```
program PengolahKalimat;

uses
  Forms,
  MainUnit in 'MainUnit.pas' {MainForm},
  OLTools in 'OLTools.pas',
  OLClass in 'OLClass.pas',
  EditKamusUnit in 'EditKamusUnit.pas' {EditKamusForm},
  AboutUnit in 'AboutUnit.pas' {AboutForm};

{$R *.RES}

begin
  Application.Initialize;
  Application.Title := 'Pengolah Kalimat';
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TEditKamusForm, EditKamusForm);
  Application.CreateForm(TAboutForm, AboutForm);
  Application.Run;
End.
```

```
unit MainUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
   Forms,Dialogs,  Menus, ComCtrls, ToolWin, ExtCtrls, StdCtrls,
   ActnList,Contnrs, OLClass, OLTools, Grids;

const KamusFile = 'Rang_F.dat';
      BonesFile = 'Bones.dat';
      SkeletonFile = 'Skeleton.dat';

type
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    N1: TMenuItem;
    mnBuka: TMenuItem;
    Panel1: TPanel;
    StatusBar1: TStatusBar;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    OpenDialog: TOpenDialog;
    MyActionList: TActionList;
    BukaAction: TAction;
    KataStringGrid: TStringGrid;
    TreeView: TTreeView;
    FrasaStringGrid: TStringGrid;
    Splitter1: TSplitter;
    Panel7: TPanel;
    KalimatListBox: TListBox;
    Panel6: TPanel;
    TeksEdit: TEdit;
    UbahBtn: TButton;
    TambahBtn: TButton;
    Splitter3: TSplitter;
    mnSimpan: TMenuItem;
    SaveAction: TAction;
    SaveDialog: TSaveDialog;
    Label1: TLabel;
    StrukturLabel: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    FileNameLabel: TLabel;
    Bevel1: TBevel;
    Bevel2: TBevel;
    ProsesBtn: TButton;
    HapusBtn: TButton;
    Bevel3: TBevel;
    Bevel4: TBevel;
    Label2: TLabel;
    Edit1: TMenuItem;
```

```
    EditKamus: TMenuItem;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    N2: TMenuItem;
    ReloadAturan: TMenuItem;
    N3: TMenuItem;
    Info1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    procedure Exit1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose:
                             Boolean);
    procedure ActionExecute(Sender: TObject);
    procedure KalimatListBoxClick(Sender: TObject);
    procedure UbahBtnClick(Sender: TObject);
    procedure TambahBtnClick(Sender: TObject);
    procedure ProsesBtnClick(Sender: TObject);
    procedure HapusBtnClick(Sender: TObject);
    procedure KalimatListBoxDblClick(Sender: TObject);
    procedure EditKamusClick(Sender: TObject);
    procedure ReloadAturanClick(Sender: TObject);
    procedure Info1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure About1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure DisplayWordList;
    procedure DisplayFrasaList( p : integer);
    procedure BacaFileText(s : string);
    procedure SimpanFileText(s : string);
    procedure ClearGrid( sg : TStringGrid);
    procedure DisplayFinalTree(cab : TTreeNode; pt: TParseTree);
    procedure Proses(kal : string);
    procedure ReloadRules;
  end;

var
    MainForm: TMainForm;
    KamusRec : array of TKamusRec;
    BonesRec : TBonesRec;
    SkeletonRec : TSkeletonRec;
    TreeRec : TObjectList;
    FinalTree : TParseTree;
    ft, ft2 : Text;
    kalimatPilih : string;
    adaError : boolean;
    runonce : boolean = true;
    TandaBaca : set of char;
```

```
implementation

uses EditKamusUnit, AboutUnit;

{$R *.DFM}


//================================================

procedure ReadKamus;
var s, s2, err_msg : string;
    i : integer;
    error_found : boolean;
begin
     AssignFile(ft, KamusFile);
     Reset(ft);
     error_found := false;
     while not EOF(ft) and not error_found do
     begin
          Readln(ft,s);

          AssignFile(ft2,s);
{$I-}
          Reset(ft2);
{$I+}
          try
               if IOResult <> 0 then
               begin
                    error_found := true;
                    err_msg := 'Error when opening file ' + s;
               end else
               begin
                    SetLength(KamusRec, Length(KamusRec)+1);
                    i := Length(KamusRec)-1;
                    KamusRec[i].filename := s;
                    Readln(ft2,s2);
                    s2 := UpperCase(s2);
                    if RemoveTag(s2) = ttNoTag then
                    begin
                         error_found := true;
                         err_msg := 'Tipe file kamus tidak
                                    terdefinisi' + s;
                    end else
                    begin
                         KamusRec[i].boneID := BonesRec.GetPos(s2,
                          false);
