

```

{ Program Pointer Pertama }
uses crt;
type ptr = ^mhs;
   mhs = record
       nim : string;
       nama : string;
       nilai: byte;
       next : ptr;
   end;

var
   m : ptr;
begin
   clrscr;
   new (m);
   write (' NIM      : '); readln (m^.nim);
   write (' NAMA      : '); readln (m^.nama);
   write (' NILAI    : '); readln (m^.nilai);
   m^.next := nil;
   writeln (m^.nim);
   writeln (m^.nama);
   writeln (m^.nilai);
   readln;
end.
=====

{ Program Pointer Single Linklist }
uses crt;
type simpul = ^data;
   data = record
       isi : string;
       next : simpul;
   end;

var
   elemen : string;
   awal,akhir,baru,posisi,hapus,bantu : simpul;

procedure tambah_belakang (elemen : string);
begin
   new(baru);
   baru^.isi := elemen;
   if awal = nil then awal := baru
   else akhir^.next := baru;
   akhir := baru;
   akhir^.next := nil;
end;

procedure tambah_depan (elemen : string);
begin
   new(baru);
   baru^.isi := elemen;
   if awal = nil then akhir := baru
   else baru^.next := awal;
   awal := baru;
   akhir^.next := nil;
end;

procedure hapus_simpul (elemen : string);
begin
   if awal = nil then writeln ('List Kosong')
   else if awal^.isi = elemen then
   begin
       hapus := awal;
       awal := hapus^.next;
       dispose(hapus);
   end;
end;

```

```

end
else
begin
    bantu := awal ; posisi := awal;
    while (elemen <> posisi^.isi) and (posisi^.next <> nil) do
    begin
        bantu := posisi;
        posisi := posisi^.next;
    end;
    hapus := posisi;
    if (hapus^.next <> nil) and (posisi^.isi = elemen) then
    begin
        posisi := hapus^.next;
        bantu^.next := posisi;
        dispose(hapus);
    end
    else if hapus^.isi = elemen then
    begin
        akhir := bantu;
        akhir^.next := nil;
        dispose(hapus);
    end
    else
        writeln ('Simpul Yang Akan Dihapus Tidak Ada');
    end;
end;

begin
    clrscr;
    tambah_belakang ('A');
    tambah_belakang ('F');
    tambah_belakang ('B');
    tambah_belakang ('C');
    tambah_depan ('S');
    tambah_depan ('K');
    hapus_simpul ('C');
    posisi := awal;
    repeat
    write (posisi^.isi);
    posisi := posisi^.next;
    until posisi = nil;
    readln;
end.

```

```

=====

{ Program Pointer Double Linklist }
uses crt;
type ptr = ^data;
    data = record
        isi : string;
        pre,
        next : ptr;
    end;
var baru,posisi,awal,akhir,bantu,hapus : ptr;

procedure tambah_belakang (elemen : string);
begin
    new (baru);
    baru^.isi := elemen;
    baru^.pre := nil;
    baru^.next := nil;
    if awal = nil then awal:=baru
    else
    begin

```

```

        akhir^.next := baru;
baru^.pre      := akhir;
end;
akhir := baru;
akhir^.next := nil;
end;

procedure tambah_depan (elemen : string);
begin
    new (baru);
    baru^.isi      := elemen;
    baru^.pre      := nil;
    baru^.next     := nil;
    if awal = nil then awal:=baru
    else
    begin
        awal^.pre := baru;
        baru^.next := awal;
    end;
    awal := baru;
    awal^.pre := nil;
end;

procedure tambah_tengah (elemen : string; n : byte);
var panjang, i : byte;
begin
    new (baru);
    baru^.isi      := elemen;
    baru^.pre      := nil;
    baru^.next     := nil;
    if awal <> nil then
    begin
        posisi := awal;
        panjang:= 0;
        repeat
            posisi := posisi^.next;
            inc(panjang);
        until posisi = nil;
        if (n>1) and (n<=panjang) then
        begin
            posisi := awal;
            for i:= 1 to n-2 do
                posisi := posisi^.next;
            bantu      := posisi^.next;
            posisi^.next := baru;
            baru^.pre   := posisi;
            baru^.next  := bantu;
            bantu^.pre  := baru;
        end
        else
            writeln ('Posisi Pointer Tidak Ada Di Tengah');
        end
    else
        writeln ('Pointer Belum Ada');
    end;
end;

procedure hapus_simpul (elemen : string);
begin
    { Hapus Pointer Di Awal }
    if awal^.isi = elemen then
    begin
        hapus      := awal;
        awal       := awal^.next;
        awal^.pre:= nil;
    end;
end;

