SEARCH ENGINE UNTUK MENCARI FILE DENGAN DETEKSI HEADER

Oleh: Agung Budi Prasetyo

ABSTRAK

Dalam konteks umum, mesin pencari (*search engine*) adalah sebuah alat bantu untuk mencari sebuah atau lebih berkas (file) di lokasi tertentu (di komputer lokal maupun di internet). Berkas yang dicari dapat berupa berkas dokumen, gambar, audio, video bahkan berkas web. Kunci keberhasilan sebuah mesin pencari adalah akurasi atau ketepatan pencarian yaitu 'kesesuaian' antara berkas hasil pencarian dengan kenyataan sebenarnya apakah berkas tersebut memang benar berkas yang dimaksud oleh si pencari.

Artikel yang ditulis wikipedia menyebutkan *"Some sites use tricks to manipulate the search engine to display them as the first result returned for some keywords. This can lead to some search results being polluted, with more relevant links being pushed down in the result list."* (wikipedia, 2009). Dari tulisan tersebut tersirat dengan jelas bahwa sedikit demi sedikit fungsi dari sebuah mesin pencari telah bergeser (oleh sebagian orang) menjadi sebuah "media promosi". Dengan munculnya fenomena ini keakurasian pencarian data/ informasi/ file menjadi rendah dan kepentingan si pencari informasi menjadi terabaikan. Kenyataan ini memaksa para pengelola mesin pencari untuk selalu memperbarui teknik pencarian mereka.

Dengan menggunakan teknik pencarian berdasarkan deteksi *header*, dalam tulisan ini telah berhasil direkayasa sebuah mesin pencari file yang dibangun menggunakan bahasa pemrograman Delphi, di mana mesin pencari ini mampu melakukan pencarian file, termasuk terhadap file-file yang tidak dilengkapi dengan *type file (extension)*, ataupun file-file yang *extension*-nya sengaja disamarkan atau diganti dengan *extension* yang bukan miliknya. Pengujian terhadap *software* ini telah memberikan hasil berupa kesuksesan-

nya dalam menemukan berkas-berkas yang tidak mampu ditemukan dengan mesin pencari lain.

Kata Kunci: Mesin Pencari, *Search Engine,* Deteksi *Header.*

1 PENDAHULUAN

Search Engine (mesin pencari) adalah program komputer yang dirancang untuk membantu seseorang menemukan file-file (berkas) yang disimpan dalam komputer, misalnya dalam sebuah *server* umum di web (www) atau dalam komputer sendiri. Mesin pencari memungkinkan kita untuk meminta *content* media dengan kriteria yang spesifik (biasanya yang berisi kata atau frasa yang kita tentukan) dan memperoleh daftar file yang memenuhi kriteria tersebut. Mesin pencari biasanya menggunakan indeks (yang sudah dibuat sebelumnya dan dimutakhirkan secara teratur) untuk mencari file setelah pengguna memasukkan kriteria pencarian.

Salah satu metode pencarian sebuah file adalah dengan cara mendeteksi *filetype* atau *extension*-nya. Misalnya pencarian file gambar dilakukan dengan mendeteksi *filetype* .GIF, .BMP, .JPG, dan sebagainya. Pencarian file video dilakukan dengan mendeteksi *filetype* .AVI, .MPG, .DAT, dan sebagainya. Demikian pula untuk jenis-jenis file yang lain seringkali dideteksi berdasarkan *filetype* atau *extension*nya.

Permasalahan yang muncul adalah tidaklah selalu sebuah file disimpan lengkap dengan *extension*nya, misalkan untuk file-file yang sengaja disamarkan, atau juga untuk file-file hasil *download* yang mana *downloader* acap kali lupa menuliskan *extension*nya, sehingga sewaktu dilakukan pencarian terhadap file-file tersebut, file-file tersebut selalu lolos dari proses pencarian.

Melalui tulisan ini disajikan hasil rekayasa pembuatan mesin pencari *(search engine)* untuk mencari file-file (berkas), termasuk file-file yang tidak disertai *filetype (extension).*

Ruang lingkup pembuatan mesin pencari ini adalah pembuatan dua buah perangkat lunak (*form*), yang pertama *form* yang berguna sebagai me-

sin pencari, dan *form* yang kedua adalah *form* yang berguna untuk mencari contoh-contoh file untuk diarsip jenis *header*nya. Penyimpanan arsip *header* yang dilakukan adalah bersifat teks.

Tujuan rekayasa ini agar pengguna (pencari file) dimungkinkan untuk dapat menemukan file-file sekalipun file-file tersebut tidak ber-*extension*, bahkan file yang *extension*nya sengaja diganti dengan *extension* yang bukan aslinya.

2 LANDASAN TEORI

2.1 Mesin Pencari

Mesin pencari adalah program komputer yang dirancang untuk membantu seseorang menemukan file-file yang disimpan dalam komputer, misalnya dalam sebuah *server* umum di web (www) atau dalam komputer sendiri. Mesin pencari memungkinkan kita untuk meminta *content* media dengan kriteria yang spesifik (biasanya yang berisi kata atau frasa yang kita tentukan) dan memperoleh daftar file yang memenuhi kriteria tersebut. Mesin pencari biasanya menggunakan indeks (yang sudah pernah dibuat sebelumnya dan dimutakhirkan secara teratur) untuk mencari file setelah pengguna memasukkan kriteria pencarian.

Dalam konteks Internet, mesin pencari biasanya merujuk kepada www dan bukan protokol ataupun area lainnya. Selain itu, mesin pencari mengumpulkan data yang tersedia di *newsgroup*, *database* besar, atau direktori terbuka seperti DMOZ.org. Karena pengumpulan datanya dilakukan secara otomatis, mesin pencari berbeda dengan direktori Web yang dikerjakan manusia.

Sebagian besar mesin pencari dijalankan oleh perusahaan swasta yang menggunakan algoritma kepemilikan dan *database* tertutup, yang paling populer adalah Google (MSN Search dan Yahoo! tertinggal sedikit di belakang). Telah ada beberapa upaya menciptakan mesin pencari dengan sumber terbuka (*open-source*), contohnya adalah Htdig, Nutch, Egothor dan OpenFTS.

Tantangan yang dihadapi mesin-mesin pencari

Tantangan yang dihadapi mesin-mesin pencari antara lain:

- Web tumbuh sangat pesat, bahkan lebih cepat dibanding dengan kecepatan munculnya teknologi mesin pencari.
- Banyak web memperbarui file mereka secara periodik, namun pada mesin pencari tidak secara periodik diperbarui *database* kunci pencarian mereka.
- Beberapa mesin pencari tidak memberikan jawaban perncarian situs/ file berdasarkan kata kunci yang diberikan, namun berdasarkan berapa banyak mereka membayar sang pembuat mesin pencari.
- Banyak situs/ file menggunakan trik-trik khusus untuk memanipulasi file mereka supaya dapat muncul saat sebuah mesin pencari melakukan pencarian menggunakan kata kunci tertentu.

2.2 Berkas

Berkas komputer atau berkas (file) adalah entitas dari data yang disimpan di dalam sistem berkas yang dapat diakses dan diatur oleh pengguna. Sebuah berkas memiliki nama yang unik dalam direktori di mana ia berada. Alamat direktori di mana suatu berkas ditempatkan diistilahkan dengan *path*.

Nama berkas

Sistem berkas akan memberikan sebuah nama terhadap sebuah berkas agar dapat dikelola dengan mudah. Meski oleh sistem berkas penamaan dilakukan dengan menggunakan angka-angka biner, sistem operasi dapat menerjemahkan angka-angka biner tersebut menjadi karakter yang mudah dipahami.

Atribut berkas

Sebuah berkas berisi aliran data (atau *data stream*) yang berisi sekumpulan data yang saling berkaitan serta atribut berkas (yang bersifat wajib atau opsional), yang kadang-kadang disebut *properties* yang berisi informasi yang berkaitan dengan berkas yang bersangkutan. Informasi mengenai kapan sebuah berkas dibuat adalah contoh dari atribut berkas.

Ukuran berkas

Ukuran sebuah berkas umumnya direpresentasikan dalam satuan byte (bits). Jika bilangan terlalu besar untuk direpresentasikan dalam satuan byte, maka dapat menggunakan satuan KiB (Kibibyte, yang berarti 1,024 byte), MiB (Mebibyte, yang berarti 1,048,576 byte), GiB (Gibibyte, yang berarti 1,073,741,824 byte), dan TiB (Tebibyte, yang berarti 1,099,511,627,776 byte), selain tentunya menggunakan satuan KB (kilobyte, yang berarti 1,000 byte), MB (Megabyte, yang berarti 1,000,000 byte), GB (Gigabyte, yang berarti 1,000,000,000 byte), dan TB (Terabyte, yang berarti 1,000,000,000,000 byte).