                         if KamusRec[i].boneID<0 then
                         begin
                              ShowMessage('Tidak dapat
                              menerjemahkan ke kata
                              kunci :'+s2);
                              exit;
                         end;
                         KamusRec[i].code := s2;
                    end;
               end;
```

```
            finally
                CloseFile(ft2);
            end;
        end;
        CloseFile(ft);

        if error_found then
        begin
            ShowMessage(err_msg);
            exit;
        end;
end;

procedure ReadBones;
var s : string;
begin
        AssignFile(ft, BonesFile);
        Reset(ft);
        while not EOF(ft) do
        begin
            Readln(ft,s);
            s := UpperCase(StripSpaces(s));
            BonesRec.Add(s);
        end;
end;

procedure ReadSkeleton;
var s, s2, bonesSeq : string;
    bID : char;
    i, j : integer;
    target : integer;
    aos : ArrayOfString;
begin
        bonesSeq := '';
        SetLength(aos,0);

        AssignFile(ft, SkeletonFile);
        Reset(ft);
        while not EOF(ft) do
        begin
            Readln(ft,s);

            target := -1;
            aos := CreateTokens(' ',s);
            for i := 0 to Length(aos) do
            begin
                if i<Length(aos) then s2 := UpperCase(aos[i]) else
                s2 := '|';
                if s2 = '::=' then
                begin
                    s2 := UpperCase(aos[i-1]);
                    RemoveTag(s2);
                    target := BonesRec.GetPos(s2, true);
                end
                else if s2 = '|' then
                begin
```

```
                SkeletonRec.AddObject(bonesSeq,Pointer(target));
                bonesSeq := '';
                end
                else
                begin
                     if target>=0 then
                     begin
                          RemoveTag(s2);
                          j := BonesRec.GetPos(s2, true);
                          bID := Chr(j + AsciiStart);
                          bonesSeq := bonesSeq + bID;
                     end;
                end;
           end;
      end;
      CloseFile(ft);
end;

//------------------------------------------------------
function CheckKamus(s : string):integer;
var i, ll : integer;
    ada : boolean;
    s2 : string;
begin
      ll := length(KamusRec);
      ada := false;
      result := -1;
      for i:= 0 to ll-1 do
      begin
           AssignFile(ft,KamusRec[i].filename);
           Reset(ft);
           while not EOF(ft) do
           begin
                Readln(ft,s2);
                s2 := UpperCase(StripSpaces(s2));
                if s2 = UpperCase(s) then
                begin
                     ada := true;
                     break;
                end;
           end;
           CloseFile(ft);
           if ada then begin result := KamusRec[i].boneID; break;
end;
      end;
end;

function BuildBaseParseTree(s: string):integer;
var ss, ss2, smsg : string;
    aos : ArrayOfString;
    i, aid, no : integer;
begin
      TreeRec.Clear;
//-------langkah 1, buat token (pisahkan kata kata)
      ss := Check4CommaDot(s);
      aos := CreateTokens(' ',ss);
```

```
//-------langkah 2, terjemahkan kata kunci dalam daftar BNF ke
                    kode
//-------terlebih dahulu bandingkan dengan kamus
     result := -1;
     for i := 0 to Length(aos)-1 do
     begin
          if (aos[i][1] in TandaBaca) then break;
          ss2 := UpperCase(aos[i]);
          aid := CheckKamus(ss2);       //periksa 1: kamus
          if aid<0 then aid := BonesRec.GetPos(ss2); //periksa 2:
                                                       selain yg
                                                       ada di
                                                       kamus
          if aid<0 then                               //periksa 3:
                                                       mungkin
                                                       adalah
                                                       Nomina(ora
                                                       ngatau
                                                       barang)
          begin
               smsg := 'Kata "'+ss2+'" tak ada di kamus maupun
               kata kunci, apakah merupakan orang/barang?';
               if MessageDlg(smsg,mtConfirmation,[mbYes,mbNo],0) =
                  mrYes then aid := BonesRec.GetPos('NOM', false)
