeKeywordPositions($#yahooLinks, \@yahooKeywordPositions,
\@yahooKeywordPositionsList, "yahoo", $keyphrase);
analizeKeywordPositions($#bingLinks, \@bingKeywordPositions,
\@bingKeywordPositionsList, "bing", $keyphrase);
print "\n..zakończono analizę zagęszczenia słów kluczowych";

#-----Analiza linków zewnętrznych-----
my @googleOutboundLinkKeywords = ( );
my @yahooKOutboundLinkKeywords = ( );
my @bingOutboundLinkKeywords = ( );
outboundLinkKeywordAnalysis($#googleLinks, \@googleLinks,
\@googleOutboundLinkKeywords, "google", $keyphrase);
outboundLinkKeywordAnalysis($#yahooLinks, \@yahooLinks,
\@yahooKOutboundLinkKeywords, "yahoo", $keyphrase);
outboundLinkKeywordAnalysis($#bingLinks, \@bingLinks,
\@bingOutboundLinkKeywords, "bing", $keyphrase);
print "\n..zakończono analizę linków zewnętrznych";

#-----Analiza PR linków zewnętrznych-----
my @googleOutboundLinksPR = ( );
my @yahooKOutboundLinksPR = ( );

```



```

my @bingOutboundLinksPR = ( );
outboundLinkPRAnalysis($#googleLinks, \@googleLinks,
\@googleOutboundLinksPR, "google", $keyphrase);
outboundLinkPRAnalysis($#yahooLinks, \@yahooLinks,
\@yahooKOutboundLinksPR, "yahoo", $keyphrase);
outboundLinkPRAnalysis($#bingLinks, \@bingLinks,
\@bingOutboundLinksPR, "bing", $keyphrase);
print "\n..zakończono analizę PR linków zewnętrznych";

#-----Analiza średniego rozmiaru strony-----
my @googlePageSize = ( );
my @yahooPageSize = ( );
my @bingPageSize = ( );
my $googleAvgPageSize = averagePageSize($#googleLinks, \@googlePageSize, "google");
my $yahooAvgPageSize = averagePageSize($#yahooLinks, \@yahooPageSize, "yahoo");
my $bingAvgPageSize = averagePageSize($#bingLinks, \@bingPageSize, "bing");
print "\n..zakończono analizę średniego rozmiaru strony";

#-----Analiza optymalnej liczby słów kluczowych-----
my @googleWords = ( );
my @yahooWords = ( );
my @bingWords = ( );
my $googleWordsPerPage = optimumWordsPerPage($#googleLinks, \@googleWords, "google");
my $yahooWordsPerPage = optimumWordsPerPage($#yahooLinks, \@yahooWords, "yahoo");
my $bingWordsPerPage = optimumWordsPerPage($#bingLinks, \@bingWords, "bing");
print "\n..zakończono analizę optymalnej liczby słów kluczowych na stronę";

#-----Analiza rozmiaru strony -----
my @googleResultsWebsiteSizes = ( );
my @yahooResultsWebsiteSizes = ( );
my @bingResultsWebsiteSizes = ( );
my $googleAverageWebSize = analyzeWebsiteSize($#googleLinks, \@googleLinks,
\@googleResultsWebsiteSizes);
my $yahooAverageWebSize = analyzeWebsiteSize($#yahooLinks, \@yahooLinks,
\@yahooResultsWebsiteSizes);
my $bingAverageWebSize = analyzeWebsiteSize($#bingLinks, \@bingLinks,
\@bingResultsWebsiteSizes);
print "\n..zakończono analizę rozmiaru strony internetowej";

#-----Analiza wieku strony-----
my @googlePageAge = ( );
my @yahooPageAge = ( );
my @bingPageAge = ( );
pageAgeAnalysis($#googleLinks, \@googleLinks, \@googlePageAge);
pageAgeAnalysis($#yahooLinks, \@yahooLinks, \@yahooPageAge);
pageAgeAnalysis($#bingLinks, \@bingLinks, \@bingPageAge);

#-----Tworzenie raportu HTML-----
# tworzenie pliku index
createIndexHTML($keyphrase);

my $numberOfLinesGoogle = $#googleLinks;
my $numberOfLinesYahoo = $#yahooLinks;
my $numberOfLinesBing = $#bingLinks;

createGoogleHTMLReport();
createYahooHTMLReport();
createBingHTMLReport();

#-----PODPROGRAMY-----
# Podprogram:
# createGoogleHTMLReport
# Opis:

```

```

# Ten podprogram tworzy plik google.html
# z podsumowaniem wyników dla wyszukiwarki Google
# Zmienne wejściowe:
# Brak
# Zmienne wyjściowe:
# Tworzy plik google.html
# Zwraca:
# Nic
sub createGoogleHTMLReport {
    #tworzenie tabeli podsumowania
    my $googleFile = "<html><head><title>Raport podsumowujący dla wyszukiwarki Google
    ↪</title>";
    $googleFile .= "<style>";
    $googleFile .=
    ↪"body, td, tr{font-family: \"Trebuchet ms\", verdana, sans-serif; font-size:9px;}";
    $googleFile .=
    ↪"b{font-family: \"Trebuchet ms\", verdana, sans-serif;font-size:10px;}";
    $googleFile .= "</style>";
    $googleFile .= "</head>";
    $googleFile .= "<body><h1>Raport podsumowujący</h1>";
    $googleFile .= "<br>";
    $googleFile .=
    ↪"<table border=\"1\" width=\"500\" cellpadding=\"2\" cellspacing=\"2\">";
    $googleFile .= "<tr><td colspan=2><b>Wartości średnie</b></td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>% dopasowania tytułu</b></td>";
    my $tmp = sprintf "%.1f", $percentMatchTitlesGoogle;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>% dokładnego dopasowania słów kluczowych w nazwie domeny
    ↪</b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordExactMatchGoogle;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>% częściowego dopasowania słów kluczowych w nazwie domeny
    ↪</b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordPartialMatchGoogle;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>% zagęszczenie słów kluczowych</b></td>";
    $tmp = sprintf "%.1f", $googleAvgDensity;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>Rozmiar strony [bajty]</b></td>";
    $tmp = sprintf "%.0f", $googleAvgPageSize;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>Liczba słów na stronę</b></td>";
    $tmp = sprintf "%.0f", $googleWordsPerPage;
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "<tr>";
    $googleFile .= "<td><b>Rozmiar witryny [bazowych adresów URL]</b></td>";
    $tmp = round($googleAverageWebSize);
    $googleFile .= "<td>$tmp</td>";
    $googleFile .= "</tr>";
    $googleFile .= "</table><br><br>";
    $googleFile .= "<b>Tabela szczegółowa</b> <br>";

```

```

$googleFile .= "<table border=1 cellpadding=2 cellspacing=2>";
$googleFile .= "<tr>";
$googleFile .= "<td nowrap>#</td>";
$googleFile .= "<td width='100'><b>URL</b></td>";
$googleFile .= "<td nowrap width='150'><b>Tytuł Google</b></td>";
$googleFile .= "<td nowrap width='150'><b>Tytuł strony</b></td>";
$googleFile .= "<td nowrap><b>Słowa kluczowe<br> znalezione w tytule? [Y|N]</b></td>";
$googleFile .= "<td nowrap><b>Słowa kluczowe <br>w treści strony[%]</b></td>";
$googleFile .= "<td nowrap><b>Dokładne dopasowania <br>w nazwie domeny</b></td>";
$googleFile .= "<td nowrap><b>Częściowe dopasowania <br>w nazwie domeny</b></td>";
$googleFile .= "<td nowrap><b>Zagęszczenie słów<br> kluczowych</b></td>";
$googleFile .= "<td nowrap><b>Dokładne dopasowania <br>w opisach META</b></td>";
$googleFile .= "<td nowrap><b>Częściowe dopasowania <br>w opisach META</b></td>";
$googleFile .= "<td nowrap><b>Znaczniki nagłówków</b></td>";
$googleFile .= "<td nowrap><b>Słowa kluczowe<br>w znacznikach nagłówków</b></td>";
$googleFile .= "<td nowrap width='350'><b>Pozycje słów kluczowych <br> na stronie  
↪</b></td>";
$googleFile .= "<td nowrap><b>Mapa zagęszczenia <br>słów kluczowych</b></td>";
$googleFile .= "<td nowrap><b>Linki zewnętrzne <br> ze słowami kluczowymi</b></td>";
$googleFile .= "<td nowrap width='150'><b>Wskaźnik PR<br> linków zewnętrznych  
↪</b></td>";
$googleFile .= "<td nowrap><b>Rozmiar strony<br>[bajty]</b></td>";
$googleFile .= "<td nowrap><b>Liczba słów<br> na stronie</b></td>";
$googleFile .= "<td nowrap><b>Rozmiar witryny <br> internetowej</b></td>";
$googleFile .= "<td nowrap><b>Wiek strony</b></td>";
$googleFile .= "</tr>";

for (my $i=0; $i < $numberOfLinesGoogle; $i++) {
    $googleFile .= "<tr>";
    $googleFile .= "<td align=left>$i<br></td>";
    $googleFile .= "<td align=left>$googleLinks[$i]<br></td>";
    $googleFile .= "<td align=left>$googleTitles[$i]<br></td>";
    $googleFile .= "<td align=left>$googleRealTitles[$i]<br></td>";
    $googleFile .= "<td align=left>$googleKeywordTitleMatch[$i]<br></td>";
    $tmp = sprintf "%.1f", $googleKeywordTitlePageCopy[$i];

    $googleFile .= "<td align=left>$tmp<br></td>";
    $googleFile .= "<td align=left>$googleDomainKeywordExactMatch[$i]<br></td>";
    $googleFile .= "<td align=left>$googleDomainKeywordPartialMatch[$i]<br></td>";
    $tmp = sprintf "%.3f", $googleKeywordDensity[$i];
    $googleFile .= "<td align=left>$tmp<br></td>";
    $googleFile .= "<td align=left>$googleDescriptionMetaExact[$i]<br></td>";
    $googleFile .= "<td align=left>$googleDescriptionMetaPartial[$i]<br></td>";
    $googleFile .= "<td align=left>$googleNumberOfHeaderTags[$i]<br></td>";
    $googleFile .= "<td align=left>$googleHeaderTagsKeywords[$i]<br></td>";
    $tmp = $googleKeywordPositionsList[$i];
    $tmp =~ s/\\|/\\, /g;
    $googleFile .= "<td align=left>$tmp<br></td>";
    $googleFile .= "<td align=left><a href='./maps/google".$i.".html'>Map</a></td>";
    printIndividualKeywordProminenceMap($i, \@googleKeywordPositions, "google");
    $googleFile .= "<td align=left>$googleOutboundLinkKeywords[$i]<br></td>";
    $googleFile .= "<td align=left>$googleOutboundLinksPR[$i]<br></td>";
    $googleFile .= "<td align=left>$googlePageSize[$i]<br></td>";
    $googleFile .= "<td align=left>$googleWords[$i]<br></td>";
    $googleFile .= "<td align=left>$googleResultsWebsiteSizes[$i]<br></td>";
    $googleFile .= "<td align=left>$googlePageAge[$i]<br></td>";
    $googleFile .= "</tr>";
}

my $filename = "./report/google.html";
open FILE, ">", "$filename" or die $!;
print FILE $googleFile;
close FILE;

```

```

}

# Podprogram:
# createYahooHTMLReport
# Opis:
# Ten podprogram tworzy plik yahoo.html
# z podsumowaniem wyników dla wyszukiwarki Yahoo
# Zmienne wejściowe:
# Brak
# Zmienne wyjściowe:
# Tworzy plik yahoo.html
# Zwraca:
# Nic
sub createYahooHTMLReport {
    # Tworzenie tabeli podsumowania
    my $yahooFile = "<html><head><title>Raport podsumowujący dla wyszukiwarki Yahoo
↳</title>";
    $yahooFile .= "<style>";
    $yahooFile .=
↳"body, td, tr{font-family: \"Trebuchet ms\", verdana, sans-serif; font-size:9px;}";
    $yahooFile .=
↳"b{font-family: \"Trebuchet ms\", verdana, sans-serif;font-size:10px;}";
    $yahooFile .= "</style>";
    $yahooFile .= "</head>";
    $yahooFile .= "<body><h1>Raport podsumowujący</h1>";
    $yahooFile .= "<br>";
    $yahooFile .=
↳"<table border=\"1\" width=\"500\" cellspacing=\"2\" cellpadding=\"2\">";
    $yahooFile .= "<tr><td colspan=2><b>Wartości średnie</b></td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b>% dopasowania tytułu</b></td>";
    my $tmp = sprintf "%.1f", $percentMatchTitlesYahoo;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b>% dokładnego dopasowania słów kluczowych w nazwie domeny
↳</b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordExactMatchYahoo;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b>% częściowego dopasowania słów kluczowych w nazwie domeny
↳</b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordPartialMatchYahoo;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b>% zagęszczenie słów kluczowych </b></td>";
    $tmp = sprintf "%.1f", $yahooAvgDensity;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b>Rozmiar strony [bajty]</b></td>";
    $tmp = sprintf "%.0f", $yahooAvgPageSize;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b> Liczba słów na stronę</b></td>";
    $tmp = sprintf "%.0f", $yahooWordsPerPage;
    $yahooFile .= "<td>$tmp</td>";
    $yahooFile .= "</tr>";
    $yahooFile .= "<tr>";
    $yahooFile .= "<td><b> Rozmiar witryny [bazowych adresów URL]</b></td>";

```