Dalam mekanisme penyimpanan berkas, komputer akan menyimpan berkas dalam dua jenis ukuran: ukuran fisik dan ukuran logis. Ukuran fisik berkas merujuk kepada ukuran aktual dari berkas, yakni berapa banyak byte yang terdapat di dalam berkas. Sementara ukuran logis merujuk kepada jumlah ruangan yang dialokasikan oleh sistem berkas untuk menempatkan berkas yang bersangkutan di dalam media penyimpanan. Ukuran berkas fisik umumnya lebih besar dibandingkan dengan ukuran berkas logisnya. Sebagai contoh, untuk mengalokasikan berkas berukuran logis 5125 byte, dalam sebuah media penyimpanan yang diformat dengan sistem berkas yang menggunakan ukuran unit alokasi 4096 byte, komputer akan mengalokasikan dua buah unit alokasi, yang berukuran 4096 dan 4096, sehingga menghabiskan 8192 byte. Meski ukuran logis berkas tersebut 5125 byte, komputer mengalokasikan 8192 byte, membuat 3067 byte tidak digunakan (disebut sebagai *wasted space* atau *slack space*).

Manajemen berkas

Berkas komputer secara fisik dapat diatur oleh sistem berkas yang digunakan oleh media penyimpanan di mana berkas disimpan. Secara logis, pengguna membutuhkan sebuah utilitas untuk melakukan manajemen berkas, yang sering disebut sebagai "File Manager", atau manajer berkas. Contoh dari file manager adalah Windows Explorer dalam sistem operasi Windows, Norton Commander, Konqueror (dalam KDE), Nautilus (dalam GNOME), Midnight Commander, dan DOS Shell (dalam sistem operasi DOS).

Berkas video

Berkas video adalah berkas komputer yang digunakan untuk menyimpan kumpulan berkas digital seperti video, audio, metadata, informasi, pembagian *chapter*, dan judul sekaligus, yang dapat dimainkan atau digunakan melalui perangkat lunak tertentu pada komputer.

Daftar ekstensi berkas video

- WAV
- AIFF
- IFF
- MOV (untuk program Quicktime)
- MPEG-1
- MPEG-2
- MPEG-4 atau MP4
- Ogg
- ASF (standard file video dari Microsoft selain WMV)
- RM (untuk program Real Video)
- MKV (Matroska)
- MKA (Matroska)
- 3gp (untuk telepon genggam)
- NUT
- MPEG
- MXF
- ratDVD
- SVI
- DivX
- DAT (file standard vcd)
- VOB (file standard dvd)
- AVI (standard file video di komputer)
- VCD
- QT (file video program Quicktime)
- m1v (file MPEG video)
- m2p
- m2v
- MPG
- WMV
- ASX
- SWF (file animasi macromedia)
- AMV (digunakan pada perangkat MP4 player)

Berkas ASCII

Berkas ASCII adalah sebuah berkas yang hanya mengandung teks-teks yang diformat dengan menggunakan pengkodean ASCII. Berkas ini hanya terdiri atas karakter, angka, tanda baca, tabulasi, dan karakter pemisah baris *(carriage return)*. Dalam berkas ASCII, tidak ada pemformatan yang ekstensif seperti dalam dokumen Microsoft Word atau Rich Text Format, melainkan pemformatan yang digunakan hanyalah pemformatan standar yakni ASCII.

Pembuatan berkas ASCII

Dalam sistem operasi Windows, berkas ASCII ini dapat dibuat dengan menggunakan program Notepad atau editor teks lainnya. Jenis ekstensi yang digunakannya adalah .txt *(default)*, .bat, .cmd, .ini, .inf, dan masih banyak lainnya. Sistem operasi lainnya juga mengimplementasikannya, tapi beberapa sistem operasi (utamanya adalah keluarga UNIX) tidak mengharuskan adanya ekstensi berkas seperti pada Windows. Normalnya, sebagian besar berkas dalam sistem operasi UNIX merupakan berkas ASCII, kecuali berkas yang dapat dieksekusi (program). Berkas ASCII umumnya digunakan sebagai berkas teks biasa, skrip, kode sumber sebuah program (*source code* sebuah bahasa pemrograman), berkas konfigurasi (terutama dalam keluarga sistem operasi UNIX), atau berkas teks lainnya. Bahkan, banyak juga orang yang membuat gambar dengan menggunakan berkas ASCII, yang sering disebut sebagai ASCII Art. Karena berkas ASCII mengandung teks yang tidak diformat, berkas tersebut dapat dipahami banyak sistem operasi, karena memang ASCII adalah sebuah standar internasional (Windows, Macintosh, dan UNIX mendukung sepenuhnya standar ini). Akibatnya, berkas tersebut dapat menjadi sangat berguna dalam rangka berbagi informasi antar *platform* atau aplikasi.

2.3 Kemampuan Bahasa Pemrograman Delphi dalam menangani file

Bahasa pemrograman Delphi menyediakan instruksi-instruksi dan fungsi-fungsi yang digunakan untuk hal-hal yang berkaitan dengan file.

FindFirst, FindNext, dan FindClose

Digunakan untuk men-*scanning* file-file yang ada dalam sebuah *directory/ folder* dengan memberikan *set attribute* dan direktori yang sesuai.

Unit : SysUtils

Category : file management routines

Delphi syntax : `function FindFirst(const Path: string; Attr: Integer; var F: TSearchRec): Integer;`

Description : FindFirst mencari file di dalam folder yang ditunjuk oleh Path dan mencocokkannya dengan attribute yang ditunjuk oleh parameter Attr. Hasil pencarian dapat dilihat dalam parameter F. FindFirst akan menghasilkan nilai returns 0 jika file berhasil ditemukan, jika tidak berhasil akan muncul error code.

Parameter *Path* berisi *directory* atau *folder* di mana file yang akan dicari berada, misalnya 'C:\test*.*' .

Parameter *Attr* adalah spesifikasi dari file yang hendak di-*scan* meliputi:

Constant	Description
faReadOnly	Read-only files
faHidden	Hidden files
faSysFile	System files
faVolumeID	Volume ID files
faDirectory	Directory files
faArchive	Archive files
faAnyFile	Any file

Penggunaan *Attributes* dapat dikombinasikan menggunakan *adding* (pada Delphi) ataupun *or-ing* (pada C++).

Misalnya, untuk mencari file yang *read-only* dan *hidden files* gunakan cara *faReadOnly* + *faHidden* dalam Delphi atau *faReadOnly* | *faHidden* dalam C++ pada parameter *Attr* nya.

FileOpen, FileSeek, dan *FileRead*

Digunakan untuk membuka file yang telah ditentukan nama filenya.

Unit : *SysUtils*

Category : *file management routines*

Delphi *syntax* : function FileOpen(const FileName: string; Mode: LongWord): Integer;

Description : Perintah *FileOpen* digunakan untuk membuka file yang telah dideklarasikan dalam *file handle*. Nilai *acces mode* ditentukan dari nilai *fmOpen* yang didefinisikan dengan perintah *fileOpen* sebelumnya. Jika nilai balik yang didapat dari fungsi ini bernilai 0 atau lebih besar maka penggunaan fungsi ini berhasil, namun jika bernilai -1 maka artinya telah terjadi *error*.

Biasanya operasi ini digunakan bersama pemakaian *AssignFile*, *Rewrite*, and *Reset*.

AssignFile, Reset, Rewrite, CloseFile

Unit : *System*

Category : *file management routines*

Delphi *syntax* : procedure AssignFile(var F; FileName: string);

Description : Dalam penggunaan *AssignFile*, diperlukan dua buah parameter yaitu parameter *FileName* yang digunakan untuk menentukan file apa yang hendak di'buka', dan parameter *F* yang digunakan untuk menampung isi file yang telah dibuka oleh perintah *AssignFile*.

Setelah file berhasil dibuka, ada 2 jenis mode operasi yang dapat dilakukan yaitu *Reset(F)* digunakan untuk membaca isi file, serta *Rewrite(F)* yang digunakan jika hendak memanipulasi ini dari file.