                                  else begin
                                       adaError := true;
                                       ShowMessage('Silakan
                                       perbaiki struktur
                                       kalimat Anda.');
                                       Exit;
                                  end;
          end;

          if aid>=0 then
          begin
               no := TreeRec.Add(TParseTree.Create);
               TParseTree(TreeRec.Items[no]).Text := aos[i];
               TParseTree(TreeRec.Items[no]).Tipe := Chr(aid +
               AsciiStart);
          end;
     end;
     result := TreeRec.Count;
end;

function Buat_Frasa:boolean;
var selesai, ada : boolean;
    mli, mulai : integer;
    i, p, len, ll, adacount : integer;
    s : string;
begin
     selesai := false;
     mulai := 0;
     len := TreeRec.Count-1;

     //buat copy untuk semua kata
     for i:= mulai to len do
```

```
begin
     p := TreeRec.Add(TParseTree.Create);
     TParseTree(TreeRec.Items[p]).Tipe:=
      TParseTree(TreeRec.Items[i]).Tipe;
     TParseTree(TreeRec.Items[p]).Text:=
      TParseTree(TreeRec.Items[i]).Text;
end;

mulai := len + 1;
len := mulai + len;
mli := mulai;
adacount := 0;
while not selesai do
begin
     ada := false;
     ll := len;
     p := -1;

     while (not ada)and(mulai<=ll) do
     begin
         s := '';
         for i:= mulai to ll do s := s +
       TParseTree(TreeRec.Items[i]).Tipe;
         p := SkeletonRec.IndexOf(s);
         if p>=0 then ada := true else Dec(ll);
     end;

     if p>=0 then
     begin
          TParseTree(TreeRec.Items[mulai]).Tipe:=
          Chr(Integer(SkeletonRec.Objects[p])+AsciiStart);
          if length(s)>1 then
          begin
               s := '';
               for i:= ll downto mulai+1 do
               begin
                    s:=TParseTree(TreeRec.Items[i]).Text + s;
                    if i<>mulai+1 then s := s + ' ';
                    TreeRec.Delete(i);
                    Dec(len);
               end;
               TParseTree(TreeRec.Items[mulai]).Text:=
              TParseTree(TreeRec.Items[mulai]).Text +' '+ s;
          end;
          Inc(adacount);
     end else
     begin
          if mulai<len then Inc(mulai)
          else begin
               if adacount>0 then
               begin
                    mulai := mli;
                    len := TreeRec.Count-1;
                    adacount := 0;
               end else
                    selesai := true;
```

```
                end
            end;

        end;

        result := false;
        for i := mli to TreeRec.Count-1 do
            if TParseTree(TreeRec.Items[i]).Tipe =
             BonesRec.AsChar('FR_VERB') then
            begin
                result := true;
                break;
            end;
    end;

    procedure BuatFinalTree( mulai : integer);

        procedure Fr_Nom(var lastP : TParseTree; tF : TParseTree);
        var pt : TParseTree;
        begin
            pt := lastP;

            if (lastP.TopParent.KakiKanan And
             lastP.TopParent.KakiKiri) = -1 then
            begin
                lastP.Kiri:=
                TParseTree.Create(BonesRec.GetPos('S'),tF.Text,
                lastP.TopParent);
                pt := lastP.Kiri;
            end else
            if lastP.TipeInt = BonesRec.GetPos('P') then
            begin
                lastP.Kiri:=TParseTree.Create(BonesRec.GetPos('P'),
                  lastP.Text,lastP.TopParent);
                lastP.Kanan:=TParseTree.Create(BonesRec.GetPos('O'),
                  tF.Text,lastP.TopParent);

                lastP.Text := lastP.Text + ' ' + tF.Text;
                pt := lastP.Kanan;
            end else
            if lastP.TopParent.KakiKiri = BonesRec.GetPos('KET')
             then
            begin
                lastP.Tipe := BonesRec.AsChar('S');
                lastP.Text := lastP.Text + ' ' + tf.Text;

               lastP.Kiri:=TParseTree.Create(BonesRec.GetPos('KET'),
                  lastP.Text,lastP.TopParent);
                lastP.Kanan:=TParseTree.Create(BonesRec.GetPos('S'),
                  tF.Text,lastP.TopParent);
                pt := lastP.Kanan;
            end;

            lastP := pt;
        end;
```

```
procedure Fr_Verb(var lastP : TParseTree; tF : TParseTree);
var pt : TParseTree;
begin
     if not lastP.TopParent.CariTipe(BonesRec.GetPos('P'))
     then    //jika belum ada P
     begin
          lastP.TopParent.Kanan :=