```

$tmp = round($yahooAverageWebSize);
$yahooFile .= "<td>$tmp</td>";
$yahooFile .= "</tr>";
$yahooFile .= "</table><br><br>";
$yahooFile .= "<b>Tabela szczegółowa</b> <br>";
$yahooFile .= "<table border=1 cellpadding=2 cellspacing=2>";
$yahooFile .= "<tr>";
$yahooFile .= "<td nowrap>#</td>";
$yahooFile .= "<td width='100'><b>URL</b></td>";
$yahooFile .= "<td nowrap width='150'><b>Tytuł Yahoo</b></td>";
$yahooFile .= "<td nowrap width='150'><b>Tytuł strony</b></td>";
$yahooFile .= "<td nowrap><b>Słowa kluczowe<br> znalezione w tytule? [Y|N]</b></td>";
$yahooFile .= "<td nowrap><b>Słowa kluczowe <br>w treści strony[%]</b></td>";
$yahooFile .= "<td nowrap><b>Dokładne dopasowania <br>w nazwie domeny</b></td>";
$yahooFile .= "<td nowrap><b>Częściowe dopasowania <br>w nazwie domeny</b></td>";
$yahooFile .= "<td nowrap><b>Zagęszczenie słów<br> kluczowych</b></td>";
$yahooFile .= "<td nowrap><b>Dokładne dopasowania <br>w opisach META</b></td>";
$yahooFile .= "<td nowrap><b>Częściowe dopasowania <br>w opisach META</b></td>";
$yahooFile .= "<td nowrap><b>Znaczniki nagłówków</b></td>";
$yahooFile .= "<td nowrap><b>Słowa kluczowe<br>w znacznikach nagłówków</b></td>";
$yahooFile .= "<td nowrap width='350'><b>Pozycje słów kluczowych <br>na stronie  

↳</b></td>";
$yahooFile .= "<td nowrap><b>Mapa zagęszczenia <br>słów kluczowych</b></td>";
$yahooFile .= "<td nowrap><b>Linki zewnętrzne <br> ze słowami kluczowymi</b></td>";
$yahooFile .= "<td nowrap width='150'><b>Wskaźnik PR<br> linków zewnętrznych  

↳</b></td>";
$yahooFile .= "<td nowrap><b>Rozmiar strony<br>[bajty]</b></td>";
$yahooFile .= "<td nowrap><b>Liczba słów<br> na stronie</b></td>";
$yahooFile .= "<td nowrap><b>Rozmiar witryny <br> internetowej</b></td>";
$yahooFile .= "<td nowrap><b>Wiek strony</b></td>";
$yahooFile .= "</tr>";

for (my $i=0; $i < $numberOfLinesYahoo; $i++) {
    $yahooFile .= "<tr>";
    $yahooFile .= "<td align=left>$i&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooLinks[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooTitles[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooRealTitles[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooKeywordTitleMatch[$i]&nbsp;</td>";
    $tmp = sprintf "%.1f", $yahooKeywordTitlePageCopy[$i];

    $yahooFile .= "<td align=left>$tmp&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooDomainKeywordExactMatch[$i]&nbsp;</td>";
    $yahooFile .=
    ↳ "<td align=left>$yahooDomainKeywordPartialMatch[$i]&nbsp;</td>";
    $tmp = sprintf "%.3f", $yahooKeywordDensity[$i];
    $yahooFile .= "<td align=left>$tmp&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooDescriptionMetaExact[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooDescriptionMetaPartial[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooNumberOfHeaderTags[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooHeaderTagsKeywords[$i]&nbsp;</td>";
    $tmp = $yahooKeywordPositionsList[$i];
    $tmp =~ s/\\|\/, /g;
    $yahooFile .= "<td align=left>$tmp&nbsp;</td>";
    $yahooFile .=
    ↳ "<td align=left><a href='./maps/yahoo".$i.".html'>Map</a></td>";
    printIndividualKeywordProminenceMap($i, \@yahooKeywordPositions, "yahoo");
    $yahooFile .= "<td align=left>$yahooOutboundLinkKeywords[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooOutboundLinksPR[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooPageSize[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooWords[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooResultsWebsiteSizes[$i]&nbsp;</td>";
    $yahooFile .= "<td align=left>$yahooPageAge[$i]&nbsp;</td>";
    $yahooFile .= "</tr>";
}

```

```

    }
    my $filename = "../report/yahoo.html";
    open FILE, ">", "$filename" or die $!;
    print FILE $yahooFile;
    close FILE;
}

# Podprogram:
# createBingHTMLReport
# Opis:
# Ten podprogram tworzy plik bing.html
# z podsumowaniem wyników dla wyszukiwarki Bing
# Zmienne wejściowe:
# Brak
# Zmienne wyjściowe:
# Tworzy plik bing.html
# Zwraca:
# Nic
sub createBingHTMLReport {
    # tworzenie tabeli podsumowania
    my $bingFile = "<html><head><title>Raport podsumowujący dla wyszukiwarki Bing
        </title>";
    $bingFile .= "<style>";
    $bingFile .=
        "\<b{font-family: \"Trebuchet ms\", verdana, sans-serif; font-size:9px;}\"";
    $bingFile .=
        "\<b{font-family: \"Trebuchet ms\", verdana, sans-serif;font-size:10px;}\"";
    $bingFile .= "</style>";
    $bingFile .= "</head>";
    $bingFile .= "<body><h1>Raport podsumowujący</h1>";
    $bingFile .= "<br>";
    $bingFile .=
        "\<table border=\"1\" width=\"500\" cellpadding=\"2\">";
    $bingFile .= "<tr><td colspan=2><b>Wartości średnie</b></td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>% dopasowania tytułu</b></td>";
    my $tmp = sprintf "%.1f", $percentMatchTitlesBing;
    $bingFile .= "<td>$tmp</td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>% dokładnego dopasowania słów kluczowych w nazwie domeny
        </b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordExactMatchBing;
    $bingFile .= "<td>$tmp</td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>% częściowego dopasowania słów kluczowych w nazwie domeny
        </b></td>";
    $tmp = sprintf "%.1f", $percentDomainKeywordPartialMatchBing;
    $bingFile .= "<td>$tmp</td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>% zagęszczenie słów kluczowych</b></td>";
    $tmp = sprintf "%.1f", $bingAvgDensity;
    $bingFile .= "<td>$tmp</td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>Rozmiar strony [bajty]</b></td>";
    $tmp = sprintf "%.0f", $bingAvgPageSize;
    $bingFile .= "<td>$tmp</td>";
    $bingFile .= "</tr>";
    $bingFile .= "<tr>";
    $bingFile .= "<td><b>Liczba słów na stronę</b></td>";

```

```

$tmp = sprintf "%.0f", $bingWordsPerPage;
$bingFile .= "<td>$tmp</td>";
$bingFile .= "</tr>";
$bingFile .= "<tr>";
$bingFile .= "<td><b>Rozmiar witryny [bazowych adresów URL]</b></td>";
$tmp = round($bingAverageWebSize);
$bingFile .= "<td>$tmp</td>";
$bingFile .= "</tr>";
$bingFile .= "</table><br><br>";
$bingFile .= "<b>Tabela szczegółowa</b> <br>";
$bingFile .= "<table border=1 cellpadding=2 cellspacing=2>";
$bingFile .= "<tr>";
$bingFile .= "<td nowrap>#</td>";
$bingFile .= "<td width='100'><b>URL</b></td>";
$bingFile .= "<td nowrap width='150'><b>Tytuł Bing</b></td>";
$bingFile .= "<td nowrap width='150'><b>Tytuł strony</b></td>";
$bingFile .= "<td nowrap><b>Słowa kluczowe<br> znalezione w tytule? [Y|N]</b></td>";
$bingFile .= "<td nowrap><b>Słowa kluczowe <br>w treści strony[%]</b></td>";
$bingFile .= "<td nowrap><b>Dokładne dopasowania <br>w nazwie domeny</b></td>";
$bingFile .= "<td nowrap><b>Częściowe dopasowania <br>w nazwie domeny</b></td>";
$bingFile .= "<td nowrap><b>Zagęszczenie słów<br> kluczowych</b></td>";
$bingFile .= "<td nowrap><b>Dokładne dopasowania <br>w opisach META</b></td>";
$bingFile .= "<td nowrap><b>Częściowe dopasowania <br>w opisach META</b></td>";
$bingFile .= "<td nowrap><b>Znaczniki nagłówków</b></td>";
$bingFile .= "<td nowrap><b>Słowa kluczowe<br>w znacznikach nagłówków</b></td>";
$bingFile .= "<td nowrap width='350'><b>Pozycje słów kluczowych <br> na stronie  
↳</b></td>";
$bingFile .= "<td nowrap><b>Mapa zagęszczenia <br>słów kluczowych</b></td>";
$bingFile .= "<td nowrap><b>Linki zewnętrzne <br> ze słowami kluczowymi</b></td>";
$bingFile .= "<td nowrap width='150'><b>Wskaźnik PR<br> linków zewnętrznych</b></td>";
$bingFile .= "<td nowrap><b>Rozmiar strony<br>[bajty]</b></td>";
$bingFile .= "<td nowrap><b>Liczba słów<br> na stronie</b></td>";
$bingFile .= "<td nowrap><b>Rozmiar witryny <br> internetowej</b></td>";
$bingFile .= "<td nowrap><b>Wiek strony</b></td>";
$bingFile .= "</tr>";

for (my $i=0; $i < $numberOfLinesBing; $i++) {
    $bingFile .= "<tr>";
    $bingFile .= "<td align=left>$i&nbsp;</td>";
    $bingFile .= "<td align=left>$bingLinks[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingTitles[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingRealTitles[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingKeywordTitleMatch[$i]&nbsp;</td>";
    $tmp = sprintf "%.1f", $bingKeywordTitlePageCopy[$i];
    $bingFile .= "<td align=left>$tmp&nbsp;</td>";
    $bingFile .= "<td align=left>$bingDomainKeywordExactMatch[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingDomainKeywordPartialMatch[$i]&nbsp;</td>";
    $tmp = sprintf "%.3f", $bingKeywordDensity[$i];
    $bingFile .= "<td align=left>$tmp&nbsp;</td>";
    $bingFile .= "<td align=left>$bingDescriptionMetaExact[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingDescriptionMetaPartial[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingNumberOfHeaderTags[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingHeaderTagsKeywords[$i]&nbsp;</td>";
    $tmp = $bingKeywordPositionsList[$i];
    $tmp =~ s/\\|/\\, /g;
    $bingFile .= "<td align=left>$tmp&nbsp;</td>";
    $bingFile .= "<td align=left><a href='./maps/bing.$i.'.html'>Map</a></td>";
    printIndividualKeywordProminenceMap($i, \@bingKeywordPositions, "bing");
    $bingFile .= "<td align=left>$bingOutboundLinkKeywords[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingOutboundLinksPR[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingPageSize[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingWords[$i]&nbsp;</td>";
    $bingFile .= "<td align=left>$bingResultsWebsiteSizes[$i]&nbsp;</td>";
}

```

```

        $bingFile .= "<td align=left>$bingPageAge[$i]&nbsp;  </td>";
        $bingFile .= "</tr>";
    }
    my $filename = "./report/bing.html";
    open FILE, ">", "$filename" or die $!;
    print FILE $bingFile;
    close FILE;
}

# Podprogram:
# createIndexHTML
# Opis:
# Ten podprogram tworzy fragment kodu HTML dla pliku index
# szukając ostatnio zmodyfikowanego łańcucha znaków
# Zmienne wejściowe:
# $keyword => słowo kluczowe
# Zmienne wyjściowe:
# Tworzy plik index.html
# Zwraca:
# Nic
sub createIndexHTML {
    my $keyword = shift;

    my $indexFile = "<html><head><title>Raport podsumowujący</title></head>";
    $indexFile .= "<body><center><strong>Raport podsumowujący";
    $indexFile .= " (for \"$keyword\") <br><br>";
    $indexFile .=
    ↪ "<a href=\"#$\" onclick=\"document.all.myiframe.src='google.html'\">";
    $indexFile .= "Google</a> |";
    $indexFile .=
    ↪ "<a href=\"#$\" onclick=\"document.all.myiframe.src='yahoo.html'\">";
    $indexFile .= "Yahoo!</a> |";
    $indexFile .=
    ↪ "<a href=\"#$\" onclick=\"document.all.myiframe.src='bing.html'\">";
    $indexFile .= "Bing</a><br><br>";
    $indexFile .= "Kliknij na link, aby wyświetlić raport...<br><br>";
    $indexFile .=
    ↪ "<iframe name='myiframe' width=5000 height=6000 border='0' frameborder='0'>";
    $indexFile .= "</iframe></center></body></html>";

    my $filename = "./report/index.html";
    open FILE, ">", "$filename" or die $!;
    print FILE $indexFile;
    close FILE;
}

