Halaman Lampiran

```
{*-----------------------------------------*
 *   Judul Program   : Pohon AVL           *
 *   Disusun oleh    : Sunarno             *
 *   N I M           : J101 94 1046        *
 *   Kepentingan     : Tugas Akhir         *
 *   Dibuat Tanggal  : 1 Oktober 1999      *
 *---------------------------------------*}

Program MEMBANGUN_POHON_AVL;

uses crt;
const lima  = 5;
type Str5   = string[lima];
     simpul = ^node;
     node   = record
              kunci : integer;
              count : integer;
              kiri,kanan : simpul;
              kode  : -1..1;
              end;

var n,k    : integer;
   akar    : simpul;
   test    : boolean;
   lagi    : char;
   pilihan : byte;


Function Ada_pohon(n:integer):simpul;

var newnode : simpul;
   x, nl, nr : integer;
    begin
    if n = 0 then Ada_pohon := nil
    else
        begin
         nl := n div 2;
         nr := n -nl-1;
         read(x);new(newnode);
         with newnode^ do
            begin
             kunci := x;
             kiri := Ada_pohon(nl);
             kanan :=Ada_pohon(nr);
             kode := 0;
            end;
         Ada_pohon := newnode;
        end;
    end;


Procedure Pencarian_pohon ( var Sb : simpul;
                            x : integer);

begin
  if Sb = nil then
   begin
```

```
      new(Sb);
      with Sb^ do
       begin
        kunci := x;
        count := 1;
        kiri  := nil;
        kanan := nil;
       end;
    end
   else if (x < Sb^.kunci) then Pencarian_pohon(Sb^.kiri,x)
   else if (x > Sb^.kunci) then Pencarian_pohon(Sb^.kanan,x)
   else Sb^.count := Sb^.count +1;
end;

Procedure SISIPKAN(var Sb : simpul; var cek : boolean;
                       Ct : integer);
var Sb1,Sb2 : simpul;
  begin
   if Sb = nil then
    { Pohon masih kosong, sisipkan}
    begin
      new(Sb); Cek := true;
      with Sb^ do
          begin
            kunci:= Ct;
            Kiri := nil;
            Kanan := nil;
            Kode := 0;
          end;
    end
   else
     if Ct < Sb^.kunci then
      begin
        SISIPKAN(Sb^.Kiri,Cek,Ct);
        if Cek then
        {* cabang kiri lebih tinggi *}
        case Sb^.kode of
          1  : begin {*pohon menjadi seimbang*}
               Sb^.Kode := 0;
               Cek := false;
               end;
          0  : Sb^.Kode:= -1;  {*Pohon berat kekiri*}

          -1 : begin {diperlukan penyeimbangan*}
               Sb1 := Sb^.Kiri;
               if Sb1^.Kode = -1 then
                 {*pemutaran tunggal ke kanan*}
                 begin
                  Sb^.Kiri := Sb1^.Kanan;
                  Sb1^.Kanan := Sb;
                  Sb^.Kode := 0;
                  Sb := Sb1
                 end
               else  {*Pemutaran ganda kanan kiri*}
                 begin
                  Sb2 := Sb1^.Kanan;
                  Sb1^.Kanan := Sb2^.Kiri;
```

```
                    Sb2^.Kiri := Sb1;
                    Sb^.Kiri := Sb2^.Kanan;
                    Sb2^.Kanan := Sb;
                  {*Menentukan kode pohon setelah
                    diseimbangkan}
                    if Sb2^.Kode = -1 then
                       Sb^.Kode := +1
                    else
                       Sb^.Kode := 0;
                        if Sb2^.Kode = +1 then
                           Sb1^.Kode := -1
                        else
                           Sb1^.Kode := 0;
                           Sb := Sb2;
                  end;
              Sb^.Kode := 0; Cek :=false;
            end;
         end;
    end
else  {*Mencari dicabang kanan *}
 if Ct > Sb^.kunci then
   begin
    SISIPKAN(Sb^.Kanan,Cek,Ct);
     if Cek then  {*cabang kanan lebih tinggi*}
      case Sb^.Kode of
           -1 : begin  {* Pohon menjadi seimbang*}
                Sb^.Kode := 0 ;
                Cek := false;
                end;
            0 : {*pohon menjadi berat kekanan*}
                Sb^.Kode := +1;
            1 : begin  {* perlu penyeimbangan*}
                Sb1 := Sb^.Kanan;
                if Sb1^.Kode = +1 then
                {* pemutaran kekiri*}
                  begin
                   Sb^.Kanan  := Sb1^.Kiri;
                   Sb1^.Kiri  := Sb;
                   Sb^.Kode   := 0;
                   Sb         := Sb1;
                  end
                else
               {* pemutaran ganda kiri_kanan *}
                   begin
                    Sb2  := Sb1^.Kiri;
                    Sb1^.Kiri := Sb2^.Kanan;
                    Sb2^.Kanan := Sb1;
                    Sb^.Kanan := Sb2^.Kiri;
                    Sb2^.Kiri   := Sb;
                    if Sb2^.Kode = +1 then
                       Sb^.Kode := -1
                    else