            TParseTree.Create(BonesRec.GetPos('P'),
            tf.Text,lastP.TopParent);
          pt := lastP.TopParent.Kanan;
     end else
             //jika sudah ada P
     begin
        lastP.Kiri:=TParseTree.Create(BonesRec.GetPos('P'),
            lastP.Text,lastP.TopParent);
        lastP.Kanan:=TParseTree.Create(BonesRec.GetPos('PL'),
            tF.Text,lastP.TopParent);

        lastP.Text := lastP.Text + ' ' + tF.Text;
        pt := lastP.Kanan;
     end;

     lastP := pt;
end;

procedure Fr_Adv(var lastP : TParseTree; tF : TParseTree);
var pt : TParseTree;
begin
     if (lastP.TopParent.KakiKanan And
      lastP.TopParent.KakiKiri) = -1 then
     begin
       lastP.Kiri:=TParseTree.Create(BonesRec.GetPos('KET'),
       tF.Text,lastP.TopParent);
       pt := lastP.Kiri;
     end else
     begin
       lastP.Kiri:=
       TParseTree.Create(lastP.TipeInt,lastP.Text,
       lastP.TopParent);
       lastP.Kanan:=
       TParseTree.Create(BonesRec.GetPos('KET'),tF.Text,
       lastP.TopParent);

       lastP.Text := lastP.Text + ' ' + tF.Text;

       pt := lastP.Kanan;
     end;

     lastP := pt;
end;
```

```
procedure Fr_Waktu(var lastP : TParseTree; tF : TParseTree);
var pt : TParseTree;
begin
     if (lastP.TopParent.KakiKanan And
      lastP.TopParent.KakiKiri) = -1 then
     begin
         lastP.Kiri:=
        TParseTree.Create(BonesRec.GetPos('KET'),tF.Text,
         lastP.TopParent);
         pt := lastP.Kiri;
     end else
     begin
         lastP.Kiri:=
          TParseTree.Create(lastP.TipeInt,lastP.Text,
          lastP.TopParent);
         lastP.Kanan:=
        TParseTree.Create(BonesRec.GetPos('KET'),tF.Text,
         lastP.TopParent);
         lastP.Text := lastP.Text + ' ' + tF.Text;

          pt := lastP.Kanan;
     end;

     lastP := pt;
end;

procedure Fr_Tempat(var lastP : TParseTree; tF : TParseTree);
var pt : TParseTree;
begin
     if (lastP.TopParent.KakiKanan And
      lastP.TopParent.KakiKiri) = -1 then
     begin
         lastP.Kiri:=
        TParseTree.Create(BonesRec.GetPos('KET'),tF.Text,
         lastP.TopParent);
          pt := lastP.Kiri;
     end else
     begin
         lastP.Kiri:=
        TParseTree.Create(lastP.TipeInt,lastP.Text,
         lastP.TopParent);
        lastP.Kanan:=
        TParseTree.Create(BonesRec.GetPos('KET'),tF.Text,
        lastP.TopParent);
          lastP.Text := lastP.Text + ' ' + tF.Text;

          pt := lastP.Kanan;
     end;

     lastP := pt;
end;
```

```
        procedure Fr_Adj(var lastP : TParseTree; tF : TParseTree);
        var pt : TParseTree;
        begin
             pt := lastP;

             if(lastP.TipeInt=BonesRec.GetPos('S'))or
             (lastP.TipeInt= BonesRec.GetPos('O')) then
             begin
                  lastP.Text := lastP.Text + ' ' + tF.Text;
             end else
             if lastP.TipeInt = BonesRec.GetPos('P') then
             begin
                  lastP.Kiri:=
                  TParseTree.Create(lastP.TipeInt,lastP.Text,
                  lastP.TopParent);
                  lastP.Kanan:=
                  TParseTree.Create(BonesRec.GetPos('PL'),tF.Text,
                  lastP.TopParent);
                  lastP.Text := lastP.Text + ' ' + tF.Text;

                  pt := lastP.Kanan;
             end;

             lastP := pt;
        end;


        procedure Jeda(var lastP : TParseTree; tF : TParseTree);
        begin
             lastP.Kiri:= TParseTree.Create(lastP.TipeInt,lastP.Text,
             lastP.TopParent);
             lastP.Kanan :=
              TParseTree.Create(tf.TipeInt,tF.Text,lastP.TopParent);

             lastP := lastP.Kanan;
             lastP.Kanan :=
              TParseTree.Create(BonesRec.GetPos('KALIMAT'),'',nil);
             lastP.Kanan.TopParent := lastP.Kanan;

             lastP := lastP.Kanan;
        end;

    var i : integer;
        tipeFrasa : string;
        pAkhir, thisFrasa : TParseTree;
    begin
         FinalTree.Text := '';
         FinalTree.Tipe := Chr(BonesRec.GetPos('KALIMAT',false) +
          AsciiStart);
         pAkhir := FinalTree;
         for i := mulai to TreeRec.Count-1 do
         begin
              thisFrasa := TreeRec.Items[i] as TParseTree;
              tipeFrasa := BonesRec.Items[Ord(thisFrasa.Tipe)-
              AsciiStart];
```

```
            if tipeFrasa = 'FR_NOM' then Fr_Nom(pAkhir, thisFrasa)
             else
            if tipeFrasa = 'FR_VERB' then Fr_Verb(pAkhir, thisFrasa)
             else
            if (tipeFrasa = 'JEDA')or(tipeFrasa = 'KOORD')