# Podprogram:
# pageAgeAnalysis
# Opis:
# Ten podprogram przegląda wszystkie adresy URL znalezione
# na stronie z wynikami wyszukiwania szukając ostatnio modyfikowanego łańcucha
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy linków
# $srcArr => macierz (odwołanie) do macierzy linków
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic
sub pageAgeAnalysis {
    my ($numberOfElements, $srcArr, $destArr) = @_;

    for(my $i=0; $i<$numberOfElements; $i++) {
        #print "\nprzetwarzanie: $srcArr->[$i]";
    }
}

```



```

        my $ua = new LWP::UserAgent;
        $ua->agent("Mozilla/3.0 (compatible)");
        my $request = new HTTP::Request("GET", "$srcArr->[$i]");
        my $response = $ua->request($request);
        my $code=$response->code;
        $destArr->[$i]= scalar(localtime($response->last_modified)),
        #print "\n$destArr->[$i]";
    }
}

# Podprogram:
#  analizeWebsiteSize
# Opis:
#  Ten podprogram przegląda strony z wynikami wyszukiwania Google,
#  aby sprawdzić rozmiar różnych witryn
# Zmienne wejściowe:
#  $numberOfElements => liczba plików do przetworzenia
#  $destArr => macierz (odwołanie) do macierzy linków
#  $srcArr => macierz (odwołanie) do macierzy linków
# Zmienne wyjściowe:
#  Brak
# Zwraca:
#  Zwraca średni rozmiar witryny
sub analizeWebsiteSize {
    my ($numberOfElements, $srcArr, $destArr) = @_;
    # tworzy linki za pomocą polecenia "site:"
    my $ua = new LWP::UserAgent;
    my $res;
    ua->timeout(25);
    $ua->agent("Mozilla/3.0 (compatible)");
    my $total = 0;

    for($i=0; $i<$numberOfElements; $i++){

        my $filename = "./serptemp/temp.txt";
        my $url = $srcArr->[$i];
        # Najpierw pobieramy bazowe adresy URL

        if($url =~ /^http/) {
            my @tmparr1 = split (/\//,$url);
            my @tmparr2 = split (/\//,$tmparr1[1]);
            my $baseurl = "";
            if($#tmparr2>0) {
                $baseurl = $tmparr2[0];
            }else {
                $baseurl = $tmparr1[1];
            }
            $baseurl =~ s/\//$/;
            $url = $baseurl;
        }

        my $tmpurl =
        ↪ 'http://www.google.com/search?hl=en&q=site%3A' . $url . '&btnG=Wyszukaj';
        my $randNum = int(rand(5));
        #print "\nOczekiwanie przez $randNum sekund.\n";
        sleep($randNum);
        $res = $ua->get("$tmpurl",':content_file' => "$filename");
        # Pobieranie treści strony z wynikami wyszukiwania Google
        my $pageCopy = "";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            # Pobieranie treści strony dla tego pliku
            while (my $token = $p->get_tag("body")) {
                $pageCopy = $p->get_text("/body");
            }
        }
    }
}

```

```

    }
  }else {
    print "\nplik nie istnieje";
  }
  #Dzielenie pliku za pomocą tekstu "z około <b>"
  my $separator1 = 'z około ';

  my @tempArr1 = split(/$separator1/, $pageCopy);
  my $separator2 = 'b';
  my @tempArr2 = split(/$separator2/, $tempArr1[1]);
  my $separator3 = ' for';
  my @tempArr3 = split(/$separator3/, $tempArr2[0]);

  my $size = $tempArr3[0];

  # Usunięcie przecinka z liczby
  $size =~ s/,//g;

  # Zapisywanie dla tego adresu URL
  $destArr->[$i] = $size;
  $total = $total + $size;
}
# Obliczanie i zwracanie wartości średniej
if ($total>0) {
  return ($total/$numberOfElements);
} else {
  return 0;
}
}

```

Podprogram:

optimumWordsPerPage

Opis:

Ten podprogram wykonuje pętlę po wszystkich plikach, aby

w macierzy wynikowej zapisać rozmiary stron.

Zmienne wejściowe:

\$numberOfElements => liczba plików do przetworzenia

\$destArr => macierz (odwołanie) do macierzy linków

\$prefix => Prefiks pliku wyszukiwarki

Zmienne wyjściowe:

Brak

Zwraca:

Zwraca średnią liczbę słów na stronę

```

sub optimumWordsPerPage {
  my ($numberOfElements, $destArr, $prefix) = @_;
  my $total = 0;
  for(my $i=0; $i< $numberOfElements; $i++) {
    my $filename = './serptemp/' . $prefix . "$i.txt";
    my $tree = HTML::TreeBuilder->new;
    $tree->parse_file("$filename");
    my $non_html = $tree->as_text();
    $non_html =~ s/^s+//g;
    my @tempSizeArr = split(/ /,$non_html);
    $destArr->[$i] = $#tempSizeArr;
    $total = $total + $#tempSizeArr;
  }
  return ($total/$numberOfElements);
}

```

Podprogram:

averagePageSize

Opis:

Ten podprogram wykonuje pętlę po wszystkich plikach, aby

w macierzy wynikowej zapisać rozmiary stron.

```

# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy linków
# $prefix => Prefiks pliku wyszukiwarki
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Zwraca średni rozmiar strony
sub averagePageSize {
    my ($numberOfElements, $destArr, $prefix) = @_;
    my $total = 0;
    for(my $i=0; $i< $numberOfElements; $i++) {
        my $filename = './serptemp/' . $prefix . "$i.txt";
        my $filesize = -s "$filename";
        $destArr->[$i] = $filesize;
        $total = $total + $destArr->[$i];
    }
    return ($total/$numberOfElements);
}

# Podprogram:
# outboundLinkPRAnalysis
# Opis:
# Ten podprogram przeprowadza analizę składniową wartości PR
# z głównych domen wszystkich zewnętrznych linków
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $srcLinksArr => macierz (odwołanie) do macierzy linków
# $prefix => Prefiks pliku wyszukiwarki
# Zmienne wyjściowe:
# Drukuje mapę słów kluczowych
# Zwraca:
# Nic
sub outboundLinkPRAnalysis {
    my ($numberOfElements, $srcLinksArr, $destArr, $prefix) = @_;
    my $PRURL = 'http://www.seowarrior.net/scripts/pr.php?pr=';
    my $range = 2;
    # Pętla przez wszystkie pliki
    for(my $i=0; $i< $numberOfElements; $i++) {
        my $filename = './serptemp/' . $prefix . "$i.txt";
        my %linkHash = ();
        my $PRs = "";
        # Sprawdzanie, czy plik istnieje
        if (-e "$filename") {
            my $p = HTML::TokeParser->new($filename);
            while (my $token = $p->get_tag("a")) {
                # Pobieranie linku i tekstu kotwiczki
                my $url = $token->[1]{href} || "-";
                my $text = $p->get_trimmed_text("/a");
                # Sprawdzanie, czy link jest zewnętrzny, czy wewnętrzny
                if($url =~ /^http/) {
                    my @tmparr1 = split (/\//, $url);
                    my @tmparr2 = split (/\./, $tmparr1[1]);
                    my $tmpbaseURLChild = $tmparr2[0] . $tmparr2[1];

                    my @tmparr3 = split (/\//, $srcLinksArr->[$i]);
                    my @tmparr4 = split (/\./, $tmparr3[1]);
                    my $tmpbaseURLParent = $tmparr4[0] . $tmparr4[1];

                    my @tmparr5 = split (/\//, $tmparr1[1]);
                    my $baseurl = "";
                    if($#tmparr5>0) {
                        $baseurl = $tmparr5[0];
                    }else {

```



```

if($url =~ /^http/) {
    @tmparr1 = split (/\:\/\/,$url);
    @tmparr2 = split (/\./,$tmparr1[1]);
    $tmpbaseURLChild = $tmparr2[0] . $tmparr2[1];

    @tmparr3 = split (/\:\/\/,$srcLinksArr->[$i]);
    @tmparr4 = split (/\./,$tmparr3[1]);
    $tmpbaseURLParent = $tmparr4[0] . $tmparr4[1];
    if($tmpbaseURLChild ne $tmpbaseURLParent) {
        # Link zewnętrzny.. przetwarzanie
        if($#keywordFragments > 0){
            # Obsługa wielu słów kluczowych
            for(my $j=0; $j <= $#keywordFragments; $j++){
                # Sprawdzenie dopasowania
                if($text =~ /$keywordFragments[$j]/i) {
                    # Dopasowanie znalezione
                    $foundCount++;
                    last;
                }
            }
        } else {

            if($text =~ /$keyword/i) {
                # Dopasowanie znalezione
                $foundCount++;
            }
        }
    }
    $total++;
}
}
else {
    #print "\nPlik: $filename nie został znaleziony!";
}
if($total>0) {
    $destArr->[$i] = ( $foundCount);
} else {
    $destArr->[$i] = 0;
}
#print "\n$destArr->[$i]";
}
}

# Podprogram:
# printKeywordProminenceMap
# Opis:
# Ten podprogram drukuje mapę każdego adresu URL
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $srcArr => macierz (odwołanie) do macierzy linków
# Zmienne wyjściowe:
# Drukuj mapę słów kluczowych
# Zwraca:
# Nic
sub printKeywordProminenceMap {
    my ($srcArr, $numberOfElements) = @_ ;
    for(my $i; $i<$numberOfElements; $i++){
        print "$srcArr->[$index]\n";
    }
}

# Podprogram:
# printIndividualKeywordProminenceMap

```

```

# Opis:
# Ten podprogram drukuje mapę każdego adresu URL
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $srcArr => macierz (odwołanie) do macierzy wyników
# Zmienne wyjściowe:
# Drukuję mapę słów kluczowych
# Zwraca:
# Nic
sub printIndividualKeywordProminenceMap {
    my ($index, $srcArr, $prefix) = @_;
    my $filename = "./report/maps/$prefix".$index.".html";
    open FILE, ">", "$filename" or die $!;
    print FILE "<html><head><title>\n";
    print FILE "Mapa wag słów kluczowych\n";
    print FILE "</title></head>\n";
    print FILE "<body><table width=400 cellpadding=2 cellspacing=0><tr><td width=400>";
    print FILE $srcArr->[$index];
    print FILE "</td></tr></table></body></html>";
    close FILE;
}

# Podprogram:
# analyzeKeywordPositions
# Opis:
# Ten podprogram analizuje relatywne pozycje słów kluczowych w treści witryny
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy wyników
# $keyword => słowo kluczowe do analizy
# $prefix => prefiks pliku
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach

sub analyzeKeywordPositions {
    my ($numberOfElements, $destArr, $destArr2, $prefix, $keyword) = @_;
    my @keywordFragments = split(/ /,$keyword);
    # Pętla przez wszystkie pliki do pobrania
    for(my $i=0; $i< $numberOfElements; $i++) {
        my $pageCopy = "";
        my $tmpMap = ":";
        my $filename = './serptemp/' . $prefix . "$i.txt";
        # Sprawdzanie, czy plik istnieje
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            # Pobranie treści strony dla tego pliku
            while (my $token = $p->get_tag("body")) {
                $pageCopy = $p->get_trimmed_text("/body");
                $pageCopy = cleanText($pageCopy);
            }
            $pageCopy =~ s/s+/ /g;
            my @tempArr = split(/ /, $pageCopy);
            $totalWords = $#tempArr;
            #print "\nSuma wyrazów na tej stronie: $totalWords";
            # Pętla przez wszystkie wyrazy
            for(my $j=0; $j < $totalWords; $j++){
                my $flag = "N";
                if($#keywordFragments > 0){
                    # Obsługa wielu słów kluczowych
                    for(my $k=0; $k <= $#keywordFragments; $k++){
                        # Sprawdzanie dopasowania
                        if($tempArr[$j] =~ /$keywordFragments[$k]/i) {

```

```

        # Aktualizacja zmiennej docelowej indeksem macierzy słów kluczowych
        $destArr->[$i] .= "$k ";
        # Aktualizacja zmiennej docelowej pozycją względną
        $destArr2->[$i] = $destArr2->[$i] . "$j" . "|";
        $flag = "T";
        last;
    } else {
        if( ($k == $keywordFragments) && ($flag ne "T") ) {
            $destArr->[$i] .= "* ";
        }
    }
}
} else {
    # Obsługa pojedynczego słowa kluczowego
    $tempArr[$j] =~ s/"//;
    $tempArr[$j] =~ s/'//;

    if($tempArr[$j] =~ /$keyword/i){
        $destArr->[$i] .= "0 ";
        $destArr2->[$i] = $destArr2->[$i] . "$j" . "|";
        $flag = "T";
    } else {
        $destArr->[$i] .= "* ";
    }
}
if($flag ne "N") {
    $destArr->[$i] .= "* ";
}
}
# print "\n\n$destArr->[$i]";
} else {
    print "\nPlik nie istnieje";
}
}
}
}

```

Podprogram:

checkHeaderTags

Opis:

Ten podprogram sprawdza, czy używane są znaczniki nagłówek, a jeśli są, to czy

w nagłówkach używane są słowa kluczowe.

Zmienne wejściowe:

\$numberOfElements => liczba plików do przetworzenia

\$destArr1 => macierz (odwołanie) do macierzy wyników

\$destArr2 => macierz (odwołanie) do macierzy wyników

\$keyword => słowo kluczowe do analizy

\$prefix => prefiks pliku

Zmienne wyjściowe:

Brak

Zwraca:

Nic, cała praca wykonywana na przekazywanych macierzach