                       Sb1^.Kode := 0;
                    if Sb2^.Kode = -1 then
                       Sb^.Kode := +1
                    else
                       Sb1^.Kode := 0;
```

```
                    Sb  := Sb2
                  end;
                Sb^.Kode := 0; cek := false
              end;
        end;
    end
  else  {*Tidak melakukan penyisipan*}
    cek := false;
end;

  Procedure Hapus_SIMPUL(var Sb : simpul;
                         var Cek : boolean;
                             Ct : integer);
   var Bantu : simpul;

      {************************************
       * Prosedur ini dikerjakan jika     *
       * Cabang kanan menjadi lebih tinggi*
       ************************************}
  procedure SEIMBANGKAN1(var Sb : simpul;
                             var Cek : boolean);

  var Sb1, Sb2 : simpul;
      Kd1, Kd2 : -1..+1;

  begin
   case Sb^.Kode of
       -1 : {* Pohon menjadi seimbang*}
              Sb^.Kode := 0;
        0  : begin
             {* Pohon masih condong ke kanan*}
             Sb^.Kode := +1;
             Cek  := false;
             end;
        1  :  begin
             {* perlu diseimbangkan *}
              Sb1 := Sb^.Kanan;
              Kd1 := Sb1^.Kode;
              if Kd1 >= 0 then
              { Pemutaran tunggal ke kiri *}
               begin
                Sb^. Kanan := Sb1^.Kiri;
                Sb1^.Kiri := Sb;
                 if Kd1 = 0 then
                   begin
                    Sb^.Kode := +1;
                    Sb1^.Kode := -1;
                    Cek := false
                   end
                 else
                   begin
                    Sb^.Kode := 0;
                    Sb1^.Kode := 0;
                   end;
                 Sb := Sb1
                 end
               else
```

```
            {* Pemutaran ganda kiri_kanan *}
            begin
             Sb2 := Sb1^.Kiri;
             Kd2 := Sb2^.Kode;
             Sb1^.Kiri := Sb2^.Kanan;
             Sb2^.Kanan := Sb1;
             Sb^.Kanan := Sb2^.Kiri;
             Sb2^.Kiri := Sb;
             if Kd2 = +1 then
              Sb^.Kode := -1
              else
               Sb^.Kode := 0;
               if Kd2 = -1 then
               Sb1^.Kode := +1
               else
               Sb1^.Kode := 0;
               Sb := Sb2;
               Sb2^.Kode := 0
               end
            end
          end;
          {* Case prosedur SEIMBANGKAN1 ****}
      { ************************************
       *   prosedur ini dikerjakan jika    *
       * cabang kiri menjadi lebih tingggi *
       ************************************}
Procedure SEIMBANGKAN2(var  Sb: simpul;
                       var Cek : boolean);
 var Sb1, Sb2 :simpul;
     Kd1, Kd2 : -1..+1;

 begin
 Case Sb^.Kode of
   1 : {* Pohon menjadi seimbang*}
            Sb^.Kode := 0;

   0 :  begin
           Sb^.Kode := -1;
           Cek := false;
        end;
  -1 : begin
           {* perlu penyeimabangan kembali *}
           Sb1 := Sb^.Kiri;
           Kd1  := Sb1^.Kode;

        if Kd1 <= 0 then
           {* Pemutaran tunggal ke kanan *}
         begin
              Sb^.Kiri := Sb1^.Kanan ;
              Sb1^.Kanan := Sb;
              if Kd1 = 0 then
               begin
                Sb^.Kode := -1;
                Sb^.Kode := +1;
                Cek := false
               end
```

```
            else
             begin
              Sb^.Kode := 0;
              Sb1^.Kode := 0;
             end;
            Sb := Sb1
           end
          else
           {* Pemutaran ganda Kanan_kiri *}
           begin
              Sb2 := Sb1^.Kanan;
              Kd2 := Sb2^.Kode;
              Sb1^.Kanan := Sb2^.Kiri;
              Sb2^.Kiri := Sb1;
              Sb^.Kiri := Sb2^.Kanan;
              Sb2^.Kanan := Sb;
            if Kd2 = -1 then
              Sb^.Kode  := +1
            else
              Sb^.Kode  := 0;
             if Kd2 = +1 then
              Sb1^.Kode := -1
             else
              Sb1^.Kode := 0;
              Sb  := Sb2;
              Sb2^.Kode := 0
           end
        end
      end
end;
      {Case prosedure SEIMBANGKAN2  *}

 {*---------------------------*
  *  Prosedure untuk menghapus*
  *---------------------------*}
Procedure HAPUS( var Rd : simpul; var Cek :boolean);
 begin
      if Rd^.Kanan <> nil then
      {cabang kanan tidak kosong*}
       begin
        HAPUS(Rd^.Kanan, Cek);
        if Cek then SEIMBANGKAN2 (Rd, Cek)
       end
      else
        begin
          Bantu^.kunci := Rd^.kunci;
          Rd := Rd^.Kiri;
          Cek := true
        end
end;
       {*------- prosedur hapus----------------*}