            or(tipeFrasa = 'SUBORD') then Jeda(pAkhir, thisFrasa)
            else
            if (tipeFrasa = 'FR_WAKTU')or(tipeFrasa = 'FR_PREP')
            then Fr_Waktu(pAkhir, thisFrasa) else
            if tipeFrasa = 'FR_ADJ' then Fr_Adj(pAkhir, thisFrasa)
             else
            if tipeFrasa = 'FR_ADV' then Fr_ADV(pAkhir, thisFrasa)
             else
            if tipeFrasa = 'FR_TEMPAT' then Fr_Tempat(pAkhir,
             thisFrasa)
            else begin
                 ShowMessage('Terdapat jenis kata yang tidak
                             dikenal, silakan perbaiki kalimat
                             Anda');
                 break;
            end;

            pAkhir.TopParent.Text := pAkhir.TopParent.Text + ' ' +
             thisFrasa.Text;
     end;
end;

//---------------------------------------
procedure Init;
begin
     BonesRec := TBonesRec.Create;
     SkeletonRec := TSkeletonRec.Create;
          SkeletonRec.Sorted := false;
     TreeRec := TObjectList.Create(true);
     FinalTree := TParseTree.Create;
     FinalTree.TopParent := FinalTree;
end;

procedure DeInit;
begin
     FinalTree.Free;
     TreeRec.Free;
     SkeletonRec.Free;
     BonesRec.Free;end;
//-------------------------

procedure TMainForm.ClearGrid( sg : TStringGrid);
var i, j:integer;
begin
     for i := 0 to sg.ColCount-1 do
          for j := 0 to sg.RowCount-1 do
               sg.Cells[i,j] := '';
end;
```

```
procedure TMainForm.DisplayWordList;
var i : integer;
begin
     ClearGrid(KataStringGrid);
     for i:= 0 to TreeRec.Count-1 do
     begin
          KataStringGrid.Cells[0,i]:=
           TParseTree(TreeRec.Items[i]).Text;
          KataStringGrid.Cells[1,i]:=
           BonesRec.Items[Ord(TParseTree(TreeRec.Items[i]).Tipe)-
           AsciiStart];
     end;
end;

procedure TMainForm.DisplayFrasaList(p : integer);
var i : integer;
begin
     ClearGrid(FrasaStringGrid);
     for i := p to TreeRec.Count-1 do
     begin
          FrasaStringGrid.Cells[0,i-p] :=
           TParseTree(TreeRec.Items[i]).Text;
          FrasaStringGrid.Cells[1,i-p]:=
           BonesRec.Items[Ord(TParseTree(TreeRec.Items[i]).Tipe)-
           AsciiStart];
     end;
end;

procedure TMainForm.BacaFileText(s : string);
var ss : string;
begin
     KalimatListBox.Items.Clear;

     AssignFile(ft,s);
     Reset(ft);
     While not EOF(ft) do
     begin
          Readln(ft,ss);
          KalimatListBox.Items.Add(ss);
     end;
     CloseFile(ft);
end;

procedure TMainForm.SimpanFileText(s : string);
var ss : string;
    i : integer;
begin
     AssignFile(ft,s);
     Rewrite(ft);
     for i := 0 to KalimatListBox.Items.Count-1 do
     begin
          ss := KalimatListBox.Items[i];
          Writeln(ft,ss);
     end;
     CloseFile(ft);end;
```

```
procedure TMainForm.DisplayFinalTree(cab : TTreeNode; pt:
TParseTree);
var tn : TTreeNode;
    s : string;
begin
     if pt.Kiri <> nil then
     begin
          s:=BonesRec.Items[Ord(pt.Kiri.Tipe)-
           AsciiStart]+'('+pt.Kiri.Text+ ')';
          tn := TreeView.Items.AddChild(cab, s);
          DisplayFinalTree(tn, pt.Kiri);
     end;
     if pt.Kanan <> nil then
     begin
        s:=BonesRec.Items[Ord(pt.Kanan.Tipe)-
         AsciiStart]+'('+pt.Kanan.Text+ ')';
        tn := TreeView.Items.AddChild(cab, s);
        DisplayFinalTree(tn, pt.Kanan);
     end;
end;

procedure TMainForm.ReloadRules;
begin
     SkeletonRec.Clear;
     SetLength(KamusRec,0);
     BonesRec.Clear;

     ReadBones;
     ReadKamus;
     ReadSkeleton;
end;

//--------------------------

procedure TMainForm.FormCreate(Sender: TObject);
begin
     Init;
     ReadBones;
     ReadKamus;
     ReadSkeleton;

     FrasaStringGrid.ColWidths[0] := 180;
     OpenDialog.InitialDir := GetCurrentDir;
     SaveDialog.InitialDir := GetCurrentDir;

     TandaBaca := ['.','?','!'];
end;

procedure TMainForm.FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
begin
     DeInit;
end;
```

```
procedure TMainForm.Exit1Click(Sender: TObject);
begin
     Close;
end;

procedure TMainForm.Proses(kal : string);

     function JenisKalimat: string;
     begin
          If((FinalTree.CariTipe(BonesRec.GetPos('SUBORD')))and
           (FinalTree.CariTipe(BonesRec.GetPos('KOORD'))))then
                    result := 'KALIMAT GABUNGAN'
          else if FinalTree.CariTipe(BonesRec.GetPos('SUBORD'))
               then
                    result := 'KALIMAT MAJEMUK BERTINGKAT/LUAS'
          else if (FinalTree.CariTipe(BonesRec.GetPos('KOORD')))or