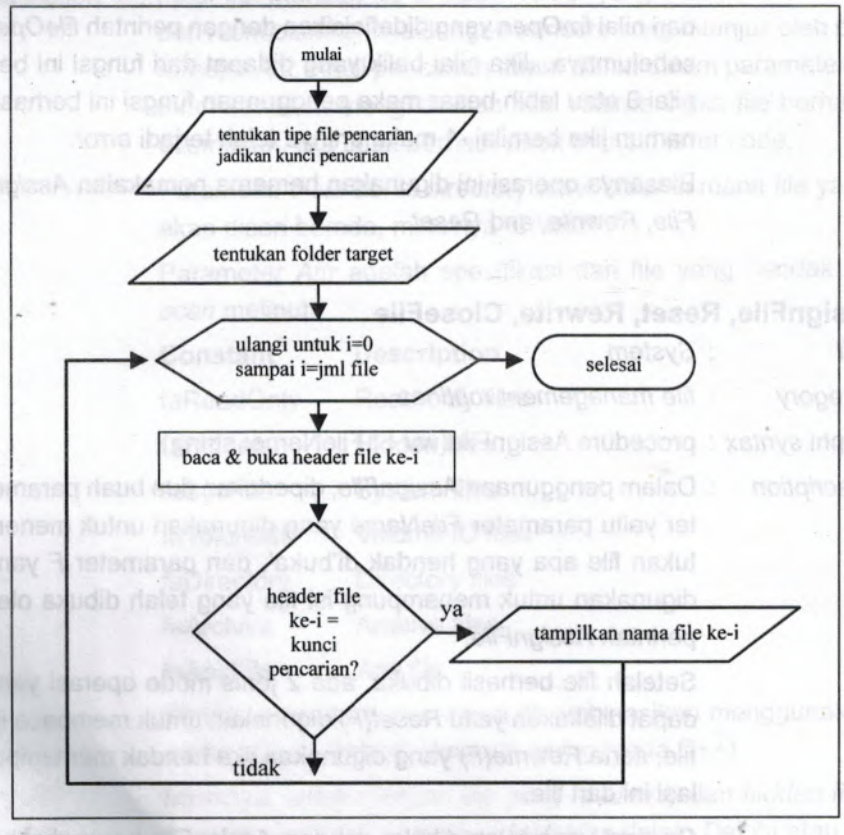
Operasi yang telah dibuka dengan *AssignFile* harus diakhiri dengan perintah *CloseFile(F)*.

3 PEMBAHASAN

3.1 Perancangan

Materi yang dikaji meliputi perancangan perangkat lunak berupa sebuah *form* utama yang berfungsi sebagai mesin pencari, dan sebuah *form* pendukung yang berfungsi sebagai pengaturan *setting-setting* yang berkaitan dengan dokumentasi dari *header* dari file yang pernah di'buka'.

Secara keseluruhan *flowchart* dari sistem yang dirancang dapat dilihat dari Gambar 3.1.



Gambar 3.1 *Flowchart Program*

Cuplikan Progam untuk mesin pencari ini dapat dilihat pada Gambar 3.2 dan Gambar 3.3.

```
unit Unit1_MesinPencari;

interface

uses
  ..........................;

type
  ..........................;
  ..........................;
  private
    {   ..........................; }
  public
    {   ..........................; }
  end;

var
  ..........................;

implementation

{$R *.dfm}

uses Unit2_Pengaturan;

//MENGKLIK TOMBOL CARI
procedure TForm1Gabungan.Button1Click(Sender: TObject);
var
  sr            : TSearchRec;
  FileAttrs     : Integer;

  nama_file_dokumentasi : string;
  folder_dokumentasi    : string;
  urutan                : integer;
```

Gambar 3.2 *Cuplikan Program Form Mesin Pencari*

```
    header_file_pencarian : String;
    nama_file_pencarian   : string;
    drive_pencarian : string;
    path_pencarian  : string;
    all_pencarian   : string;


    iFileHandle : Integer;
    iFileLength : Integer;
    iBytesRead  : Integer;
    Buffer      : PChar;
    i           : Integer;
begin
  //CHECKBOX UNTUK MENENTUKAN ATTRIBUTE-ATRIBUT FILE APA
SAJA YANG INGIN DICARI
  StringGrid1.RowCount := 1;
  if CheckBox1.Checked then  FileAttrs := faReadOnly
        else FileAttrs := 0;
  if CheckBox2.Checked then  FileAttrs := FileAttrs + fa-
Hidden;
  if CheckBox3.Checked then  FileAttrs := FileAttrs + fa-
SysFile;
  if CheckBox4.Checked then  FileAttrs := FileAttrs + fa-
VolumeID;
  if CheckBox5.Checked then  FileAttrs := FileAttrs + fa-
Directory;
  if CheckBox6.Checked then  FileAttrs := FileAttrs +
faArchive;
  if CheckBox7.Checked then  FileAttrs := FileAttrs +
faAnyFile;

  //MENENTUKAN LOKASI PENCARIAN
  drive_pencarian :=  ComboBox1.Items.Strings [ComboBox1.
ItemIndex]; //berisi DRIVE --> C:\
  path_pencarian  :=  Edit1.Text; //berisi PATH-->Docu-
ments and Settings\All Users\Documents\
  all_pencarian   :=  Edit2.Text; //berisi ALL   --> *.*
  Edit3.Text  := drive_pencarian + path_pencarian + all_
pencarian;
```

Gambar 3.2 *Cuplikan Program Form Mesin Pencari (lanjutan)*

```
  //MENGAMBIL DAFTAR HEADER FILE DARI ARSIP DOKUMENTASI
  nama_file_dokumentasi := ComboBox2.Items.Strings [Combo-
Box2.ItemIndex]; //berisi --> JPG
  folder_dokumentasi := Edit4.Text; //folder_dokumentasi
:= Form2Pengaturan.Edit3.Text;
  Label6.Caption := 'Daftar Header dari File ' + nama_
file_dokumentasi + ' :';
  ListBox1.Items.LoadFromFile(folder_dokumentasi + '_' +
nama_file_dokumentasi  + '.---');

with StringGrid1 do
begin
    RowCount := 0;
    //PERULANGAN DILAKUKAN SEBANYAK JUMLAH DAFTAR HEADER
YG ADA DARI TYPE FILE YG DICARI
    //LISTBOX1 : BERISI BEBERAPA CONTOH HEADER
    For urutan := 1 to ListBox1.Items.Count  do
    begin
          //mulai men-scan file dalam 1 folder
          if FindFirst(drive_pencarian + path_pencarian
+all_pencarian,FileAttrs,sr) = 0 then
          begin
              repeat
                    //scan file ke - i
                    nama_file_pencarian := drive_pencarian
+ path_pencarian + sr.Name;
                    if (sr.Attr and FileAttrs) = sr.Attr
then
                    begin
                         try
                         //buka header file ke- i
                         iFileHandle := FileOpen(nama_
file_pencarian, fmOpenRead);
                         iFileLength := FileSeek(iFileHand
le,0,2);
                         FileSeek(iFileHandle,0,0);
                         Buffer :=
```

Gambar 3.2 *Cuplikan Program Form Mesin Pencari (lanjutan)*

```
gth + 1));
                        iBytesRead :=
FileRead(iFileHandle, Buffer^, iFileLength);
                        //header dimasukkan ke variabel
'header file pencarian'
                        header_file_pencarian := Buf-
fer;
                        FileClose(iFileHandle);
                        finally
                            FreeMem(Buffer);
                        end;
                      //PROSES PENCOCOKAN...   LISTBOX
KE-i BERISI CONTOH HEADER
                      if header_file_pencarian = List-
Box1.Items.Strings[urutan-1] then
                      begin
                        RowCount := RowCount + 1;
                        Cells[1,RowCount-1] := sr.Name;
                        Cells[2,RowCount-1] :=
IntToStr(sr.Size);
                        //membaca atribut berdasarkan
'truth tabel' dan operasi AND
                        if (sr.Attr and 1) = 1 then
Cells[3,RowCount-1] := 'R';
                        if (sr.Attr and 2) = 2 then
Cells[4,RowCount-1] := 'H';
                        if (sr.Attr and 4) = 4 then
Cells[5,RowCount-1] := 'S';
                        if (sr.Attr and 32) = 32 then
Cells[6,RowCount-1] := 'A';
                        if (sr.Attr and 16) = 16 then
Cells[7,RowCount-1] := 'D';
                      end;
                    end; //end if ..attribute
              until FindNext(sr) <> 0;
              FindClose(sr);
          end; //end if (findfirst..)
      end; //for
end; //end with
end; //end procedure
```