```

```

        dispose (hapus);
    end
    else
    begin
        { Hapus Pointer Di Tengah }
        posisi := awal;
        while (posisi^.isi <> elemen) and (posisi^.next <> nil) do
            posisi := posisi^.next;
            if posisi^.next <> nil then
            begin
                bantu := posisi^.pre;
                hapus := posisi;
                posisi := posisi^.next;
                bantu^.next := posisi;
            posisi^.pre := bantu;
                dispose(bantu);
            end
            { Hapus Pointer Di Akhir }
            else if posisi^.isi = elemen then
            begin
                hapus := posisi;
                akhir := posisi^.pre;
                akhir^.next:= nil;
                dispose(hapus);
            end
            else
                writeln ('Pointer Tidak Ditemukan');
        end;
    end;

procedure baca_depan;
begin
    posisi := awal;
    repeat
        write (posisi^.isi);
        posisi := posisi^.next;
    until posisi = nil;
end;

procedure baca_belakang;
begin
    posisi := akhir;
    repeat
        write (posisi^.isi);
        posisi := posisi^.pre;
    until posisi = nil;
end;

begin
    clrscr;
    tambah_belakang ('A');
    tambah_belakang ('B');
    tambah_belakang ('C');
    tambah_belakang ('D');
    tambah_depan ('Z');
    tambah_depan ('X');
    tambah_tengah ('P',3);
    hapus_simpul ('A');
    baca_depan;
    writeln;
    baca_belakang;
    readln;
end.

```

```
{ Hasil Eksekusi :  
  XZPBCD  
  DCBPZX }  
=====
```

```
{ Program Queue }
```

```
uses crt;  
const max = 100;  
type antri = array[1..max] of char;  
var  
  q      : antri;  
  depan,belakang : integer;  
  
procedure tambah (x : char);  
begin  
  if belakang = max then  
    writeln ('Antrian Penuh')  
  else  
    begin  
      belakang := belakang + 1;  
      q[belakang] := x;  
    end;  
end;  
  
{ procedure untuk menghapus dari belakang }  
procedure hapus_geser_belakang;  
var  
  i : byte;  
begin  
  if belakang < depan then  
    writeln ('Antrian Kosong')  
  else  
    begin  
      for i:= belakang-3 downto 1 do  
        q[i] := q[i-1];  
      belakang:= belakang+1;  
    end;  
end;  
  
{ procedure untuk menghapus dari depan }  
procedure hapus_geser_depan;  
var  
  i : byte;  
begin  
  if belakang < depan then  
    writeln ('Antrian Kosong')  
  else  
    begin  
      for i:= 1 to belakang-1 do  
        q[i] := q[i+1];  
      belakang := belakang-1;  
    end;  
end;  
procedure baca;  
var  
  i : byte;  
begin  
  for i:= depan to belakang do  
    write (q[i]);  
end;  
  
procedure info;  
begin  
  writeln;
```

```

        writeln ('Depan      = ', depan);
        writeln ('Belakang = ', belakang);
end;

```

```

begin
    clrscr;
    depan      := 1;
    belakang:= 0;
    tambah ('A');
    tambah ('B');
    tambah ('C');
    tambah ('D');
    tambah ('E');
    hapus_geser_belakang;
    hapus_geser_belakang;
    hapus_geser_belakang;
    hapus_geser_belakang;
    baca;
    info;
    readln;
end.