               (FinalTree.CariTipe(BonesRec.GetPos('JEDA'))) then
                    result := 'KALIMAT MAJEMUK SETARA'
          else result := 'KALIMAT SEDERHANA/TUNGGAL'
     end;

var i, p : integer;
begin
     adaError := false;
     FinalTree.FreeChild;

     kalimatPilih := kal;

     BuildBaseParseTree(kalimatPilih);
     if not adaError then
     begin
          DisplayWordList;
          p := TreeRec.Count;
          if not Buat_Frasa then
          begin
               ShowMessage('Kalimat tidak berpredikat kata kerja,
                           silakan diperbaiki.');
               exit;
          end;
          DisplayFrasaList(p);
          BuatFinalTree(p);

          StrukturLabel.Caption := JenisKalimat;

          for i := TreeView.Items.Count-1 downto 1 do
           TreeView.Items[i].Free;
          DisplayFinalTree(TreeView.Items[0],FinalTree);

          TreeView.Items[0].Text:='KALIMAT'+'('+kalimatPilih+')';
          TreeView.FullExpand;
     end;
end;
```

```
procedure TMainForm.ActionExecute(Sender: TObject);
begin
     case TAction(Sender).Tag of
          1 :  if OpenDialog.Execute then
               begin
                    BacaFileText(OpenDialog.FileName);
                    FileNameLabel.Caption:=
                     ExtractFileName(OpenDialog.FileName);
               end;
          2 : if SaveDialog.Execute then
              begin
                    SimpanFileText(SaveDialog.FileName);
                   FileNameLabel.Caption:=
                   ExtractFileName(SaveDialog.FileName);
              end;
     end;
end;

procedure TMainForm.KalimatListBoxClick(Sender: TObject);
begin
     TeksEdit.Text :=
      KalimatListBox.Items[KalimatListBox.ItemIndex];
end;

procedure TMainForm.UbahBtnClick(Sender: TObject);
begin
     KalimatListBox.Items[KalimatListBox.ItemIndex] :=
      TeksEdit.Text;
end;

procedure TMainForm.TambahBtnClick(Sender: TObject);
begin
     KalimatListBox.Items.Add(TeksEdit.Text);
end;

procedure TMainForm.ProsesBtnClick(Sender: TObject);
begin
     Proses(TeksEdit.Text);
end;

procedure TMainForm.HapusBtnClick(Sender: TObject);
begin
     if Message Dlg ('Apakah Anda yakin kalimat berikut akan
      dihapus?',mtConfirmation,[mbYes,mbNo],0) = mrYes then
          KalimatListBox.Items.Delete(KalimatListBox.ItemIndex);
end;

procedure TMainForm.KalimatListBoxDblClick(Sender: TObject);
begin
     Proses(KalimatListBox.Items[KalimatListBox.ItemIndex]);
end;

procedure TMainForm.EditKamusClick(Sender: TObject);
begin
     EditKamusForm.ShowModal;
end;
```

```
procedure TMainForm.ReloadAturanClick(Sender: TObject);
begin
     if MessageDlg('Aturan akan di-update?',mtConfirmation,[mbYes,
      mbNo],0)<> mrYes then exit;
     ReloadRules;
end;

procedure TMainForm.Info1Click(Sender: TObject);
var msgstr : string;
begin
     msgstr := 'Jumlah kata kunci =
     '+IntToStr(BonesRec.Count)+#13#10;
     msgstr := msgstr + 'Jumlah kamus =
     '+IntToStr(Length(KamusRec))+#13#10;
     msgstr := msgstr + 'Jumlah aturan =
     '+IntToStr(SkeletonRec.Count);
     ShowMessage(msgstr);
end;

procedure TMainForm.FormActivate(Sender: TObject);
begin
     if runonce then
     begin
          AboutForm.ShowModal;
          runonce := false;
     end;
end;

procedure TMainForm.About1Click(Sender: TObject);
begin
     AboutForm.ShowModal;
end;

end.
```

```
unit OLTools;

interface

uses SysUtils;

type TTagTyle = (ttNoTag, ttPanah, ttPetik);
     ArrayOfString = array of string;

function StripSpaces(s: string):string;
function RemoveTag(var s:string):TTagTyle;
function CreateTokens(divider: char; s : string):ArrayOfString;
function Check4CommaDot(s : string):string;

implementation

function StripSpaces(s : string):string;
var i : integer;
begin
     result := '';
     for i := 1 to Length(s) do if (s[i]<>' ')and(Ord(s[i])<>9)
      then result := result + s[i];
end;

function RemoveTag(var s:string):TTagTyle;
begin
     if (s[1] = '<')and(s[Length(s)] = '>') then result := ttPanah
     else if (s[1] = '''')and(s[Length(s)] = '''') then result :=
      ttPetik
     else result := ttNoTag;
     if result <> ttNoTag then s := Copy(s,2,Length(s)-2);
end;

function CreateTokens(divider: char; s : string):ArrayOfString;
var i : integer;
    ps : string;
begin
     SetLength(result,0);
     i := 1;
     ps := '';
     while i<=Length(s) do
     begin
          if s[i] <> divider then
               ps := ps + s[i]
          else begin
               if Length(ps)>0 then
               begin
                    SetLength(result, Length(result) + 1);