Gambar 3.2 *Cuplikan Program Form Mesin Pencari (lanjutan)*

```
//MENGKLIK TOMBOL KELUAR
procedure TForm1Gabungan.Button2Click(Sender: TObject);
begin
   Application.Terminate;
end;

//MENGKLIK TOMBOL PENGATURAN
procedure TForm1Gabungan.Button3Click(Sender: TObject);
begin
   Form2Pengaturan.ShowModal;
end;

//MERUBAH TIPE PENCARIAN
procedure TForm1Gabungan.ComboBox2Change(Sender: TOb-
ject);
var
  nama_file_dokumentasi : string;
  folder_dokumentasi    : string;
  urutan                : integer;
begin
  //MENGAMBIL DAFTAR HEADER DARI ARSIP
  nama_file_dokumentasi := ComboBox2.Items.Strings [Com-
boBox2.ItemIndex]; //berisi --> JPG
  folder_dokumentasi := Edit4.Text; //folder_dokumentasi
:= Form2Pengaturan.Edit3.Text;
  Label6.Caption := 'Daftar Header dari File ' + nama_
file_dokumentasi + ' :';
  ListBox1.Items.LoadFromFile(folder_dokumentasi + '_' +
nama_file_dokumentasi  + '.---');
end;
end.
```

Gambar 3.2 *Cuplikan Program Form Mesin Pencari (lanjutan)*

```
unit Unit2_Pengaturan;
interface
uses
  ..........................;
type
  ..........................;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  ..........................;
implementation
{$R *.dfm}
uses Unit1_MesinPencari;

//mengosongkan semua Edit dan ListBox di awal program
procedure TForm2Pengaturan.FormActivate(Sender: TObject);
begin
    Edit1.Text := '';
    Edit2.Text := '';
    ListBox1.Clear();
end;

//MENGKLIK TOMBOL BROWSE
procedure TForm2Pengaturan.Button1Click(Sender: TObject);
var
  iFileHandle: Integer;
  iFileLength: Integer;
  iBytesRead: Integer;
  Buffer: PChar;
begin
  //MEMBUKA SEBUAH SAMPEL FILE UNTUK DILIHAT HEADERNYA VIA
'OPEN DIALOG'
  if OpenDialog1.Execute then
  begin
    try
```

Gambar 3.3 *Cuplikan Program Form Pengaturan*

```
      iFileHandle := FileOpen(OpenDialog1.FileName, fmO-
penRead);
      iFileLength := FileSeek(iFileHandle,0,2);
      FileSeek(iFileHandle,0,0);
      Buffer := PChar(AllocMem(iFileLength + 1));
      iBytesRead := FileRead(iFileHandle, Buffer^, iFile-
Length);
      FileClose(iFileHandle);

      edit1.Text := OpenDialog1.FileName;      //masukkan
ke --> Nama File yang dibuka
      edit2.Text := Buffer;                    //masukkan
ke --> Header Filenya
      Button5.Enabled := true;
    finally
      FreeMem(Buffer);
    end;
  end;
end;

//MENENTUKAN DIRECTORY SETTING
procedure TForm2Pengaturan.Edit3Change(Sender: TObject);
begin
   Form1Gabungan.Edit4.Text := Edit3.Text;
end;


//MENG KLIK TOMBOL "LOAD"
procedure TForm2Pengaturan.Button4Click(Sender: TObject);
var
  iFileHandle: Integer;
  iFileLength: Integer;
  iBytesRead: Integer;
  Buffer: PChar;
  i: Integer;
begin
  if OpenDialog1.Execute then
  begin
```

Gambar 3.3 *Cuplikan Program Form Pengaturan (lanjutan)*

```
      ListBox1.Items.LoadFromFile(OpenDialog1.FileName);
  end;
end;
//MENG KLIK TOMBOL "SAVE"
procedure TForm2Pengaturan.Button3Click(Sender: TObject);
var  a : integer;
begin
      if SaveDialog1.Execute then
      begin
        if FileExists(SaveDialog1.FileName) then
        begin
          with Application do
          begin
                NormalizeTopMosts;
                a := MessageBox('file sudah ada. Ingin
Overwrite?','Peringatan',4);
                if (a = 6) then // 6=yes,   7=no
                   ListBox1.Items.SaveToFile(SaveDialog1.
FileName);
                RestoreTopMosts;
          end;
        end
        else
          ListBox1.Items.SaveToFile(SaveDialog1.File-
Name);
      end;
end;

//mengklik tombol ">>"
procedure TForm2Pengaturan.Button5Click(Sender: TObject);
begin
  ListBox1.Items.Add(Edit2.Text);
  Edit2.Clear();
  Button5.Enabled := false;
end;

//menghapus isi ListBox1 dengan klik ganda
procedure TForm2Pengaturan.ListBox1DblClick(Sender: TOb-
ject);
```

Gambar 3.3 *Cuplikan Program Form Pengaturan (lanjutan)*

```
var
  i : integer;
begin
  i := ListBox1.ItemIndex;
  ListBox1.Items.Delete(i);
end;

//MENG KLIK TOMBOL "KELUAR"
procedure TForm2Pengaturan.Button2Click(Sender: TObject);
begin
   Form2Pengaturan.Close;
end;
end.
```

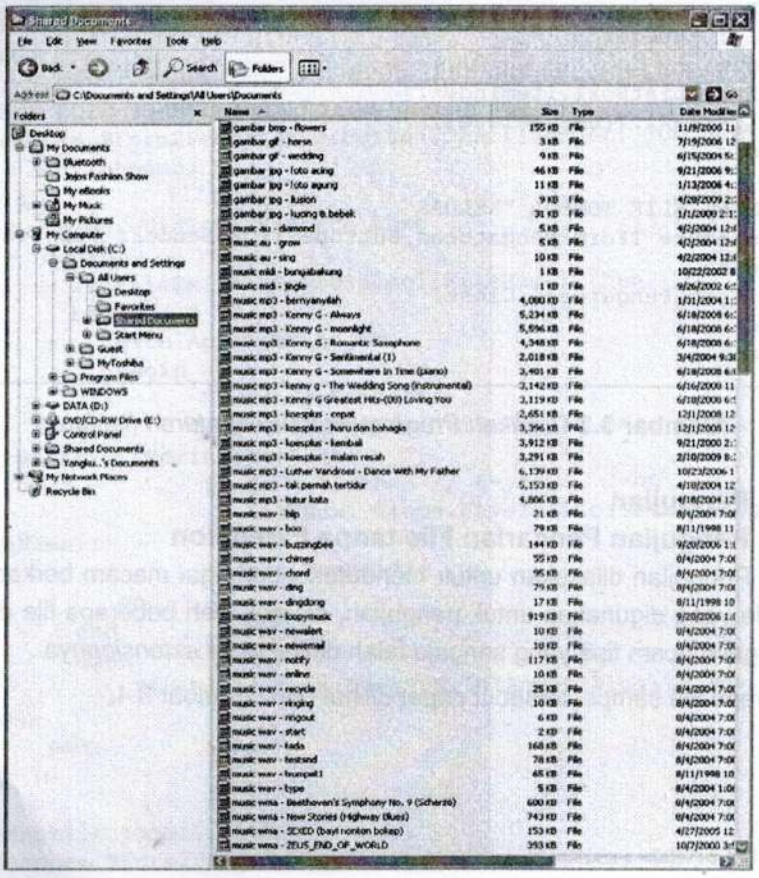
Gambar 3.3 *Cuplikan Program Form Pengaturan (lanjutan)*

3.2 Pengujian

3.2.1 Pengujian Pencarian File tanpa Extension

Pengujian dilakukan untuk mendeteksi berbagai macam berkas (file). File-file yang digunakan untuk pengujian, diwakili oleh beberapa file dengan berbagai macam tipe yang sengaja telah dihilangkan *extension*nya .

Beberapa file sampel tersebut dapat dilihat pada gambar 3.4.