```

sub checkHeaderTags {
    my ($numberOfElements, $destArr1, $destArr2, $prefix, $keyword) = @_;
    my @keywordFragments = split(/ /,$keyword);

```

```

    for(my $i=0; $i < $numberOfElements; $i++) {
        my $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            my $h1Text = "";
            my $h2Text = "";
            my $h3Text = "";
            my $h4Text = "";
            my $h5Text = "";

```

```

my $h6Text = "";
my $separator = '|s|e|p|a|r|a|t|o|r|';
while(my $token = $p->get_token) {
    if($token->[0] eq 'S' and $token->[1] eq 'h1') {
        $h1Text = $h1Text . $separator . $p->get_text("/h1");
    }
    if($token->[0] eq 'S' and $token->[1] eq 'h2') {
        $h2Text = $h2Text . $separator . $p->get_text("/h2");
    }
    if($token->[0] eq 'S' and $token->[1] eq 'h3') {
        $h3Text = $h3Text . $separator . $p->get_text("/h3");
    }
    if($token->[0] eq 'S' and $token->[1] eq 'h4') {
        $h4Text = $h4Text . $separator . $p->get_text("/h4");
    }
    if($token->[0] eq 'S' and $token->[1] eq 'h5') {
        $h5Text = $h5Text . $separator . $p->get_text("/h5");
    }
    if($token->[0] eq 'S' and $token->[1] eq 'h6') {
        $h6Text = $h6Text . $separator . $p->get_text("/h6");
    }
}
$h1Text = cleanText($h1Text);
$h2Text = cleanText($h2Text);
$h3Text = cleanText($h3Text);
$h4Text = cleanText($h4Text);
$h5Text = cleanText($h5Text);
$h6Text = cleanText($h6Text);

my @h1Arr = split($separator, $h1Text);
my @h2Arr = split($separator, $h2Text);
my @h3Arr = split($separator, $h3Text);
my @h4Arr = split($separator, $h4Text);
my @h5Arr = split($separator, $h5Text);
my @h6Arr = split($separator, $h6Text);

my $h1Cnt = ($#h1Arr == 1) ? 0 : $#h1Arr;
my $h2Cnt = ($#h2Arr == 1) ? 0 : $#h2Arr;
my $h3Cnt = ($#h3Arr == 1) ? 0 : $#h3Arr;
my $h4Cnt = ($#h4Arr == 1) ? 0 : $#h4Arr;
my $h5Cnt = ($#h5Arr == 1) ? 0 : $#h5Arr;
my $h6Cnt = ($#h6Arr == 1) ? 0 : $#h6Arr;

my $h1Flag = "N";
my $h2Flag = "N";
my $h3Flag = "N";
my $h4Flag = "N";
my $h5Flag = "N";
my $h6Flag = "N";

$destArr1->[$i] =
    "$h1Cnt."|$h2Cnt."|$h3Cnt."|$h4Cnt."|$h5Cnt."|$h6Cnt;
if($#keywordFragments > 0) {
    # Obsługa wielu słów kluczowych
    for(my $j=0; $j<=$#keywordFragments; $j++) {

        if( $keywordFragments[$j] =~ /$h1Text/i ) {
            $h1Flag = "T";
        }
        if( $keywordFragments[$j] =~ /$h2Text/i ) {
            $h2Flag = "T";
        }
        if( $keywordFragments[$j] =~ /$h3Text/i ) {
            $h3Flag = "T";
        }
    }
}

```



```

        }
        if( $keywordFragments[$j] =~ /$h4Text/i ) {
            $h4Flag = "T";
        }
        if( $keywordFragments[$j] =~ /$h5Text/i ) {
            $h5Flag = "T";
        }
        if( $keywordFragments[$j] =~ /$h6Text/i ) {
            $h6Flag = "T";
        }
    }
} else {
    # Obsługa słowa kluczowego
    if($keyword =~ /$h1Text/i) {
        $h1Flag = "T";
    }
    if($keyword =~ /$h2Text/i) {
        $h2Flag = "T";
    }
    if($keyword =~ /$h3Text/i) {
        $h3Flag = "T";
    }
    if($keyword =~ /$h4Text/i) {
        $h4Flag = "T";
    }
    if($keyword =~ /$h5Text/i) {
        $h5Flag = "T";
    }
    if($keyword =~ /$h6Text/i) {
        $h6Flag = "Y";
    }
}
$destArr2->[$i] =
    ↪ "$h1Flag."|$h2Flag."|$h3Flag."|$h4Flag."|$h5Flag."|$h6Flag;
} else {
    # Brak pliku => wstaw wartości domyślne;
    $destArr1->[$i] = "O|O|O|O|O|O|";
    $destArr2->[$i] = "N|N|N|N|N|N|";
}
#print "\n".$destArr1->[$i]."\n".$destArr2->[$i];
}
}

```

Podprogram:

checkExactDescriptionMeta

Opis:

*# Ten podprogram sprawdza, czy występują dokładne dopasowania w opisie słów
kluczowych.*

Zmienne wejściowe:

\$numberOfElements => liczba plików do przetworzenia

\$destArr => macierz (odwołanie) do macierzy wyników

\$keyword => słowo kluczowe do analizy

\$prefix => prefiks pliku

Zmienne wyjściowe:

Brak

Zwraca:

Nic, cała praca wykonywana na przekazywanych macierzach

```

sub checkExactDescriptionMeta {
    my ($numberOfElements, $destArr, $keyword, $prefix) = @_;
    for(my $i=0; $i<$numberOfElements; $i++){
        $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            while (my $token=$p->get_tag("meta")) {

```

```

        if ($token->[1]{name} =~ /description/i) {
            my $metaDescription = $token->[1]{content};
            $metaDescription =~ s/"//;
            $metaDescription =~ s/'//;

            if($metaDescription =~ /$keyword/i) {
                $destArr->[$i] = "T";
            } else {
                $destArr->[$i] = "N";
            }
        }
    }
}

if ( !(exists $destArr->[$i]) ) {
    $destArr->[$i] = "N";
}
}

}

# Podprogram:
# checkExactDescriptionMeta
# Opis:
# Ten podprogram sprawdza, czy występują dokładne dopasowania w opisie słów
# kluczowych.
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy wyników
# $keyword => słowo kluczowe do analizy
# $prefix => prefiks pliku
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach
sub checkPartialDescriptionMeta {
    my ($numberOfElements, $destArr, $keyword, $prefix) = @_;
    my @keywordFragments = split(/ /, $keyword);

    for(my $i=0; $i<$numberOfElements; $i++){
        $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            while (my $token=$p->get_tag("meta")) {
                if ($token->[1]{name} =~ /description/i) {
                    my $metaDescription = $token->[1]{content};

                    if($#keywordFragments > 0) {
                        for (my $j=0; $j<=#keywordFragments; $j++){
                            if($metaDescription =~ /$keywordFragments[$j]/i) {
                                $destArr->[$i] = "T";
                                last;
                            } else {
                                $destArr->[$i] = "N";
                            }
                        }
                    } else {
                        if($metaDescription =~ /$keyword/i) {
                            $destArr->[$i] = "T";
                            last;
                        } else {
                            $destArr->[$i] = "N";
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    if ( !(exists $destArr->[$i])) {
        $destArr->[$i] = "N";
    }
}

# Podprogram:
# keywordDensity
# Opis:
# Ten podprogram oblicza gęstość występowania danego słowa kluczowego.
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy wyników
# $keyword => słowo kluczowe do analizy
# $prefix => prefiks pliku
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach

sub keywordDensity {
    my ($numberOfElements, $keyword, $destArr, $prefix) = @_;
    my $total = 0;
    # Pętla przez wszystkie pliki

    for(my $i=0; $i<$numberOfElements; $i++) {
        my $pageCopy = "";
        my $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            while (my $token = $p->get_tag("body")) {
                $pageCopy = $p->get_trimmed_text("/body");
            }
        } else {
            print "\nW czasie obliczania gęstości występowania słowa kluczowego nie
                ↳odnaleziono pliku.";
        }
        # Porównanie treści i macierzy (funkcja sep)
        $pageCopy =~ s/"//g;
        $pageCopy =~ s/'//g;

        $total = $total + calculateKD($i, $pageCopy, $destArr, $keyword);
    }
    return ($total/$numberOfElements);
}

# Podprogram:
# calculateKD
# Opis:
# Pomocniczy podprogram do obliczania gęstości występowania słów kluczowych
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $destArr => macierz (odwołanie) do macierzy wyników
# $keyword => słowo kluczowe do analizy
# $prefix => prefiks pliku
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach
sub calculateKD {
    my ($index, $pageCopy, $destArr, $keyword) = @_;

    my @keywordFragments = split (/ /,$keyword);

```

```

if ($#keywordFragments>0) {
    for (my $i=0; $i<= $#keywordFragments; $i++){
        my @tempArr = split(/$keywordFragments[$i]/, $pageCopy);
        my @tempArr2 = split(/ /, $pageCopy);
        if( ($#tempArr == 1) || ($#tempArr2 == 1)) {
            $destArr->[$index] = 0;
        }else {
            $destArr->[$index] = $destArr->[$index] + ($#tempArr/$#tempArr2)*100;
        }
    }
    return $destArr->[$index];
} else {
    my @tempArr = split(/$keyword/, $pageCopy);
    my @tempArr2 = split(/ /, $pageCopy);
    $destArr->[$index] = ($#tempArr/$#tempArr2)*100;
    return $destArr->[$index];
}
}

```

Podprogram:

keywordDomainExactMatch

Opis:

Ten podprogram analizuje słowa kluczowe w nazwach domen. Sprawdza,

czy słowo kluczowe jest częścią nazwy domeny.

Potencjalne udoskonalenie wiąże się z wyszukiwaniem odmian słowa kluczowego.

Zmienne wejściowe:

\$numberOfElements => liczba plików do przetworzenia

\$linksArr => macierz (odwołanie) do macierzy linków

\$destArr => macierz (odwołanie) do macierzy wyników

\$keyword => prefiks pliku dla trzech wyszukiwarek

Zmienne wyjściowe:

Brak

Zwraca:

Nic, cała praca wykonywana na przekazywanych macierzach

```

sub keywordDomainExactMatch {
    my ($keyword, $linksArr, $numberOfElements, $destArr) = @_;
    my $matchCnt=0;
    my @keywordFragments = split(/ /, $keyword);
    my $numberOfKeywordFragments = $#keywordFragments;
    my $total = 0;
    for (my $i=0; $i<=$numberOfElements; $i++) {
        $matchCnt=0;
        my $tmp = $linksArr->[$i];
        $tmp =~ s/^http:\\\\\/\\\/g;
        $tmp =~ s/^https:\\\\\/\\\/g;
        my @linkFragments = split(/\\\/, $tmp);
        my $link = $linkFragments[0];

        if($numberOfKeywordFragments>0) {
            for(my $j=0; $j<=$numberOfKeywordFragments; $j++) {
                if ($link =~ /$keywordFragments[$j]/i) {
                    $matchCnt++;
                }
            }
        } else {
            if($link =~ /$keyword/i) {
                $matchCnt++;
            }
        }
        if($matchCnt>0) {
            if($numberOfKeywordFragments>0) {
                if($matchCnt == ($numberOfKeywordFragments+1)) {
                    $destArr->[$i] = "T";
                }
            }
        }
    }
}

```

```

        } else {
            $destArr->[$i] = "N";
        }
    } else {
        # Pojedyncze słowo kluczowe
        $destArr->[$i] = "T";
    }
} else {
    $destArr->[$i] = "N";
}
if($destArr->[$i] eq "T") {
    $total++;
}
}
return ( ($total/$numberOfElements)* 100);
}

```

Podprogram:

keywordDomainPartialMatch

Opis:

Ten podprogram analizuje słowa kluczowe w nazwach domen. Szuka częściowych dopasowań między słowami kluczowymi a nazwą domeny.

Zmienne wejściowe:

\$numberOfElements => liczba plików do przetworzenia

\$linksArr => macierz (odwołanie) do macierzy linków

\$destArr => macierz (odwołanie) do macierzy wyników

\$keyword => słowo kluczowe do analizy

Zmienne wyjściowe:

Brak

Zwraca:

Nic, cała praca wykonywana na przekazywanych macierzach

```

sub keywordDomainPartialMatch {
    my ($keyword, $linksArr, $numberOfElements, $destArr) = @_;
    my $totalNumber = $numberOfElements;
    my $matchCnt=0;
    my @keywordFragments = split (/ /, $keyword);
    my $numOfKeywordFragments = $#keywordFragments;

```

```

    my $keywordHyphen = $keyword;
    my $keywordUnderscore = $keyword;
    my $keywordNoSpace = $keyword;
    $keywordHyphen =~ s/ /-/g;
    $keywordNoSpace =~ s/ //g;

```

Pętla przez wszystkie linki