                    result[Length(result)-1] := ps;
                    ps := '';
               end;
          end;
          Inc(i);
     end;
     SetLength(result, Length(result) + 1);
     result[Length(result)-1] := ps;
```

```
end;

function Check4CommaDot(s : string):string;
var i : integer;
begin
     result := '';
     for i := 1 to Length(s) do
          if s[i] = ',' then result := result + ' , '
          else if s[i] = '.' then result := result + ' .'
          else result := result + s[i];
end;

end.
```

```
unit OLClass;

interface

uses SysUtils, OLTools, Classes;

const AsciiStart = 120;

type TKamusRec = record
          filename : string[20];
          code : string[20];
          boneID : integer;
     end;

     TBonesRec = class
     private
          FBones : array of string;
          function Get(i : integer):string;
     public
          property Items[i : integer]: string read Get;
          constructor Create;
          destructor Destroy;override;
          function Count:integer;
          function Add(s : string):integer;
          function GetPos(s : string): integer; overload;
          function GetPos(s : string; force : boolean): integer;
           overload;
          function AsChar(s : string):char;
          procedure Clear;
     end;

     TSkeletonRec = class(TStringList);

     TParseTree = class;
     TParseTree = class
     public
          Text : string;
          Tipe : char;
          Parent : TParseTree;
          TopParent : TParseTree;
          Kanan : TParseTree;
          Kiri : TParseTree;
          constructor Create; overload;
          constructor Create( tp : integer; txt : string; prnt :
           TParseTree); overload;
          destructor Destroy; override;
          function KakiKanan : integer;
          function KakiKiri : integer;
          function CariTipe(e : integer) : boolean;
          function CekTipe(e : integer) : boolean;
          function TipeInt: integer;
          procedure FreeChild;
     end;
```

```
implementation

constructor TBonesRec.Create;
begin
     inherited;
     SetLength(FBones,0);
end;

destructor TBonesRec.Destroy;
begin
     SetLength(FBones,0);
     inherited Destroy;
end;

function TBonesRec.Count:integer;
begin
     result := Length(FBones);
end;

function TBonesRec.Add(s : string):integer;
begin
     SetLength(FBones, Length(FBones)+1);
     FBones[Length(FBones)-1] := s;
     result := Length(FBones)-1;
end;

function TBonesRec.Get(i : integer):string;
begin
     result := FBones[i];
end;

function TBonesRec.GetPos(s : string):integer;
var i : integer;
begin
     result := -1;
     for i := 0 to Count-1 do if s=FBones[i] then begin result:=i;
    break; end;
end;

function TBonesRec.GetPos(s : string; force : boolean):integer;
var i : integer;
begin
     result := -1;
     for i := 0 to Count-1 do if s=FBones[i] then begin result:=i;
      break; end;
     if (result<0)and(force) then
     begin
          Add(s);
          result := Count-1;
     end;
end;
```

```
function TBonesRec.AsChar(s : string):char;
var i : integer;
begin
     i := GetPos(s);
     if i>=0 then result := Chr(i+AsciiStart) else result := #0;
end;

procedure TBonesRec.Clear;
begin
     SetLength(FBones,0);
end;

//=========================================

constructor TParseTree.Create;
begin
     Parent := nil;
     Kanan := nil;
     Kiri := nil;
end;

constructor TParseTree.Create( tp : integer; txt : string; prnt :
TParseTree);
begin
     Kanan := nil;
     Kiri := nil;
     Parent := nil;
     TopParent := prnt;

     Text := txt;
     Tipe := Chr(tp + AsciiStart);
end;

destructor TParseTree.Destroy;
begin
     Kanan.Free;
     Kiri.Free;
     inherited Destroy;
end;

function TParseTree.KakiKanan : integer;
begin
     if Kanan <> nil then result := Ord(Kanan.Tipe)-AsciiStart
else result := -1;
end;

function TParseTree.KakiKiri : integer;
begin
     if Kiri <> nil then result := Ord(Kiri.Tipe)-AsciiStart else
result := -1;
end;
```

```
function TParseTree.CariTipe(e : integer) : boolean;
begin
     if (Ord(Tipe)-AsciiStart) = e then result := true
     else
     begin
          result := false;
          if Kanan <> nil then result := Kanan.CariTipe(e);
          if (not result)and (Kiri <> nil) then result :=
           Kiri.CariTipe(e);
     end;
end;

function TParseTree.CekTipe(e : integer) : boolean;
begin
 if(Ord(Tipe)-AsciiStart)=e then result:=true else result:= false;
end;

function TParseTree.TipeInt: integer;
begin