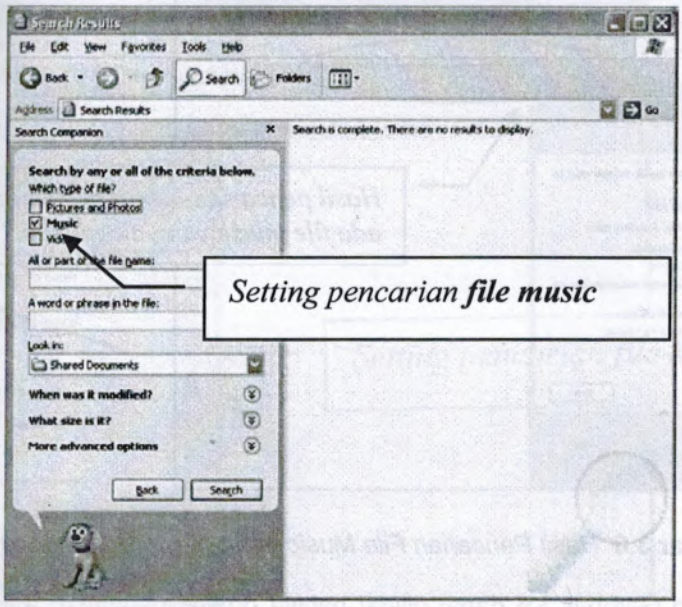


Gambar 3.4 *Beberapa Sample File tanpa Type atau Extension*

3.2.1.1 Pengujian dengan Mesin Pencari Konvensional

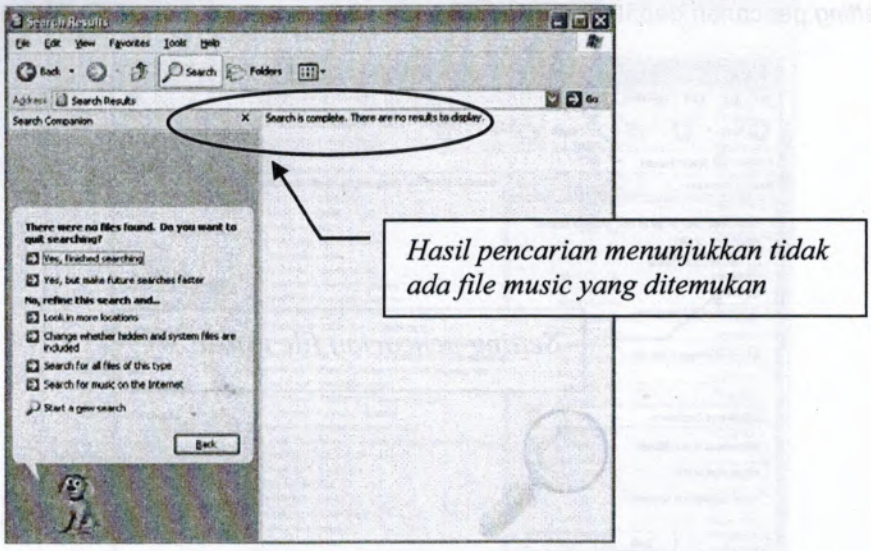
Pengujian berikut akan dilakukan terhadap file-file sampel yang telah disebutkan dalam Gambar 3.4 dengan menggunakan mesin pencari bawaan dari sistem operasi Windows. Dari file tersebut akan dicari file-file berbasis musik, seperti MP3, WAV, dan lain-lain.

Setting pencarian dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Setting Pencarian File Music pada Mesin Pencari Windows*

Setelah *setting* pencarian ditentukan dan proses pencarian dilakukan, akan didapatkan hasil pencarian seperti ditunjukkan pada Gambar 3.6.

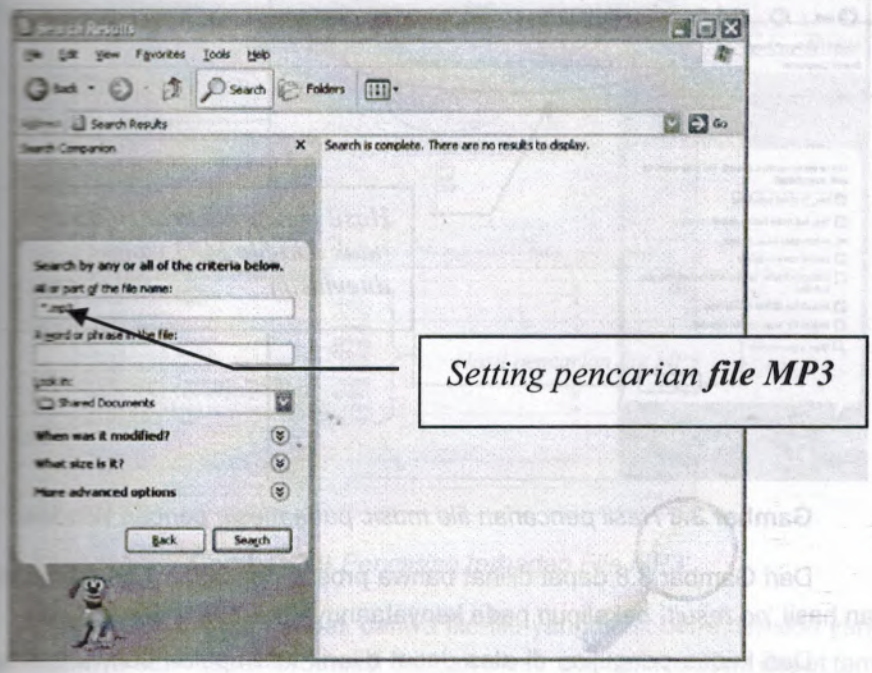


Gambar 3.6 *Hasil Pencarian File Music pada Mesin Pencari Windows*

Dari Gambar 3.6 dapat dilihat bahwa proses pencarian memberikan hasil *'no result'* sekalipun pada kenyataannya file musik tersebut ada.

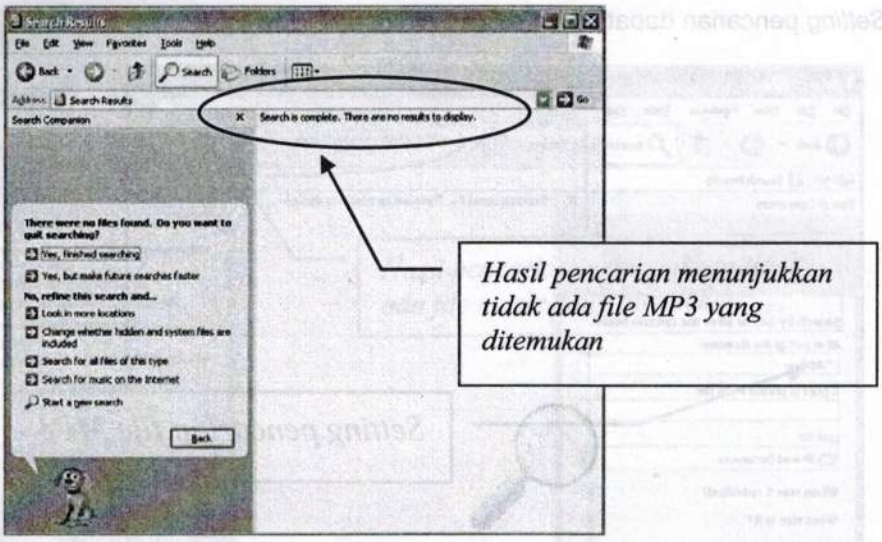
Pengujian berikutnya juga akan dilakukan terhadap file-file sampel yang telah disebutkan sebelumnya dalam Gambar 3.4 dengan menggunakan mesin pencari bawaan dari sistem operasi Windows. Namun bedanya adalah pada pencarian ini *extension* dari file yang dicari disebutkan secara spesifik, yaitu file ber-*extension* MP3.

Setting pencarian dapat dilihat pada Gambar 3.7.



Gambar 3.7 *Setting Pencarian File Berextension MP3 dengan Mesin Pencari Windows*

Setelah *setting* pencarian ditentukan dan proses pencarian dilakukan, akan didapatkan hasil pencarian seperti ditunjukkan pada Gambar 3.8.