```

if($numOfKeywordFragments >0) {
    for(my $i=0; $i<$numberOfElements; $i++) {
        my $tmp = $linksArr->[$i];
        $tmp =~ s/^http:\/\/\/gi;
        $tmp =~ s/^https:\/\/\/gi;
        my @linkFragments = split(/\/, $tmp);
        my $link = $linkFragments[0];
        for(my $j=0; $j<=$numOfKeywordFragments; $j++) {
            if($link =~ /$keywordFragments[$j]/i) {
                $destArr->[$i] = "T";
                $j = $numOfKeywordFragments;
                $matchCnt++;
            } else {
                $destArr->[$i] = "N";
            }
        }
    }
} else {

```

```

for(my $i=0; $i<$numberOfElements; $i++) {
    my $tmp = $linksArr->[$i];
    $tmp =~ s/^http:\\\\//g;
    $tmp =~ s/^https:\\\\//g;
    my @linkFragments = split(/\\/, $tmp);
    my $link = $linkFragments[0];

    if( ($link =~ /$keyword/) ||
        ($link =~ /$keywordHyphen/) ||
        ($link =~ /$keywordNoSpace/) ) {
        $destArr->[$i] = "T";
        $matchCnt++;
    } else {
        $destArr->[$i] = "N";
    }
}

}
return ( ($matchCnt/$totalNumber)* 100);
}

# Podprogram:
# compareTitlePageCopy
# Opis:
# Ten program porównuje tytuł strony z jej treścią
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $titlesArr => macierz (odwołanie) do macierzy tytułów
# $destArr => macierz (odwołanie) do macierzy wyników
# $prefix => prefiks pliku dla trzech wyszukiwarek
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach
sub compareTitlePageCopy {
    my ($numberOfElements, $titlesArr, $destArr, $prefix) = @_;
    # Pętla przez wszystkie pliki
    for(my $i=0; $i<=$numberOfElements; $i++) {
        # Dzielenie bieżącego tytułu na słowa tokeny
        my $title = $titlesArr->[$i];

        $title = cleanText($title);
        $title =~ s/\'//g;
        $title =~ s/\\/g;

        my @titleFragments = split(/ /, $title);
        # Pobieranie treści dla każdego pliku
        my $pageCopy = "";
        my $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            while (my $token = $p->get_tag("body")) {
                $pageCopy = $p->get_trimmed_text("/body");
                $pageCopy =~ s/\'//g;
                $pageCopy =~ s/\\/g;

                last;
            }
        }
        # Porównanie treści i macierzy (funkcja sep)
        compareTitlePageCopyHelper($i, $#titleFragments,
            ↪\@titleFragments, $pageCopy, $destArr);
    }
}

```

```

# Podprogram:
# compareTitlePageCopyHelper
# Opis:
# Ten podprogram używany jest przez podprogram compareTitlePageCopy
# do porównywania tytułu strony i jej treści
# Zmienne wejściowe:
# $index => reprezentuje indeks numeryczny macierzy
# $numberOfElements => liczba plików do przetworzenia
# $titleFragments => macierz (odwołanie) do macierzy fragmentów tytułów
# $pageCopy => treść strony
# $pageCopyTitleArr => macierz (odwołanie) do macierzy wyników
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach
sub compareTitlePageCopyHelper {
    my ($index, $numberOfElements, $titleFragments, $pageCopy, $pageCopyTitleArr) = @_;
    my $foundCnt = 0;
    my $totalTitleFragments = $numberOfElements;

    for(my $j=0; $j<=$numberOfElements; $j++){
        my $tmpfragment = $titleFragments->[$j];

        if( $pageCopy =~ /$tmpfragment/i ){
            $foundCnt++;
        }
    }
    if($foundCnt == 0){
        $pageCopyTitleArr->[$index] = 0;
    } else {
        $pageCopyTitleArr->[$index] = ( ($foundCnt/($totalTitleFragments+1)) * 100);
    }
}

```

```

# Podprogram:
# compareArrays
# Opis:
# Ten podprogram porównuje ze sobą elementy dwóch macierzy, aby sprawdzić
# czy się w sobie nawzajem znajdują.
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $realArr => macierz (odwołanie) do pierwszej macierzy źródłowej
# $foundArr => macierz (odwołanie) do drugiej macierzy źródłowej
# $destArr => macierz (odwołanie) do macierzy wyników
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazywanych macierzach
sub compareArrays {
    my ($numOfElements, $realArr, $foundArr, $destArr) = @_;
    my $found = 0;
    my $percentMatch = 0;

    for(my $i=0; $i<$numOfElements; $i++){
        $tmpVar = $foundArr->[$i];
        $tmpVar =~ s/\/(\/\/\/\/g;
        $tmpVar =~ s/\/\/(\/\/\/\/g;
        $tmpVar =~ s/\/-\/\/\/-g;
        $tmpVar =~ s/\/+\/\/\/+g;
        $tmpVar =~ s/\/$\/\/\/$g;
        $tmpVar =~ s/\/^\/\/\/^g;
        $tmpVar =~ s/\/[\/\/\/[g;
        $tmpVar =~ s/\/]/\/\/]/g;
        $tmpVar =~ s/\/}/\/\/}/g;
    }
}

```

```

$tmpVar =~ s/{/\\{/g;

if ($realArr->[$i] =~ /$tmpVar/i) {
    $destArr[$i] = "T";
    $found++;
}else {
    $destArr[$i] = "N";
}
}
return ( ($found/$numOfElements)*100);
}

# Podprogram:
# getRealTitles
# Opis:
# Ten podprogram pobiera rzeczywiste tytuły
# Zmienne wejściowe:
# $numberOfElements => liczba plików do przetworzenia
# $titlesArr => macierz (odwołanie) do macierzy, która będzie zawierać
# rzeczywiste tytuły
# $prefix => prefiks pliku, który ma być użyty
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Podprogram operuje na macierzy zdefiniowanej już wcześniej poza funkcją.
# Podprogram niczego nie zwraca.
sub getRealTitles {
    my ($numberOfElements, $titlesArr, $prefix) = @_;
    for(my $i=0; $i<$numberOfElements; $i++){
        $filename = './serptemp/' . $prefix . "$i.txt";
        if (-e "$filename"){
            my $p = HTML::TokeParser->new($filename);
            while (my $token = $p->get_token) {
                if ($token->[0] eq "S" and lc $token->[1] eq 'title') {
                    my $title = $p->get_text() || "not found";
                    $title =~ s/^\s+//;
                    $title =~ s/\s+$//;
                    $titlesArr->[$i]=$title;
                    last;
                }
            }
        }else {
            $titlesArr->[$i]="not found";
        }
    }
}

# Podprogram:
# getKeywordsTitleMatch
# Opis:
# Ten podprogram porównuje dane słowo kluczowe z wpisami macierzy
# budując jednocześnie trzecią macierz z wynikami tego porównania.
# Zmienne wejściowe:
# $keyword => słowo lub fraza kluczowa, na której przeprowadzona będzie analiza
# $sourceArr => macierz (odwołanie), która ma zostać użyta do porównania
# $numOfElements => rozmiar macierzy odwołania
# $destArr => macierz (odwołanie)zawierająca wyniki porównania
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Podprogram operuje na macierzy zdefiniowanej już wcześniej poza funkcją.
# Podprogram niczego nie zwraca.
sub getKeywordsTitleMatch {
    my ($keyword, $sourceArr, $numOfElements, $destArr) = @_;

```



```

$keyword = cleanText($keyword);
$keyword =~ s/'//g;
$keyword =~ s/"//g;
@keywordFragments = split(/ /, $keyword);
my $numberOfKeywordTokens = $#keywordFragments;

for(my $i=0; $i<= $numOfElements; $i++) {
my $tmp = $sourceArr->[$i];
$tmp = cleanText($tmp);
$tmp =~ s/'//;
$tmp =~ s/"//;
my $foundCnt = 0;
if ($numberOfKeywordTokens > 0) {
    for(my $j=0; $j<=$#keywordFragments; $j++){
        if ($tmp =~ /$keywordFragments[$j]/i) {
            $foundCnt++;

            last;
        }
    }
} else {
    if ($tmp =~ /$keyword/i) {
        $foundCnt++;
    }
}
if($foundCnt > 0) {
    $destArr->[$i] = "T";
} else {
    $destArr->[$i] = "N";
}
}
}

```

Podprogram:

initializeKeyVariables

Opis:

Głównym celem jest zdefiniowanie macierzy linków i tytułów, które będą

używane w różnych częściach skryptu.

Zmienne wejściowe:

\$keyword => słowo lub fraza kluczowa, na której przeprowadzona będzie analiza

\$googleLinksArr => macierz (odwołanie) z linkami Google

\$googleTitlesArr => macierz (odwołanie) z tytułami Google

\$yahooLinksArr => macierz (odwołanie) z linkami Yahoo!

\$yahooTitlesArr => macierz (odwołanie) z tytułami Yahoo!

\$bingLinksArr => macierz (odwołanie) z linkami Bing

\$bingTitlesArr => macierz (odwołanie) z tytułami Bing

Zmienne wyjściowe:

Brak

Zwraca:

Podprogram operuje na przekazywanych macierzach zdefiniowanych już wcześniej poza funkcją.

Podprogram niczego nie zwraca.

```

sub initializeKeyVariables {
    my ($keyword, $googleLinksArr, $googleTitlesArr,
        $yahooLinksArr, $yahooTitlesArr, $bingLinksArr,
        $bingTitlesArr) = @_;

    # Tworzenie agentów
    my $uaGoogle = new LWP::UserAgent;
    my $uaYahoo = new LWP::UserAgent;
    my $uaBing = new LWP::UserAgent;

```

Ustawienie maksymalnego czasu wykonania polecenia na 25 sekund

\$uaGoogle->timeout(25);

\$uaYahoo->timeout(25);

\$uaBing->timeout(25);

```

# Konfiguracja agenta
my $useragent = "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)";
$uaGoogle->agent("$useragent");
$uaYahoo->agent("$useragent");
$uaBing->agent("$useragent");

# Konfigurowanie i pobieranie 100 wyników wyszukiwania z każdej wyszukiwarki
my $gurl=
↳ "http://www.google.com/search?num=$numres&hl=en&safe=off&q=$keyword&sa=N";
my $yurl=
↳ "http://search.yahoo.com/search?p=$keyword&ei=UTF-8&fr=sfp&n=$numres&b=1";
my $lurl=
↳ "http://search.bing.com/results.aspx?q=$keyword&first=1&count=$numres&";

my $reqGoogle = new HTTP::Request GET => "$gurl";
my $reqYahoo = new HTTP::Request GET => "$yurl";
my $reqBing = new HTTP::Request GET => "$lurl";

my $resGoogle = $uaGoogle->request($reqGoogle);
my $resYahoo = $uaYahoo->request($reqYahoo);
my $resBing = $uaBing->request($reqBing);

# Przypisanie stron z wynikami wyszukiwania do specjalnych zmiennych
my $ghtml = $resGoogle->content;
my $yhtml = $resYahoo->content;
my $lhtml = $resBing->content;

# Pobranie linków dla każdej strony z wynikami wyszukiwania
my $streamGoogle = HTML::TokeParser->new($ghtml);
my $streamYahoo = HTML::TokeParser->new($yhtml);
my $streamBing = HTML::TokeParser->new($lhtml);

# Przetwarzanie linków Google
my $cnt=0;
my $threeDots = '...';
while (my $token = $streamGoogle->get_token) {
    if ($token->[0] eq 'S' && $token->[1] eq 'a') {
        if( ($token->[2]{ 'href' } !~ /cache/i) &&
            !($token->[2]{ 'href' } !~ /^http/i) &&
            ($token->[2]{ 'href' } !~ /^https/i) &&
            ($token->[2]{ 'href' } !~ /google/i) &&
            ($token->[2]{ 'href' } !~ /aclk/i) &&
            ($token->[2]{ 'href' } !~ /youtube/i) &&
            ($token->[2]{ 'href' } !~ /wikipedia/i) ) {
            $googleLinksArr->[$cnt] = $token->[2]{ 'href' };
            $googleTitlesArr->[$cnt] = $streamGoogle->get_trimmed_text("/a");
            $googleTitlesArr->[$cnt] =~ s/$threeDots$//;
            $cnt++;
        }
    }
}

# Przetwarzanie linków Yahoo!
my $cnt2=0;
while (my $token = $streamYahoo->get_token) {
    if ($token->[0] eq 'S' && $token->[1] eq 'a') {
        @tmpurl= split (/\/\*\/, $token->[2]{ 'href' });
        $tmpurl[1] =~ s/%3f/?/g;
        $tmpurl[1] =~ s/%26/&/g;

        if( ($tmpurl[1] !~ /cache/i) &&
            ($tmpurl[1] !~ /^https/i) &&
            ($tmpurl[1] !~ /yahoo/i) &&
            ($tmpurl[1] !~ /wikipedia/i) &&
            ($tmpurl[1] !~ /overture/i) ){

```