     result := Ord(Tipe) - AsciiStart;
end;

procedure TParseTree.FreeChild;
begin
     if Kiri<>nil then Kiri.Free;
     if Kanan<>nil then Kanan.Free;
     Kiri := nil;
     Kanan := nil;
end;

end.
```

```
unit EditKamusUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  Grids, ExtCtrls, StdCtrls;

type
  TEditKamusForm = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    KamusStringGrid: TStringGrid;
    EditMemo: TMemo;
    CloseButton: TButton;
    Label1: TLabel;
    KamusLabel: TLabel;
    SaveButton: TButton;
    SortBtn: TButton;
    procedure FormShow(Sender: TObject);
    procedure KamusStringGridDblClick(Sender: TObject);
    procedure SaveButtonClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose:
     Boolean);
    procedure SortBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ClearGrid;
    procedure BacaFile(fn : string);
    procedure SimpanFile(fn : string);
    procedure ClearMemo;
  end;

var
  EditKamusForm: TEditKamusForm;
  ft : Text;
  lastfile : string;

implementation

uses MainUnit, OLClass, OLTools;

{$R *.DFM}

procedure TEditKamusForm.ClearMemo;
begin
     EditMemo.Lines.Clear;
end;
```

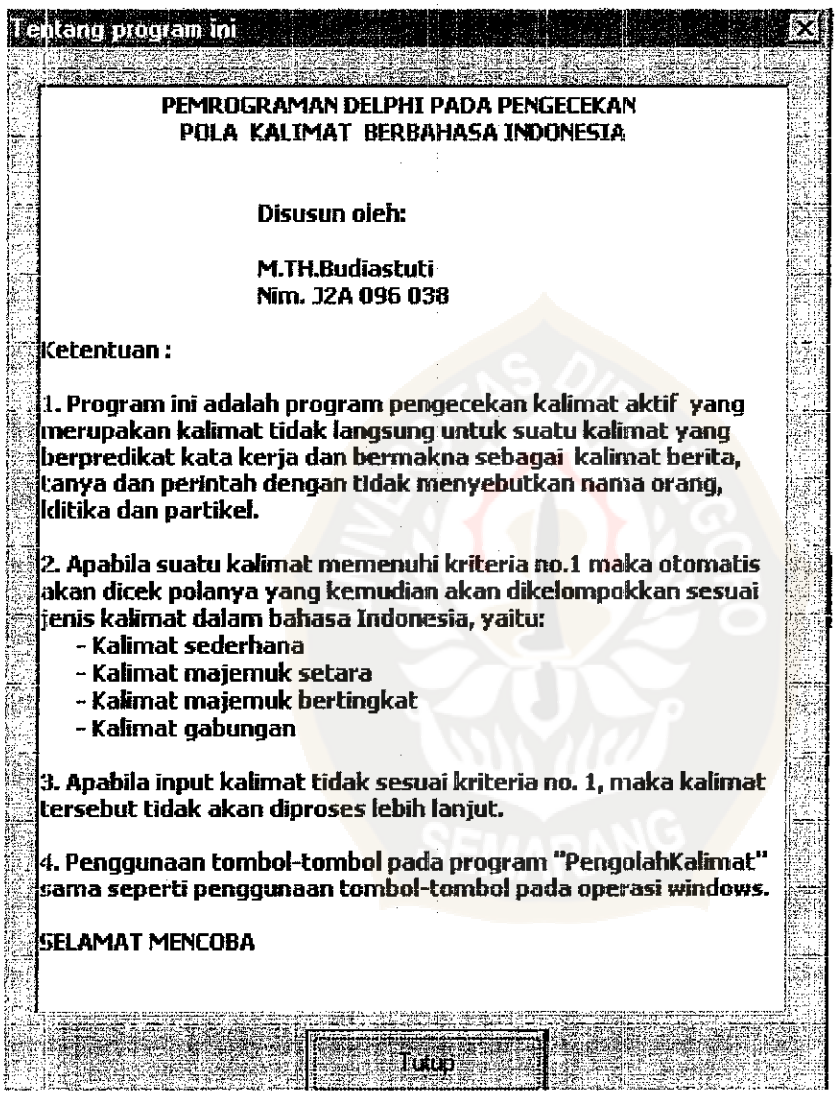
```
procedure TEditKamusForm.BacaFile(fn : string);
var s : string;
begin
     AssignFile(ft, fn);
     Reset(ft);
     while not EOF(ft) do
     begin
          Readln(ft,s);
          EditMemo.Lines.Add(s);
     end;
     CloseFile(ft);
end;

procedure TEditKamusForm.SimpanFile(fn : string);
var i : integer;
begin
     AssignFile(ft, fn);
     Rewrite(ft);
     for i := 0 to EditMemo.Lines.Count-1 do
          Writeln(ft,EditMemo.Lines[i]);
     EditMemo.Modified := false;
     CloseFile(ft);
end;

procedure TEditKamusForm.ClearGrid;
var i, j : integer;
begin
     for i := 0 to 1 do
          for j := 1 to KamusStringGrid.RowCount-1 do
               KamusStringGrid.Cells[i,j] := '';
end;

procedure TEditKamusForm.FormShow(Sender: TObject);
var i : integer;
begin
     lastfile := '';
     ClearMemo;
     ClearGrid;
     KamusStringGrid.Cells[0,0] := 'Nama file';
     KamusStringGrid.Cells[1,0] := 'Jenis';

     for i := 0 to Length(KamusRec)-1 do
     begin
          KamusStringGrid.Cells[0,i+1] := KamusRec[i].filename;
          KamusStringGrid.Cells[1,i+1] := KamusRec[i].code;
     end;
end;
```

```
procedure TEditKamusForm.KamusStringGridDblClick(Sender: TObject);
begin
     if KamusStringGrid.Cells[KamusStringGrid.Col,
KamusStringGrid.Row]<>'' then
     begin
          if (lastfile<>'')and(EditMemo.Modified) then
               if MessageDlg('File sudah diubah, apakah akan
               disimpan?',mtConfirmation,[mbYes, mbNo],0) = mrYes
               then
                              SimpanFile(lastfile);

          ClearMemo;
          BacaFile(KamusStringGrid.Cells[0, KamusStringGrid.Row]);

          lastfile :=
           KamusStringGrid.Cells[0,KamusStringGrid.Row];
          EditMemo.Modified := false;
          KamusLabel.Caption := UpperCase(lastfile);
     end;
end;

procedure TEditKamusForm.SaveButtonClick(Sender: TObject);
begin
     if lastfile <> '' then SimpanFile(lastfile);
end;

procedure TEditKamusForm.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
var r : integer;
begin
     if (lastfile<>'')and(EditMemo.Modified) then
     begin
          r := MessageDlg('File sudah diubah, apakah akan
           disimpan?',mtConfirmation,[mbYes, mbNo, mbCancel],0);
          if r = mrYes then SimpanFile(lastfile)
          else if r = mrCancel then CanClose := false;
     end;
end;

procedure TEditKamusForm.SortBtnClick(Sender: TObject);
var sortst : TStringList;
    i : integer;
begin
     sortst := TStringList.Create;
     sortst.Duplicates := dupIgnore;
     sortst.Sorted := true;
     for i := 0 to EditMemo.Lines.Count-1 do
      sortst.Add(UpperCase(EditMemo.Lines[i]));
     EditMemo.Lines.Clear;
     for i := 0 to sortst.Count-1 do
      EditMemo.Lines.Add(sortst.Strings[i]);
     sortst.Free;
end;

end.
```

```
unit AboutUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
   Dialogs,StdCtrls, ComCtrls;

type
  TAboutForm = class(TForm)
    TextRichEdit: TRichEdit;
    CloseButton: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutForm: TAboutForm;

implementation

{$R *.DFM}

procedure TAboutForm.FormCreate(Sender: TObject);
begin
     TextRichEdit.Lines.LoadFromFile('pesan.txt');
end;

procedure TAboutForm.FormShow(Sender: TObject);
begin
     CloseButton.SetFocus;
end;

end.
```

OUT PUT

TAMPILAN MENU INFORMASI



TAMPILAN PENGECEKAN POLA KALIMAT