Gambar 3.8 *Hasil pencarian file music pada mesin pencari Windows*

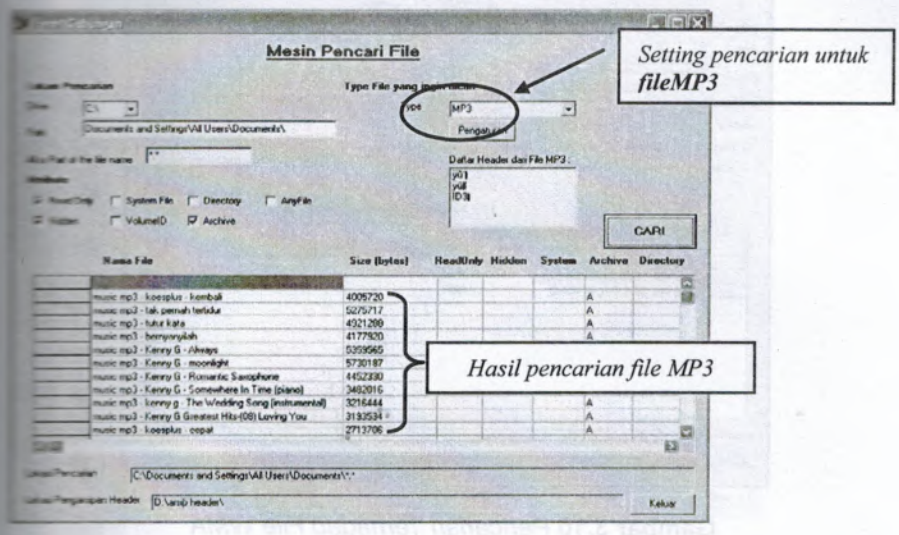
Dari Gambar 3.8 dapat dilihat bahwa proses pencarian juga memberikan hasil *'no result'* sekalipun pada kenyataannya file musik tersebut ada.

Dari kedua pengujian di atas dapat ditarik kesimpulan bahwa pencarian file menggunakan mesin pencari bawaan sistem operasi Windows menggunakan metode deteksi *extension* sehingga jika *extension* dari file tersebut disembunyikan maka mesin pencari tidak dapat menemukannya.

3.2.1.2 Pengujian dengan Mesin Pencari Hasil Rekayasa

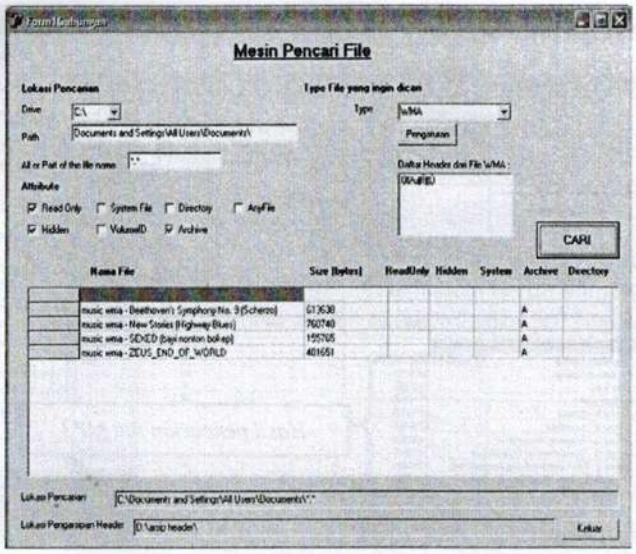
Berikutnya pengujian yang sama yang akan dilakukan terhadap file-file sampel yang telah disebutkan sebelumnya dalam Gambar 3.4 dengan menggunakan mesin pencari hasil rekayasa penulis. Dari file tersebut akan dicari file-file yang sama, yaitu file berbasis MP3.

Setting serta hasil pencarian dapat dilihat pada Gambar 3.9.

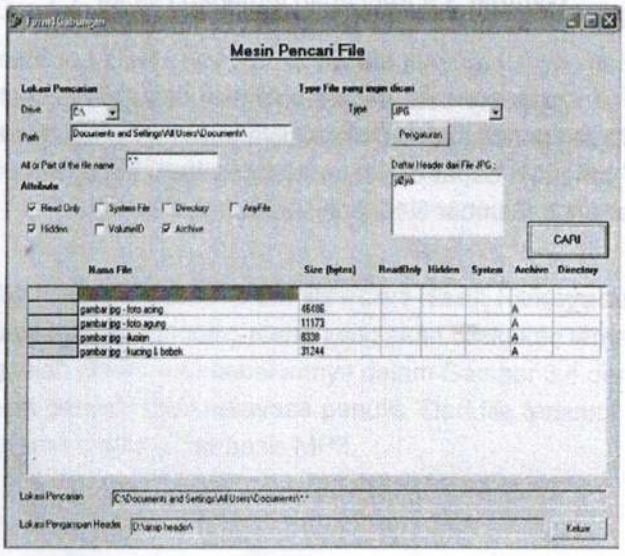


Gambar 3.9 *Pencarian terhadap File MP3*

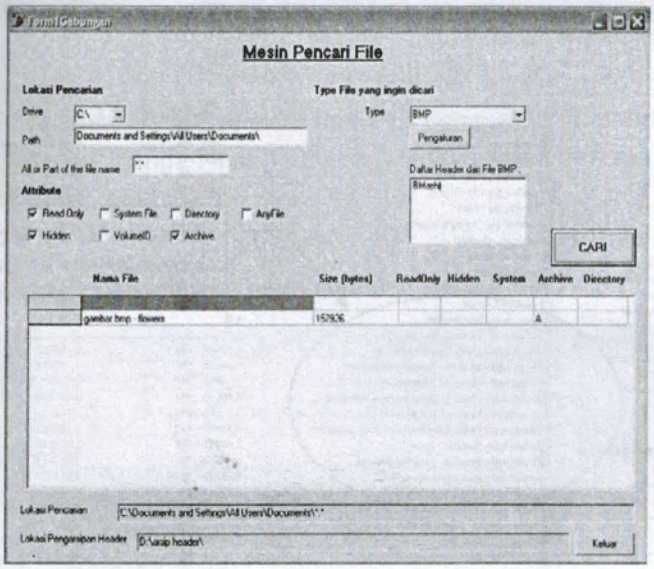
Dari Gambar 3.9 tampak bahwa file-file yang tidak ber-*extension* yang sebelumnya tidak dapat terdeteksi dengan mesin pencari windows dapat tampak melalui mesin pencari ini. Demikian pula hasil pencarian terhadap file-file dengan tipe yang lain dapat dilakukan dengan hasil sebagaimana disajikan pada Gambar 3.10, Gambar 3.11, dan Gambar 3.12.



Gambar 3.10 *Pencarian Terhadap File WMA*



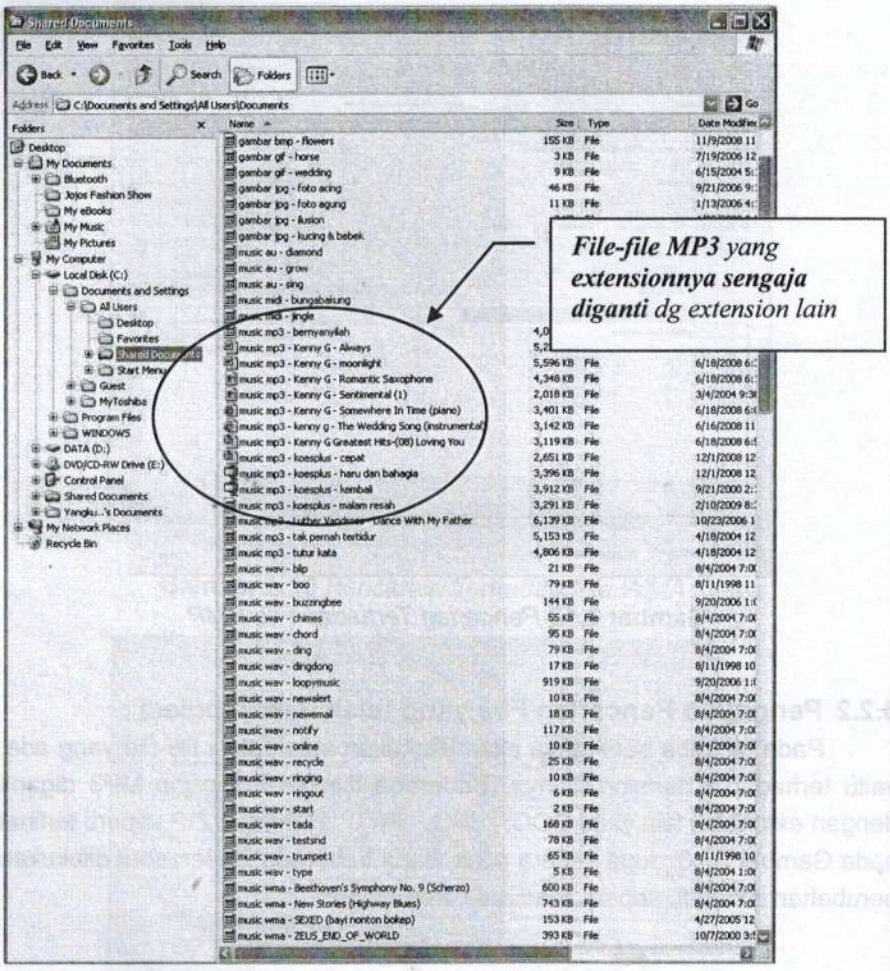
Gambar 3.11 *Pencarian Terhadap File JPG*



Gambar 3.12 *Pencarian Terhadap File BMP*

3.2.2 Pengujian Pencarian File yang telah dimanipulasi

Pada uji coba berikutnya akan dilakukan manipulasi file-file yang ada, yaitu terhadap *extension* filenya. Beberapa file ber-*extension* MP3 diganti dengan *extension* lain yaitu DOC, JPG, PPT, HTM dan ZIP seperti terlihat pada Gambar 3.13. Juga secara acak, pada beberapa file tersebut dilakukan perubahan *attribute* seperti *read only*, dan *hidden*.