```

$tmpurl[1] =~ s/%3a/:/g;
$tmpurl[1] =~ s/^s+//g;
if( $tmpurl[1] ne "" ) {
    $yahooLinksArr->[$cnt2] = $tmpurl[1];
    $yahooTitlesArr->[$cnt2] = $streamYahoo->get_trimmed_text("/a");
    $yahooTitlesArr->[$cnt2] =~ s/$threeDots$//;
    $cnt2++;
}
}
}
}
# Przetwarzanie linków Bing
my $cnt3=0;
while (my $token = $streamBing->get_token) {
    if ($token->[0] eq 'S' && $token->[1] eq 'a') {
        if( !($token->[2]{ 'href' } =~ /^http/i) &&
            ($token->[2]{ 'href' } =~ /^https/i) &&
            ($token->[2]{ 'href' } =~ /cache/i) &&
            ($token->[2]{ 'href' } =~ /wikipedia/i) &&
            ($token->[2]{ 'href' } =~ /msn/i) &&
            ($token->[2]{ 'href' } =~ /hotmail/i) &&
            ($token->[2]{ 'href' } =~ /microsoft/i) &&
            ($token->[2]{ 'href' } =~ /bing\.com/i) ) {
            $token->[2]{ 'href' } =~ s/^s+//g;
            if($token->[2]{ 'href' } ne "") {
                $bingLinksArr->[$cnt3] = $token->[2]{ 'href' };
                $bingTitlesArr->[$cnt3] = $streamBing->get_trimmed_text("/a");
                $bingTitlesArr->[$cnt3] =~ s/$threeDots$//;
                $cnt3++;
            }
        }
    }
}
}

# Podprogram:
# getSERPResults
# Opis:
# Ten podprogram pobiera wszystkie pliki htmls wszystkich zdefiniowanych adresów urls
# zdefiniowanych w macierzy, do której odwołuje się zmienna $urlArr
# Zmienne wejściowe:
# $numberOfElements => rozmiar macierzy odwołania
# $urlArr => macierz (odwołanie) z adresami URL do przetworzenia
# $name => prefiks pliku, który ma być użyty
# Zmienne wyjściowe:
# pliki tekstowe z kodem html z pobranych linków
# Zwraca:
# Podprogram operuje na macierzy zdefiniowanej już wcześniej poza funkcją.
# Podprogram niczego nie zwraca.
sub getSERPResults {
    my ($numberOfElements, $urlArr, $name) = @_;
    my $ua = new LWP::UserAgent;
    my $res;

    $ua->timeout(25);
    $ua->agent("My Crawler");

    for($i=0;$i<$numberOfElements;$i++){
        $filename = "./serptemp/". $name . $i . ".txt";
        $res = $ua->get("$urlArr->[$i]",':content_file' => "$filename");
    }
}

```

```

# Podprogram:
# cleanText
# Opis:
# Ten podprogram używany jest to wyczyszczenia fragmentów kodu HTML.
# Zmienne wejściowe:
# $text => treść pliku do wyczyszczenia
# Zmienne wyjściowe:
# Brak
# Zwraca:
# Nic, cała praca wykonywana na przekazanej macierzy
sub cleanText {
    my $text = shift;
    $text =~ s/\(/ /g;
    $text =~ s/\)/ /g;
    $text =~ s/\[/ /g;
    $text =~ s\/\]/ /g;
    $text =~ s\/\./ /g;
    $text =~ s\/\-/ /g;
    $text =~ s\/\=/ /g;
    $text =~ s\/\|/ /g;
    $text =~ s\/\!/ /g;
    $text =~ s\/\,/ /g;
    $text =~ s\/\?/ /g;
    $text =~ s\/\~/ /g;
    $text =~ s\/\:/ /g;
    $text =~ s\/\;/ /g;
    $text =~ s\/\&/ /g;
    $text =~ s\/\*/ /g;
    $text =~ s\/\$/ /g;
    $text =~ s\/\s+/ /g;
    return $text;
}

```

Rozdział 5.

linkchecker.pl

```

#!/usr/local/bin/perl
#####
# Plik: linkchecker.pl #
# Opis: Skrypt sprawdzający linki #
# Wywołanie: perl linkchecker.pl http://nazwadomeny.net > raport.csv #
use WWW::Mechanize;
use LWP::Simple;
my $baseurl = shift;
my @url=();
my @level=();
my @type=();
my @title=();
my @status=();
my @page=();
my %uniqueURL=();
my %checkedURL=();
my $masterCnt=0;
my $masterLevel=1;
$mech = WWW::Mechanize->new();
#### Przetwarzanie poziomu pierwszego
$mech->get( $baseurl );

@links = $mech->links();
foreach $link (@links) {
    $tmpurl = $baseurl . '/' . $link->url();
}

```

```

if ( ($link->url() !~ /mailto/i) &&
    ($link->url() !~ /javascript/i ) ) {
if ($link->url() !~ /^http/) {
    #Zbieranie unikalnych adresów URL
    $uniqueURL{$tmpurl}=$link->text();
    $url[$masterCnt]=$tmpurl;
    $type[$masterCnt]= "względn";
} else {
    $tmpurl = $link->url();
    $uniqueURL{$link->url()}=$link->text();
    $url[$masterCnt]=$link->url();
    if( $link->url() =~ /$baseurl/ ){
        $type[$masterCnt]= "bezwzględny wewnętrzny";
    } else {
        $type[$masterCnt]= "zewnętrzny";
    }
}
$level[$masterCnt]=$masterLevel;
$title[$masterCnt]=$link->text();
$page[$masterCnt]=$baseurl;
$masterCnt++;
}
}
$masterLevel++;
$linksOnFirstLevel=$masterCnt;

#### Przetwarzanie poziomu drugiego
%levTwoURLs = ();
$masterCnt = processSubLevel(2, $masterCnt, \@url, \@level, \@type,
    \@title, \@status, \@page, \%uniqueURL,
    $baseurl, $masterLevel, \%levTwoURLs);

$masterLevel++;
$linksOnSecondLevel = keys(%levTwoURLs);

#### Przetwarzanie poziomu trzeciego
%levThreeURLs = ();
$masterCnt = processSubLevel(3, $masterCnt, \@url, \@level,
    \@type, \@title, \@status, \@page,
    \%levTwoURLs, $baseurl, $masterLevel,
    \%levThreeURLs);

$masterLevel++;
$linksOnThirdLevel = keys(%levThreeURLs);

#### Przetwarzanie poziomu czwartego
%levFourURLs = ();
$masterCnt = processSubLevel(4, $masterCnt, \@url, \@level, \@type,
    \@title, \@status, \@page, \%levThreeURLs,
    $baseurl, $masterLevel, \%levFourURLs);
$linksOnFourthLevel = keys(%levFourURLs);
printReport(\@level, \@page, \@url, \@type, \@title, \@status, $masterCnt);
#### podprogramy
sub processSubLevel {
    my ($currentLevel, $mstCnt, $urlArr, $leArr, $tyArr, $tiArr,
        $stArr, $paArr, $urls, $burl, $mlevel,
        $uniqueHashRef) = @_;
    my %urlHash = ();
    foreach $item (@$urlArr){
        $urlHash{$item} = 1;
    }
    foreach $lURL (keys %$urls) {
        if( ($lURL !~ /\.gif$/) && ($lURL !~ /\.jpg$/) &&
            ($lURL !~ /\.png$/) && ($lURL !~ /\.pdf$/) &&
            ($lURL !~ /\.doc$/) && ($lURL !~ /\.xls$/) &&
            ($lURL !~ /\.asf$/) && ($lURL !~ /\.mov$/) &&
            ($lURL !~ /\.avi$/) && ($lURL !~ /\.xvid$/) &&
            ($lURL !~ /\.flv$/) && ($lURL !~ /\.mpg$/) &&
            ($lURL !~ /\.3gp$/) && ($lURL !~ /\.mp4$/) &&
            ($lURL !~ /\.qt$/) && ($lURL !~ /\.rm$/) &&

```

```

($LURL !~ /.swf$/) && ($LURL !~ /.wmv$/) &&
($LURL !~ /.txt$/) && ($LURL !~ /.js$/) &&
($LURL !~ /.css$/) && ($LURL =~ /$burl/) &&
($LURL !~ /mailto/i)&&($LURL !~ /javascript/i) ) {
$mech->get( $LURL );
@sublinks = $mech->links();
$cnt2=0;
foreach $link (@sublinks) {
    my $tmpurl = "";
    # założenie, że względny link tworzy zmienną tymczasową
    if ( $link->url() !~ /^http/i ) {
        $tmpurl = $burl . '/' . $link->url();
    } else {
        $tmpurl = $link->url();
    }
    if(!(exists $urlHash{$tmpurl})) {
        if ( ($link->url() !~ /mailto/i) &&
            ($link->url() !~ /javascript/i) ) {
            # Sprawdzanie unikalności
            if( !(exists $urls->{$tmpurl}) ) {
                $urls->{$tmpurl}=$link->text();
                $uniqueHashRef->{ $tmpurl } = $link->text();
            }
            # Sprawdzanie, czy link jest względny, czy bezwzględny
            if ( $link->url() !~ /^http/ ) {
                ## WZGLĘDNY
                $urlArr->[$mstCnt]= $tmpurl;
                $tyArr->[$mstCnt]= "względny wewnętrzny";
            } else {
                ## BEZWZGLĘDNY
                # dostrajanie zmiennej tymczasowej
                $urlArr->[$mstCnt]=$link->url();
                if( $link->url() =~ /$baseurl/ ){
                    $tyArr->[$mstCnt]= "bezwzględny wewnętrzny";
                } else {
                    $tyArr->[$mstCnt]= "zewewnętrzny";
                }
            }
            $leArr->[$mstCnt]=$mlevel;
            $tiArr->[$mstCnt]=$link->text();
            $paArr->[$mstCnt]=$tmpurl;
            $mstCnt++;
        }
    }
}
}
return ($mstCnt);
}
sub printReport {
    my ($levelArr, $pageArr, $urlArr, $typeArr, $titleArr,
        $statusArr, $mCnt) = @_;
    %tmpCleanupHash=();
    print "Poziom\tLokalizacja macierzysta\t
Unikatowy adres URL\tTyp odnośnika\tTytuł\tKody stanu";
    for($i=0;$i<$mCnt;$i++) {
        if ( !(exists $tmpCleanupHash{$url[$i]}) ){
            $tmpCleanupHash{$url[$i]} = 1;
            if ($levelArr->[$i] ne "") {
                print
"\n$levelArr->[$i]\t$pageArr->[$i]\t$urlArr->[$i]\t$typeArr->[$i]\t$titleArr->[$i]\t
t".getstore($urlArr->[$i], "temp");
            }
        }
    }
}
}

```

mymonitor.pl

```
#####
# Plik: mymonitor.pl #
# Opis: Ten skrypt pobiera argument #
# reprezentujący adres URL witryny #
# Format: perl mymonitor.pl http://www.xyz.com #
#####
use threads;
use Benchmark;
use Time::HiRes qw(gettimeofday tv_interval);
use LWP::Simple;
use LWP::UserAgent;
use File::Path;
# Pobieranie strony do monitorowania
my $pageToMonitor = shift;
my $ua = new LWP::UserAgent;
my $res;
# Czyszczenie plików tymczasowych
rmtree( './temp', {keep_root => 1} );
# Uruchomienie zegara
my $start_time = [ gettimeofday ];
$res = $ua->get("$pageToMonitor", ':content_file' => './temp/temp.dat');
# Zatrzymanie zegara
my $end_time = [ gettimeofday ];
my $elapsedtime = tv_interval($start_time,$end_time);
##### TWORZENIE PLIKÓW Z DANYMI #####
my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)
    = localtime time;
$year += 1900;
$mon++;
# Tworzenie pliku today.txt
open OUTPUT, ">>./report/today/today.txt";
print OUTPUT "$mday-$mon-$year $hour:$min:$sec;10;15;20;
$elapsedtime\n";
close OUTPUT;
# Tworzenie pliku month.txt
open OUTPUT, ">>./report/month/month.txt";
print OUTPUT "$mday-$mon-$year $hour:$min:$sec;10;15;20;
$elapsedtime\n";
close OUTPUT;
# Tworzenie pliku year.txt
open OUTPUT, ">>./report/year/year.txt";
print OUTPUT "$mday-$mon-$year $hour:$min:$sec;10;15;20;
$elapsedtime\n";
close OUTPUT;
# Tworzenie pliku historical.txt
open OUTPUT, ">>./report/historical/historical.txt";
print OUTPUT "$mday-$mon-$year $hour:$min:$sec;10;15;20;$elapsedtime\n";
close OUTPUT;
```

inlinksAnalysis.pl

```
#!/usr/local/bin/perl
#####
# Plik: inlinksAnalysis.pl #
# Opis: Ten skrypt analizuje linki zwrotne pozyskane #
# z pliku TSD usługi Yahoo! #
#####
```