    {*********************************************
     *  bagian Utama dari prosedure HAPUS_SIMPUL*
     *********************************************}
 begin
    if Sb = nil then
      begin
```

```
            Write('Bilangan Tersebut');
            write('  Tidak Ditemukan ');
            Cek  := false;
        end
      else
         if Ct < Sb^.Kunci then
          begin
           HAPUS_SIMPUL(Sb^.Kiri, Cek, Ct);
           if Cek then SEIMBANGKAN1( Sb, Cek)
          end
         else
            if Ct > Sb^.Kunci then
              begin
                HAPUS_SIMPUL(Sb^.Kanan, Cek, Ct);
                if Cek then SEIMBANGKAN2(Sb, Cek)
                end
            else
             begin
                Bantu := Sb;
                If Bantu^.Kanan = nil then
                 begin
                  Sb := Bantu^.Kiri;
                  Cek := true
                 end
                else
                 if Bantu^.Kiri = nil then
                   begin
                    Sb  := Bantu^.Kanan;
                    Cek  := true
                   end
                 else
                  begin
                   HAPUS(Bantu^.Kiri, Cek);
                   if Cek then
                     SEIMBANGKAN1(Sb, Cek)
                  end;
             end
  end;{* Prosedure HAPUS_SIMPUL *}

Procedure Cetak_Pohon(w : simpul; h :integer);

var i :integer;

begin
 if w <> nil then
    with w^ do
     begin
       Cetak_pohon (kiri, h+1);
       for i := 1 to h do
           Write('    ');
           writeln(kunci);
           Cetak_pohon(kanan, h+1);
     end
end;
{**********************
 *  Daftar Menu Pilihan*
```

```
 ***********************}

begin
   clrscr;
   writeln(' Program ini di baca dengan cara ');
   writeln(' bagian bawah sebagai cabang kanan');
   writeln(' bagian atas sebagai cabang kiri ');
   writeln;writeln;
   lagi := 'Y';
   while  (upcase (lagi)  = 'Y') do
   begin
   clrscr;
   gotoxy(20,3);  writeln('ÕÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¸');
   gotoxy(20,4);  writeln('³         Pohon Avl         ³');
   gotoxy(20,5);  writeln('ÆÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍµ');
   gotoxy(20,6);  writeln('³                           ³');
   gotoxy(20,7);  writeln('³<<< Daftar menu Pilihan >>>³');
   gotoxy(20,8);  writeln('ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´');
   gotoxy(20,9);  writeln('³                           ³');
   gotoxy(20,10); writeln('³  1. Membuat Pohon         ³');
   gotoxy(20,11); writeln('³                           ³ ');
   gotoxy(20,12); writeln('³  2. Menyisipkan Simpul    ³ ');
   gotoxy(20,13); writeln('³                           ³ ');
   gotoxy(20,14); writeln('³  3. Menghapus Simpul      ³ ');
   gotoxy(20,15); writeln('³                           ³ ');
   gotoxy(20,16); writeln('³  0. Selesai               ³ ');
   gotoxy(20,17); writeln('ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ ');
   writeln;
   pilihan :=9;
   while (pilihan < 0 ) or (pilihan > 3 ) do
   begin
     gotoxy(20,20);
     write(' Pilih (0-3)? ');
     read(pilihan);
   end;
{* Program Utama*}
   clrscr;
   if pilihan = 1 then
      {*Membentuk Pohon*}
        begin
          write('Banyaknya data masukkan :');
          readln(n);
          writeln;
          writeln('Silahkan Anda Masukan data :');
          akar := Ada_pohon(n);
          writeln('Inilah hasil yang diperoleh:');
          Cetak_pohon(akar,0);
          readln;
          write('TEKAN TOMBOL SEMBARANG UNTUK KEMBALI');
          writeln( ' KE MENU PILIHAN  ');
          writeln;
          repeat until keypressed;
        end;
     if pilihan = 2 then
        begin
          clrscr;
          lagi := 'Y';
```

```
      while (upcase (lagi) = 'Y') do
        begin

         write('Mau menyisipkan :');
         readln(k);
         sisipkan(akar,test,k); {Pencarian_Pohon(akar,k);}
         writeln;
         cetak_pohon(akar,0);
         writeln;
         write('Mau menyisipkan lagi(Y/T)?');
         readln(Lagi);
        end;
     end;
  if pilihan = 3 then
     begin
     clrscr;
     lagi := 'Y';
     while (upcase (lagi) = 'Y') do
        begin
        write('Mau Menghapus Simpul:');
        readln(k);
        HAPUS_SIMPUL(akar, test,k);{Pencarian_pohon(akar,k);}
        writeln;
        cetak_pohon(akar,0);
        writeln;
        write('Mau Menghapus lagi(Y/T)?');
        readln(lagi);
        end;
     end;

  if pilihan = 0 then
        begin
         gotoxy(40,30);
         writeln(' Sudah Selesai ');
         writeln;
         end;
         gotoxy(40,35);
         write('Kembali ke menu lagi(Y/T)?');
         readln(lagi);

    end;
end.
```