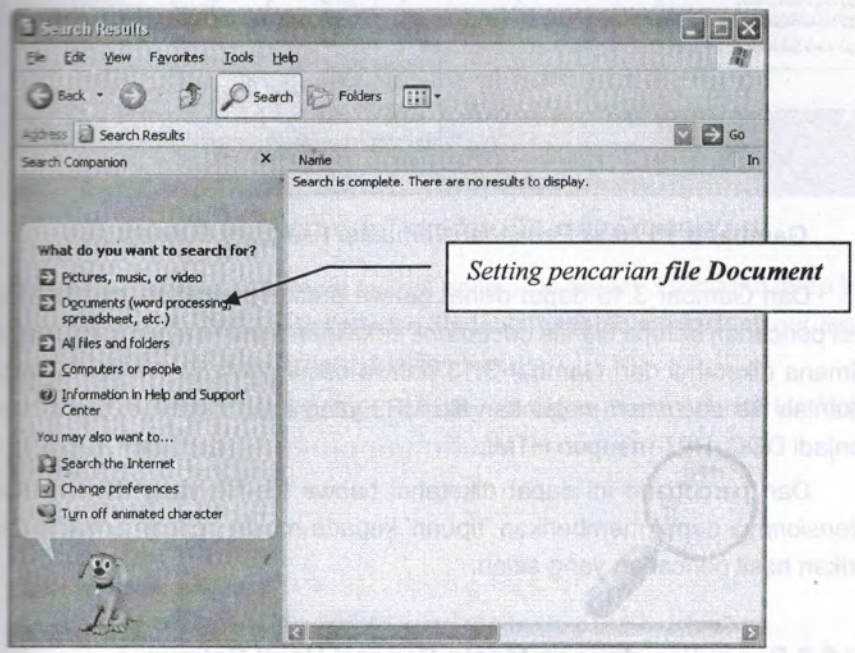


Gambar 3.13 *Mengganti Extension File mp3 dengan Extension Lain*

3.2.2.1 Pengujian dengan Mesin Pencari Konvensional

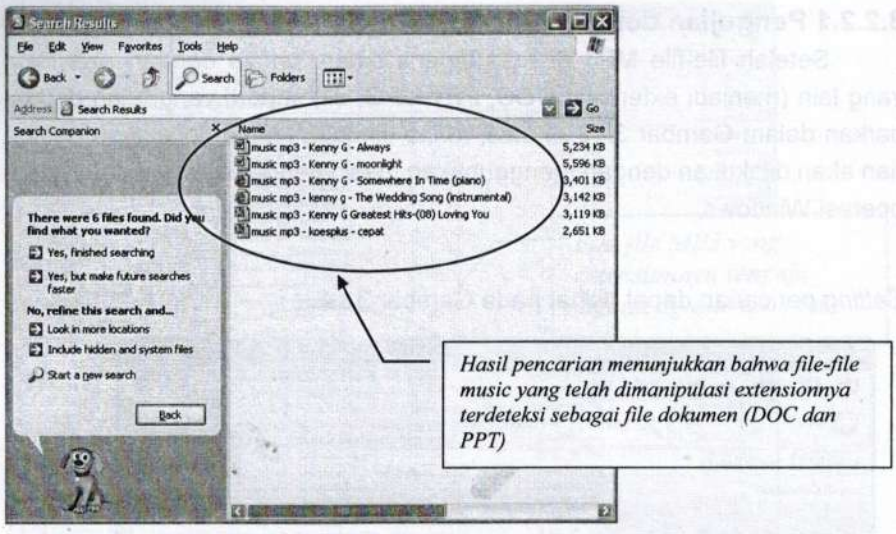
Setelah file-file MP3 di atas diganti *extension*nya dengan *extension* yang lain (menjadi extension DOC, PPT, JPG, dll) seperti yang telah digambarkan dalam Gambar 3.13 di atas, maka uji coba untuk melakukan pencarian akan dilakukan dengan menggunakan mesin pencari bawaan dari sistem operasi Windows.

Setting pencarian dapat dilihat pada Gambar 3.14.



Gambar 3.14 *Setting Pencarian File-File Document*

Setelah *setting* pencarian ditentukan dan proses pencarian dilakukan, akan didapatkan hasil pencarian seperti ditunjukkan pada Gambar 3.15.



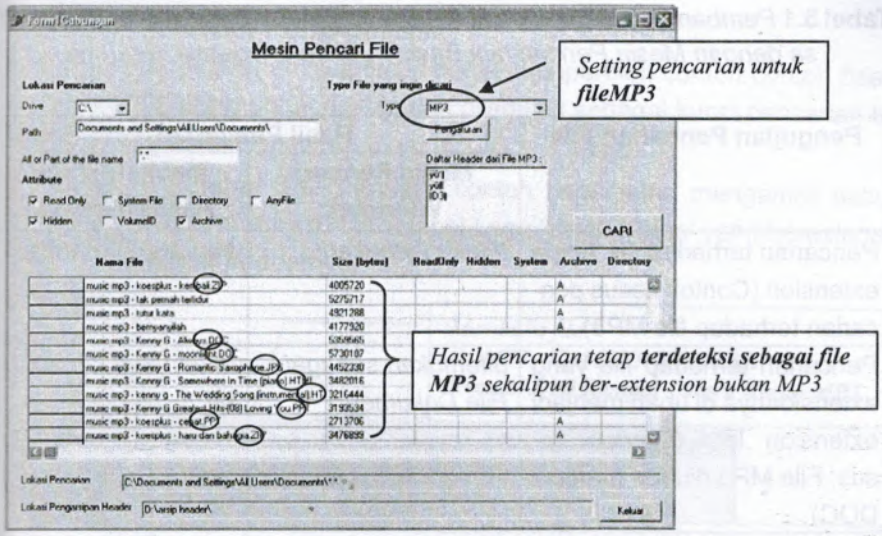
Gambar 3.15 *Hasil Pencarian Terhadap File yang Dimanipulasi*

Dari Gambar 3.15 dapat dilihat bahwa proses pencarian memberikan hasil pencarian berupa file-file *document*, sekalipun pada kenyataannya sebagaimana diketahui dari Gambar 3.13 bahwa sesungguhnya file-file tersebut bukanlah file *document* melainkan file MP3 yang telah diganti *extension*nya menjadi DOC, PPT maupun HTML.

Dari percobaan ini dapat diketahui bahwa file-file yang dimanipulasi *extension*nya dapat memberikan 'tipuan' kepada mesin pencari untuk memberikan hasil pencarian yang salah.

3.2.2.2 Pengujian dengan Mesin Pencari Hasil Rekayasa

Pengujian yang sama untuk mencari file yang ditunjukkan pada Gambar 3.13 juga dilakukan dengan menggunakan mesin pencari hasil rekayasa yang telah dibuat, dengan hasil seperti yang ditunjukkan pada Gambar 3.16.



Gambar 3.16 *Pencarian Terhadap File yang Dimanipulasi*

Pada Gambar 3.16 dapat terlihat bahwa file-file MP3 tersebut tetap terdeteksi sebagai file MP3 sekalipun file-file tersebut telah mengalami perubahan pada *extension*nya dengan *extension* lain.

Hasil pengujian beserta komparasinya dengan mesin pencari lain disajikan dalam Tabel 3.1.