```

use LWP::Simple;
use LWP::UserAgent;
use HTML::TokeParser;
my @URLs = ();
# Pobieranie wejściowego parametru — nazwy pliku
my $fileToProcess = $ARGV[0];
my $baseurl = $ARGV[1];
print "\nPrzetwarzanie: $fileToProcess";
my $cnt = 0;
# Otwórz plik
if (-e "$fileToProcess"){
    open FILE, "$fileToProcess" or die $!;
    while (<FILE>) {
        my $line = $_;
        my @fragments = split(/\t/, $line);
        my $url = $fragments[1];
        $URLs[$cnt] = $url;
        $cnt++;
    }
} else {
    print "\nPlik ($fileToProcess) nie istnieje";
}
my $ua = new LWP::UserAgent;
my $res;
$ua->agent("My Crawler");
my %linkPopHash = ();
my %anchorPopHash = ();
for(my $i=0; $i<=$cnt; $i++) {
    $res = $ua->get("$URLs[$i]", ':content_file' => "temp.txt");
    if (-e "temp.txt") {
        my $p = HTML::TokeParser->new("temp.txt");
        while (my $token = $p->get_tag("a")) {
            # Pobieranie linku i tekstu kotwiczki
            my $url = $token->[1]{href} || "-";
            my $anchorText = $p->get_trimmed_text("/a");
            $url =~ s/^\s+//g;
            $url =~ s/\s+$//g;
            my $text = $p->get_trimmed_text("/a");
            if ($url =~ /$baseurl/i) {
                #print "\n$Bazowy adres URL: $URLs[$i] LINK: $url";
                if(exists $linkPopHash{$url}){
                    $linkPopHash{$url} = $linkPopHash{$url} + 1;
                    $anchorPopHash{$url} = $anchorText;
                } else {
                    $linkPopHash{$url} = 1;
                    $anchorPopHash{$url} = $anchorText;
                }
            }
        }
    }
}
open (FP, '>report.txt');
foreach my $key ( sort { $linkPopHash{$b} <=> $linkPopHash{$a} }
keys %linkPopHash ) {
    print FP "$key, $linkPopHash{$key}, \"$anchorPopHash{$key}\"\\n\\n";
}
close (FP);

```


Rozdział 6.

searchPhraseReportGoogle.pl

```
#!/usr/bin/perl
#-----#
# PROGRAM: Raport dotyczący fraz wyszukiwania #
#-----#

$numArgs = $#ARGV + 1;

%googleDirCnt = ();

foreach $argnum (0 .. $#ARGV) {
    print "Analizowanie pliku $ARGV[$argnum] \n\n";
    $LOGFILE = "$ARGV[$argnum]";
    open(LOGFILE) or die("Nie można otworzyć pliku: $ARGV[$argnum].");
    foreach $line (<LOGFILE>) {
        # Analiza Google
        if(($line =~ /q=/) && ($line =~ /google/)) {
            @tmp1 = split ('GET ', $line);
            @tmp2 = split (' ', $tmp1[1]);
            @tmp3 = split ('q=', $tmp1[1]);
            @tmp4 = split ('\\', $tmp3[1]);
            # Czyszczenie
            $tmp4[0] =~ s/\\+/ /;
            $tmp4[0] =~ s/\\%20/ /g;
            $tmp4[0] =~ s/\\%3C/\\</gi;
            $tmp4[0] =~ s/\\%3E/\\>/gi;
            $tmp4[0] =~ s/\\%23/\\#/g;
            $tmp4[0] =~ s/\\%22/\\"/g;
            $tmp4[0] =~ s/\\%25/\\%/g;
            $tmp4[0] =~ s/\\%3A/\\:/gi;
            $tmp4[0] =~ s/\\%2F/\\/gi;
            $tmp4[0] =~ s/\\%2B/\\+/gi;
            @tmp5 = split ('\\', $tmp4[0]);
            $tmpKey = "<tr><td>". $tmp2[0]. " </td><td>". $tmp5[0]. " </td>";
            $googleDirCnt{$tmpKey} = $googleDirCnt{$tmpKey} + 1;
        }
    }
    close(LOGFILE);
}

open (FP, '>keywordssummary.html');
print FP "<html><head><title>Podsumowanie słów kluczowych</title><head>";
print FP "<body><strong>Podsumowanie dla wyszukiwarki Google</strong>";
print FP "<table width=400><tr><td><b>Adres URL/Źródło</b></td><td><b>Słowa kluczowe</b></td>";
print FP "<td><b>Suma</b></td><tr>";
foreach $key (sort hashValueDescendingNum (keys(%googleDirCnt))) {
    print FP $key. "<td>". $googleDirCnt{$key}. " </td></tr>";
}
print FP "</table></body></html>";
close (FP);

sub hashValueDescendingNum {
    $googleDirCnt{$b} <=> $googleDirCnt{$a};
}
```

Rozdział 13.

getRankings.pl

```
#!/usr/local/bin/perl

#####
# Plik: getRankings.pl                                #
# Opis: Skrypt ten odpytuje wyszukiwarki,             #
# aby wygenerować raport dotyczący rankingów         #
#####

### Podstawowa konfiguracja
$numOfArgs = $#ARGV + 1;
$originalkeywordphrase = "";
$targeturl="";

if ( ($numOfArgs == 0) || ($numOfArgs == 1) || ($numOfArgs < 0) ) {
    print ("\n\nWywołanie: perl getRanking.pl [DocelowyAdresURL] [SłowoKluczowe]\n");
    print ("\nOR\n");
    print ("\nUsage: perl getRanking.pl [DocelowyAdresURL] [SłowoKluczowe1]
    ↪[SłowoKluczowe2] ... [SłowoKluczoweN]\n\n");
    exit(0);
}

$targeturl=$ARGV[0];

if ( $numOfArgs == 2){
    $originalkeywordphrase = $ARGV[1];
}else {
    foreach $argnum (1 .. $#ARGV) {
        $originalkeywordphrase = $originalkeywordphrase . " " . $ARGV[$argnum];

        # Usunięcie spacji poprzedzających i końcowych
        $originalkeywordphrase =~ s/^\s+//;
        $originalkeywordphrase =~ s/\s+$//;
    }
}

$keywordphrase= $originalkeywordphrase;
$keywordphrase =~ s/([A-Za-z0-9])/sprintf("%%02X", ord($1))/seg;

# Zdefiniowanie źródłowych adresów URL
$listingNo=100;

$gurl=
"http://www.google.com/search?num=$listingNo&hl=en&safe=off&q=$keywordphrase&sa=N";
$burl= "http://www.bing.com/search?q=$keywordphrase&first=1&count=100&";

### Pobranie części stron z wynikami wyszukiwania
# Pobranie stron z wynikami wyszukiwania Google
$gserp = `wget "$gurl"
--user-agent="Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)"
--output-document="gserp.html" --cookies=off`;
# Pobranie stron z wynikami wyszukiwania Bing
$bserp = `wget "$burl"
--user-agent="Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)"
--output-document="bserp.html" --cookies=off`;

### Analizowanie
$googlePositionNumber = getPosition ($targeturl, "google");
$bingSearchPositionNumber = getPosition ($targeturl, "bing");
```

```

#Raport
#####
print "\nRaport podsumowujący ranking\n";
print "~~~~~\n";
print "Słowo kluczowe/fraza: $originalkeywordphrase\n";
print "Docelowy adres URL: $targeturl\n";
print "~~~~~\n";
print " Google.....: $googlePositionNumber\n";

if($bingSearchPositionNumber ne "nie znaleziono"){
    $cntAdjusted = $bingSearchPositionNumber + 1;
    print "Bing: $cntAdjusted\n";
}else{
    print "Bing: $bingSearchPositionNumber\n";
}
print "\nUwaga: Sprawdź z określoną z wyszukiwarką, aby zweryfikować poprawność
↳wyniku.\n";

##### PODPROGRAMY #####
sub getContent {
    $filename=shift;
    open INPUT, "<$filename";
    undef $/;
    $content = <INPUT>;
    close INPUT;
    #Przywracanie
    $/ = "\n";

    #Zamiana znaku nowego wiersza znakiem spacji
    $content =~ s/\n/ /g;
    #Zamiana cudzysłowów pustym ciągiem
    $content =~ s/\\"//g;

    #Czyszczenie bing
    $content =~ s/<strong>//g;
    $content =~ s/<\/strong>//g;

    $content =~ s/<cite>//g;
    $content =~ s/<\/cite>//g;

    return $content;
}

sub getPosition {
    $targeturl= shift;
    $se = shift;
    @tokens = ();
    $offset = 0;
    if($se eq "google") {
        $gcontent = getContent("gserp.html");
        @tokens = split(/h3 class=r/, $gcontent);
    } elsif($se eq "bing") {
        $bcontent = getContent("bserp.html");
        @tokens = split(/sa_cc/, $bcontent);
        $offset=2;
    }

    $mastercnt = "nie znaleziono";
    $cnt=0;
    $foundFlag = "nie";
    print "liczba tokenów:". $#tokens;
    foreach $token (@tokens) {
        #print "\ntoken: $token";
        if ($token =~ /$targeturl/gi) {

```

```

        if($foundFlag eq "nie") {
            $mastercnt = $cnt - $offset;
        } else {
            $mastercnt = "" . $mastercnt . "," . $cnt;
        }
        #print "\nDOPASOWANIE: $targeturl cnt: $cnt $mastercnt\n token";
        # Pobieranie wstecznej pozycji zwrotnej dopasowania
        $foundFlag = "tak";
    }
    $cnt = $cnt + 1;
}
return $mastercnt;
}

```

Rozdział 15.

sql.txt

```

CREATE TABLE `mojtest`.`lista` (
  `id` INT( 6 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  `wiadomosc` TEXT NOT NULL ,
  `stan` INT( 1 ) NOT NULL DEFAULT '0'
) ENGINE = MYISAM ;

```

config.php

```

<?
# Należy zmienić wszystkie wiersze poza ostatnim (link aktualizacji stanu Twittera)
# baza danych
$username="nazwa-uzytkownika-bazy-danych";
$password="haslo-do-bazy-danych";
$database="nazwa-bazy-danych";

# twitter
$tusrid = 'iduzytkownika-platfomy-Twitter';
$passwd = 'haslo-uzytkownika-platfomy-Twitter';
$tURL = 'http://twitter.com/statuses/update.xml';

?>

```

index.php

```

<html>

<head>

<title>Strona domowa</title>

<script>
function limitText(limitField, limitNum) {
    if (limitField.value.length > limitNum) {
        limitField.value = limitField.value.substring(0, limitNum);
    }
}
</script>

</head>

<body>

```

```

<h3> Co będziesz robić? <br>(lub co chcesz, by myśleli inni, że robisz)

</h3><br>
<form name=mainform method=post action=add.php onSubmit="return checkLength(this)">

<textarea name="message" rows="3" cols="80" onKeyDown="limitText(this,140);"
onKeyUp="limitText(this,140);">
</textarea> <br>
<input type=submit value='Dodaj przyszłą wiadomość'>

</form>
<br>

<?php
include("config.php");
mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Brak połączenia z bazą danych");
$query="SELECT * FROM lista where status=0 order by id desc";
$result=mysql_query($query);

$numOfRecords=mysql_numrows($result);

mysql_close();

echo "<b>Moje przyszłe wiadomości</center></b><br><hr>";

?>

<table border="1" cellspacing="2" cellpadding="2">
<tr>
<td><b>id</b></td>
<td><b>Wiadomość</b></td>
<td><b>Stan</b></td>
</tr>

<?
$i=0;
while ($i < $numOfRecords) {
    $id=mysql_result($result,$i,"id");
    $message=mysql_result($result,$i,"message");
    $status=mysql_result($result,$i,"status");
?>

<tr>
<td nowrap><? echo "$id"; ?></td>
<td width=350><? echo "$message"; ?>

    <?php

        $tmp = "";
        if ($status < 1) {
            $tmp = "Nie wysłano";
        }

        ?>

    &nbsp; <a href="delete.php?id=<?php echo $id ?>">Usuń</a>
</td><td nowrap><? echo "$tmp"; ?></td>
</tr>
<?
    $i=$i+1;
}
?>
</table>
</body>
</html>

```

add.php

```
<?
include("config.php");

mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Brak połączenia z bazą danych");

$message = $_POST['message'];

$query = "INSERT INTO lista (wiadomosc) VALUES ('$message')";
mysql_query($query);

mysql_close();

?>

<script>
alert('Tweet Added');
window.location.href = "index.php";
</script>
```

delete.php

```
<?
include("config.php");

mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Brak połączenia z bazą danych");

$myid = $_GET['id'];

$query="UPDATE lista SET stan=2 WHERE id=$myid";

mysql_query($query);

mysql_close();

?>

<script>
alert('Tweet Removed');
window.location.href = "index.php";
</script>
```

sendTweet.php

```
<?php

include("config.php");

mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Brak połączenia z bazą danych");

### Pobierz wiadomość
$result =
mysql_query("select id, wiadomosc from lista where stan=0 order by id asc LIMIT 1");

$row = mysql_fetch_array($result);

### Wyślij wiadomość
```

```

$curl_handle = curl_init();
curl_setopt($curl_handle, CURLOPT_URL, "$tURL");
curl_setopt($curl_handle, CURLOPT_CONNECTTIMEOUT, 2);
curl_setopt($curl_handle, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_handle, CURLOPT_POST, 1);

$message = $row['wiadomosc'];

curl_setopt($curl_handle, CURLOPT_POSTFIELDS, "status=$message");
curl_setopt($curl_handle, CURLOPT_USERPWD, "$tusr:$tpasswd");

$response = curl_exec($curl_handle);

curl_close($curl_handle);