Tabel 3.1 *Pembandingan Hasil pencarian antara Mesin Pencari Hasil Rekayasa dengan Mesin Pencari lain Bawaan Sistem Operasi Windows*

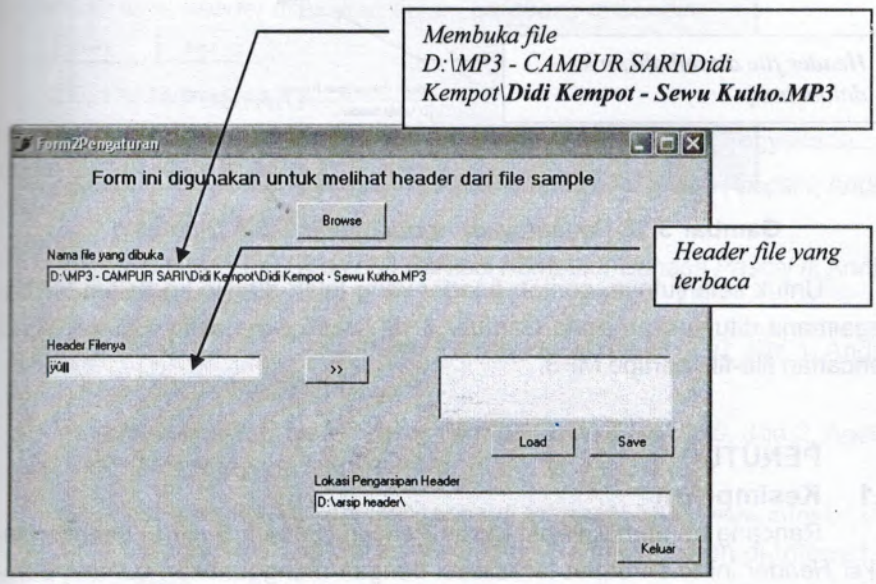
Pengujian Pencarian File	Hasil Pencarian	
	Mesin Pencari Windows	**Mesin Pencari Hasil Rekayasa**
Pencarian terhadap file tanpa extension (Contoh kasus pencarian terhadap file MP3)	*tidak ditemukan*	*berhasil ditemukan*
Pencarian terhadap file yang extensionnya di ubah menjadi extension lain (Contoh kasus: File MP3 diubah menjadi DOC)	*ditemukan sebagai File Dokumen*	*ditemukan sebagai MP3*
Pencarian terhadap file yang extensionnya di ubah menjadi extension lain (Contoh kasus: File MP3 diubah menjadi JPG)	*ditemukan sebagai File Gambar*	*ditemukan sebagai MP3*
Pencarian terhadap file yang extensionnya di ubah menjadi extension lain (Contoh kasus: File MP3 diubah menjadi HTML)	*ditemukan sebagai HTML*	*ditemukan sebagai MP3*

Dari Tabel 3.1 dapat dilihat bahwa tingkat keakurasian mesin pencari hasil rekayasa ini (dibandingkan dengan mesin pencari lain) dalam melakukan pencarian file, baik yang ber*extension*, tidak ber*extension* maupun yang *extension*nya tidak valid (*extension* telah diubah) menunjukkan hasil yang baik.

3.2.3 Pengaturan Pengarsipan

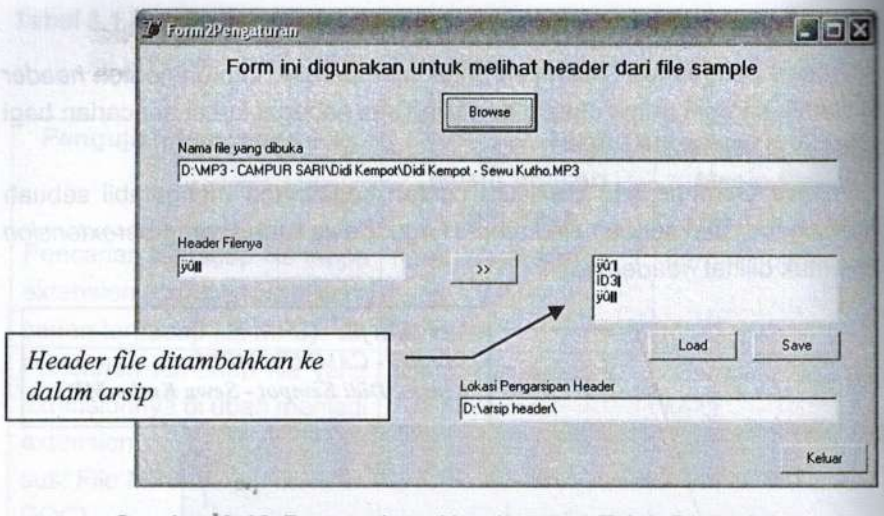
Form pengaturan diciptakan untuk memperoleh contoh-contoh *header* dari file-file sampel untuk diarsip dan dijadikan sebagai kunci pencarian bagi file-file yang sejenis.

Pada Gambar 3.17 disajikan contoh bagaimana mengambil sebuah contoh *header* dari sebuah file sampel Lagu 'Sewu Kutho' yang ber*extension* MP3 untuk dilihat *header*nya.



Gambar 3.17 *Pengambilan Contoh Header dari File Sample*

Hasil yang telah diperoleh seperti Gambar 3.17 dapat diarsipkan ke dalam file arsip, digabungkan menjadi satu bersama contoh-contoh header yang lain untuk sebuah tipe file yang sejenis, sebagaimana disajikan dalam Gambar 3.18.



Gambar 3.18 *Pengarsipan Header yang Telah Diperoleh*

Untuk selanjutnya, contoh *header* yang telah diarsip ke dalam file sebagaimana ditunjukkan pada Gambar 3.18, akan digunakan sebagai kunci pencarian file-file bertipe MP3.

4 PENUTUP

4.1 Kesimpulan

Rancang bangun Aplikasi Mesin Pencari (*Search Engine*) dengan Deteksi *Header* ini telah dapat terealisasi dengan menggunakan bahasa pemrograman Borland Delphi, dengan desain berbasis GUI, dan telah berhasil diuji-cobakan untuk melakukan sejumlah pencarian file. Dari hasil pengujian dan komparasinya terhadap mesin pencari lain menunjukkan bahwa Mesin Pencari hasil rekayasa ini telah berhasil menemukan file-file yang tidak dapat ditemukan oleh mesin pencari lain, yaitu file-file yang tidak dilengkapi dengan *filetype* atau *extension* maupun file-file yang tipe filenya telah sengaja 'diubah' dengan tipe yang bukan miliknya. Selain itu, Mesin Pencari hasil rekayasa ini juga dilengkapi kemampuan untuk mendeteksi file yang di*hidden*.

4.2 Saran

Dua buah file dengan tipe/ *extension* yang sama terkadang memiliki *header* file yang berbeda, sehingga semakin banyak 'koleksi' *header* yang dimiliki untuk sebuah tipe file akan membuat hasil pencarian yang semakin kaya. Dalam program aplikasi ini penyimpanan *header* dilakukan dalam file teks dan belum dilengkapi dengan *database* untuk menyimpan banyak jenis *header*, sehingga untuk pengembangan selanjutnya akan baik apabila untuk menyimpan jenis *header* dilakukan dalam *database* tersendiri.

5 DAFTAR PUSTAKA

1. Jogiyanto HM, 1988, *Pascal Tingkat Lanjutan*, Andi Offset, Yogyakarta.
2. Jogiyanto HM, 1989, *Teori dan Aplikasi Komputer Bahasa Pascal I*, Andi Offset, Yogyakarta.
3. Jogiyanto HM, 1989, *Teori dan Aplikasi Komputer Bahasa Pascal II*, Andi Offset, Yogyakarta
4. Kadir, Abdul, 2001, *Dasar-dasar Pemrograman Delphi 5.0*, Jilid 1, Andi Offset, Yogyakarta.
5. Kadir, Abdul, 2001, *Dasar-dasar Pemrograman Delphi 5.0*, Jilid 2, Andi Offset, Yogyakarta.
6. Kahartospisan, 2009, *Seni Pencarian di Internet*, http://www.infoskrip-si.com/Tips-and-Tricks/40-Google-Seni-Pencarian-di-Internet.html
7. Kurniawan, Yoga Dwi, 2008, *Steganografi sebagai Suatu Teknik Menyembunyikan Pesan dalam Suatu File Citra, Audio dan Video.* STMIK AKAKOM, Yogyakarta
8. Pramono, Sigit, 2008, *Image Steganography Berbasis LSB dengan Pembangkit Bilangan Acak*, STMIK AKAKOM, Yogyakarta
9. Programming with Delphi, Development Guide.
10. http://id.wikipedia.org