// Pobierz stan wiadomości
if (empty($response)) {
    echo 'wiadomość nie została wysłana';
} else {
    echo 'wiadomość została wysłana';
    ### Aktualizacja stanu bazy danych
    $mid = $row['id'];
    mysql_query("UPDATE lista SET stan = 1 WHERE id = $mid");
}

mysql_close();
?>

```

Crontab

```

# Wyślij wiadomość 5 razy dziennie, o 7, 9, 11, 13 i 15
* 8,10,12,14,16 * * * php sendTweet.php

```

Rozdział 18.

Poniższy wykaz kodu zawiera jedynie główne fragmenty skryptów. Pełna wersja kodu źródłowego dostępna jest pod adresem book.seowarrior.com.

index.html

```

<html>
<head>

<title>SEO Warrior: Keyboard Dashboard (Alfa)</title>
<link rel="stylesheet" type="text/css" href="pagestyle.css" />
<script src="functions.js" type="text/javascript"></script>
<script src="dockablewindow.js" type="text/javascript"></script>
</head>

<body>
<table width=100% cellpadding=0 cellspacing=0 border=0>
<tr>
<td valign=top align=left><h1 style='color=blue'>SEO Warrior:
Keyword Dashboard (Alpha) </h1>
</td>
<td valign=top align=right>


```

```

<a href="http://www.seowarrior.net/contact/" title="Report Bugs">
<font size=2>Report Bugs</font></a> |
<a href="http://www.seowarrior.net/contact/" title="Make a Suggestion">
<font size=2>Suggestion</font></a>
  <a href="http://www.seowarrior.net"></a>
</td>
</tr>
</table>

<div id="formdiv">
  <form name="mainform" onSubmit="return false;">
    Keyword: <input type="text" id="keyword" name="keyword" size="20">
    <input type="button" id="phaseGoogleBtn" name="phaseGoogleBtn" value="Google"
    ↳onclick="stepOne('google')">
    <input type="button" id="phaseBingBtn" name="phaseBingBtn" value="Bing"
    ↳onclick="stepOne('bing')">
    <input type="button" id="phaseYahooBtn" name="phaseYahooBtn" value="Yahoo!"
    ↳onclick="stepOne('yahoo')">

    <input type="radio" name="resultLimit" value="10" checked >10
    <input type="radio" name="resultLimit" value="20">20
    <input type="radio" name="resultLimit" value="50">50
    [Results]
  </form>

</div>
<iframe onLoad="resizeG()" name="responsedivgoogle" id="responsedivgoogle"
scrolling="no"></iframe>
<iframe onLoad="resizeY()" name="responsedivyahoo" id="responsedivyahoo"
scrolling="no"></iframe>
<iframe onLoad="resizeB()" name="responsedivbing" id="responsedivbing"
scrolling="no"></iframe>

<iframe name="detailsframe" id="detailsframe" class="dockclass"></iframe>

<script type="text/javascript">
var dock0=new dockit("detailsframe", 0);
</script>

</body>

</html>

```

bParser.php

```

<html>
<head>

<style>
body {
  font-weight : normal;
  font-size : 12px;
  font-family : helvetica;
  text-decoration : bold;
  background : #f3f3f3;
}

a:hover {
  font-weight : normal;
  font-size : 12px;
  font-family : helvetica;

```



```

        background : #989898;
        text-decoration : bold;
    }

a:visited, a:link, a:active {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    color : #000022;
    text-decoration : normal;
}
</style>

</head>

<body>

<b>Bing</b>
<br>Keyword: <?=$ GET["keyword"]?>
<br>Showing</b> <?=$ _GET["resultLimit"]?> results
<br><hr>
<?

function getBaseURL($url){
    list($part1, $part2) = split("://", $url);
    list($part3, $part4) = split("/", $part2);
# $baseurl = $part1 . "://" . $part3;
    $baseurl = $part3;
    return $baseurl;
}

function getBingSERP($mykeyword, $myindex){
    $reg_ex = "[[:space:]]";
    $replace_word = "+";
    $str = $mykeyword;
    $mykeyword = ereg_replace($reg_ex, $replace_word, $str);

    $url = "http://www.bing.com/search?q=" . $mykeyword . "&first=" . $myindex . "&";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_COOKIEFILE, "c:\cookie.txt");
    $client = $ _SERVER['HTTP_USER_AGENT'];
    curl_setopt($ch, CURLOPT_USERAGENT, "$client");
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    $output = curl_exec($ch);
    curl_close($ch);
    return $output;
}

function processSERP($serp, $masterCnt, $rowLimit) {
    $dom = new DOMDocument();
    @$dom->loadHTML($serp);
    $xpath = new DOMXPath($dom);
    $hrefs = $xpath->evaluate("/html/body//a");
    $sofar = "";
    for ($i = 0; $i < $hrefs->length; $i++) {
        $href = $hrefs->item($i);
        $url = $href->getAttribute('href');

        $baseurl = getBaseURL($url);

        $urlChunks = split (" ", $_GET["keyword"]);

```

```

foreach ($urlChunks as $chunk) {
    $highChunk = '<B>'.$chunk.'</B>';
    $baseurl = str_replace("$chunk", "$highChunk", $baseurl);
}
$anchortext = $href->nodeValue;

if ( (preg_match("/live.com/i", "$url")) ||
    (preg_match("/msn.c/i", "$url")) ||
    (preg_match("/microsoft.com/i", "$url")) ) {
} else {
    if (preg_match("/^http/i", "$url") || preg_match("/^ftp/i", "$url")) {
        if (strpos($sofar, $baseurl) !== false) {
        } else {
            if($masterCnt < $rowLimit){
                ?>
<a target=detailsframe href='kw.php?url=<?=$url?>&keyword=<?=$_GET['keyword']?>'
title='<?=$anchortext?>'><?=$baseurl?></a><br><?
                $masterCnt++;
            }
        }
    }
    $sofar = $sofar . $baseurl;
}
return $masterCnt;
}

$rowLimit = $_GET["resultLimit"];

$masterCnt = 0;

$next = 1;
$keyword = $_GET["keyword"];
$serpRes = getBingSERP($keyword, $next);
$masterCnt = processSERP($serpRes, $masterCnt, $rowLimit);
flush();

if($masterCnt<$rowLimit) {
    sleep(rand(1, 3));
    $next = $first+10;
    sleep(rand(2, 6));
    $masterCnt = processSERP($serpRes, $masterCnt, $rowLimit);
    flush();
}

if($masterCnt<$rowLimit) {
    $next = $next+10;
    sleep(rand(1, 3));
    $serpRes = getBingSERP($keyword, $next);
    $masterCnt = processSERP($serpRes, $masterCnt, $rowLimit);
    flush();
}

if($masterCnt<$rowLimit) {
    $next = $next+10;
    sleep(rand(1, 3));
    $serpRes = getBingSERP($keyword, $next);
    $masterCnt = processSERP($serpRes, $masterCnt, $rowLimit);
    flush();
}

if($masterCnt<$rowLimit) {
    $nextRes = $next+10;
    sleep(rand(1, 3));

```

```

        $serpRes = getBingSERP($keyword, $next);
        $masterCnt = processSERP($serpRes, $masterCnt, $rowLimit);
    }
    ?>

</body>
</html>

```

gParser.php

```

<html>

<head>

<style>

body {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    text-decoration : bold;
    background : #f3f3f3;
}

a:hover {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    background : #989898;
    text-decoration : bold;
}

a:visited, a:link, a:active {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    color : #000022;
    text-decoration : normal;
}
</style>

</head>
<body>

<b>Google</b>
<br>Keyword: <?=$_GET["keyword"]?>
<br>Showing</b> <?=$_GET["resultLimit"]?> results
<br><hr>
<?

function getBaseURL($url){
    list($part1, $part2) = split("://", $url);
    list($part3, $part4) = split("/", $part2);
    # $baseurl = $part1 . "://" . $part3;
    $baseurl = $part3;
    return $baseurl;
}

function getGoogleSERP($mykeyword){
    $reg_ex = "[[:space:]]";
    $replace_word = "+";
    $str = $mykeyword;
    $mykeyword = ereg_replace($reg_ex, $replace_word, $str);

```

```

$url = "http://www.google.com/search?q=".$mykeyword."&num=50&";
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
$client = $_SERVER['HTTP_USER_AGENT'];
curl_setopt($ch, CURLOPT_USERAGENT, $client);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
$output = curl_exec($ch);
curl_close($ch);
return $output;
}

$rowLimit = $_GET["resultLimit"];

$keyword = $_GET["keyword"];
$serp = getGoogleSERP($keyword);

$dom = new DOMDocument();
@$dom->loadHTML($serp);
$xpath = new DOMXPath($dom);
$hrefs = $xpath->evaluate("/html/body//a");
$sofar = "";
$intCnt = 0;
for ($i = 0; $i < $hrefs->length; $i++) {
    $href = $hrefs->item($i);
    $url = $href->getAttribute('href');

    $baseUrl = getBaseUrl($url);

    $anchortext = $href->nodeValue;

    $urlChunks = split(" ", $keyword);

    foreach ($urlChunks as $chunk) {
        $highChunk = '<B>'.$chunk.'</B>';
        $baseUrl = str_replace("$chunk", "$highChunk", $baseUrl);
    }

    if ( (preg_match("/google.com/i", "$url")) ||
        (preg_match("/youtube.com/i", "$url")) ||
        (preg_match("/^\\/i", "$url")) ||
        (preg_match("/cache:/i", "$url")) ) {
    } else {
        if (preg_match("/^http/i", "$url") || preg_match("/^ftp/i", "$url")) {

            if (preg_match("/^http/i", "$url") || preg_match("/^ftp/i", "$url")) {
                if (strpos($sofar, $baseUrl) !== false) {
                } else {
                    if ($intCnt < $rowLimit) {
                        ?>
<a target=detailsframe href='kw.php?url=<?=$url?>&keyword=<?=$_GET['keyword']?>'
title='<?=$anchortext?>'><?=$baseUrl?></a><br><?
                        $intCnt++;
                    }
                }
            }
        }
    }
    $sofar = $sofar . $baseUrl;
}
?>

</body>
</html>

```

yParser.php

```
<html>
<head>

<style>
body {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    text-decoration : bold;
    background : #f3f3f3;
}

a: hover {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    background : #989898;
    text-decoration : bold;
}

a:visited, a:link, a:active {
    font-weight : normal;
    font-size : 12px;
    font-family : helvetica;
    color : #000022;
    text-decoration : normal;
}
</style>

</head>

<body>

<b>Yahoo!</b>
<br>Keyword: <?=$_GET["keyword"]?>
<br>Showing</b> <?=$_GET["resultLimit"]?> results
<br><hr>
<?

function getBaseURL($url){
    list($part1, $part2) = split("://", $url);
    list($part3, $part4) = split("/", $part2);
    # $baseurl = $part1 . "://" . $part3;
    $baseurl = $part3;
    return $baseurl;
}

function getYahooSERP($mykeyword){
    $reg_ex = "[[:space:]]";
    $replace_word = "+";
    $str = $mykeyword;
    $mykeyword = ereg_replace($reg_ex, $replace_word, $str);

    $url = "http://search.yahoo.com/search;_ylt=?p=".$mykeyword."&n=100&";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_REFERER, "http://search.yahoo.com/");
    $client = $_SERVER['HTTP_USER_AGENT'];
    curl_setopt($ch, CURLOPT_USERAGENT, $client);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```

    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
    $output = curl_exec($ch);
    curl_close($ch);
    return $output;
}

$rowLimit = $_GET["resultLimit"];

$keyword = $_GET["keyword"];
$serp = getYahooSERP($keyword);

$dom = new DOMDocument();
@$dom->loadHTML($serp);
$xpath = new DOMXPath($dom);
$hrefs = $xpath->evaluate("/html/body//a");
$sofar = "";

$intCnt = 0;
for ($i = 0; $i < $hrefs->length; $i++) {
    $href = $hrefs->item($i);
    $url = $href->getAttribute('href');
    $tmpurl = "";
    list($tmp1, $tmpurl) = split('\*\\*', $url, 2);
    $tmpurl = urldecode($tmpurl);
    $baseurl = getBaseURL($tmpurl);

    $urlChunks = split(" ", $keyword);

    foreach ($urlChunks as $chunk) {
        $highChunk = '<B>'.$chunk.'</B>';
        $baseurl = str_replace("$chunk", "$highChunk", $baseurl);
    }

    $anchor = $href->getAttribute('title');
    $anchortext = $href->nodeValue;
    if ( preg_match("/\*\\*/i", "$url") ) {
        if ( preg_match("/yahoo.com/i", "$baseurl") || preg_match("/cache/i",
            ↵"$url") ) {
        } else {
            if (preg_match("/^http/i", "$url") || preg_match("/^ftp/i", "$url")) {
                if (strpos($sofar, $baseurl) !== false) {
                } else {
                    if($intCnt < $rowLimit) {
                        ?>
<a target=detailsframe href='kw.php?url=<?=$tmpurl?>&keyword=<?=$_GET['keyword']?>'
title='<?=$anchortext?>'><?=$baseurl?></a><br><?
                        $intCnt++;
                    }
                }
            }
        }
    }
    $sofar = $sofar . $baseurl;
}

?>
</body>
</html>

```