```

```

=====
{ Program Queue Tambah Putar }

```

```

uses crt;
const max = 100;
type antri = array[1..max] of char;
var
    q      : antri;
    depan,belakang : integer;

procedure tambah_putar (x : char);
begin
    if (belakang = max) and (depan = 1) or
        ((belakang + 1) = depan) then
        writeln ('Antrian Penuh')
    else
    begin
        if belakang = max then
            belakang := 1
        else
            belakang := belakang + 1;
            q[belakang] := x;
        end;
        if depan = 0 then depan := 1;
    end;
end;

procedure hapus_putar;
begin
    if depan = belakang then
    begin
        depan      := 0;
        belakang := 0;
        writeln ('Antrian Penuh');
    end
    else
    begin
        if depan = max then
            depan := 1
        else
            depan := depan +1;
        end;
    end;
end;

```

```

procedure baca;
var
  i : byte;
begin
  for i:= depan to belakang do
    write (q[i]);
end;

procedure info;
begin
  writeln;
  writeln ('Depan      = ', depan);
  writeln ('Belakang = ', belakang);
end;

begin
  clrscr;
  depan := 0;
  belakang:= 0;
  tambah_putar ('A');
  tambah_putar ('B');
  tambah_putar ('C');
  tambah_putar ('D');
  tambah_putar ('E');
  hapus_putar;
  hapus_putar;
  hapus_putar;
  hapus_putar;
  tambah_putar ('H');
  tambah_putar ('K');
  baca;
  info;
  readln;
end.

```

```

{ Hasil Eksekusi :
  EHK }

```

```

{ Program Stack / Tumpukan (FIFO) }

```

```

uses crt;
const max = 10;
type tumpukan = array [1..max] of char;
var
  t      : tumpukan;
  atas  : byte;

procedure push (x : char);
begin
  if atas = max then
    writeln ('Tumpukan Penuh')
  else
    begin
      atas := atas + 1;
      t[atas] := x;
    end;
end;

procedure pop;
begin
  if atas = 0 then
    writeln ('Tumpukan Kosong')
  else
    atas := atas - 1;
end;

```

```

end;

procedure baca;
var
  i : byte;
begin
  writeln ('Atas = ', atas);
  for i:= 1 to atas do
  begin
    gotoxy(2,atas+3-i);
    write (t[i]);
  end;
end;

begin
  clrscr;
  push ('A');
  push ('B');
  push ('C');
  push ('D');
  push ('E');
  push ('F');
  pop;
  pop;
  baca;
  readln;
end.

```

{ Program Searching1 }

```

Program Searching;
uses crt;
type larik = array [1..1000] of integer;
var a      : larik;
    posisi : larik;
    n,m,x   : integer;
    ada     : boolean;

procedure cari_vektor (var a : larik);
var i, data : integer;
begin
  write (' Data Yang Dicari : '); readln (data);
  ada := false;
  x   := 0;
  for i:= 1 to n do
    if a[i] = data then
      begin
        inc (x);
        posisi [x] := i;
        ada := true;
      end;
    if not ada then
      begin
        inc (n);
        a [n] := data;
      end;
  end;
end;

procedure isi;
var i : integer;
begin
  clrscr;
  write (' Jumlah Data : '); readln (m);
  for i:= 1 to m do

```

```

begin
    write (' A[' ,i, ' ] : '); readln (a[i]);
end;
n := m;
end;

procedure info;
var i : integer;
begin
    writeln (' Data Awal ');
    for i:= 1 to m do
        write (' ',a[i], ' ');
    writeln;
    if ada then
        begin
            write (' Data Ditemukan Pada Posisi : ');
            for i:= 1 to x do
                write (posisi[i], ' ');
            end
        else
            begin
                writeln (' Data Tidak Ditemukan ');
                writeln (' Data Akhir ');
                for i:= 1 to n do
                    write (' ',a[i], ' ');
                end;
            end;
end;

begin
    clrscr;
    isi;
    cari_vektor (a);
    info;
    readln;
end.

```

=====

```

{ Program Searching2 }
Program Searching2;
uses crt;
type larik = array [1..1000] of integer;
var a      : larik;
    posisi : integer;
    n,m,x,j : integer;
    ada     : boolean;

procedure cari_vektor_urut_naik (var a : larik);
var i, data : integer;
begin
    write (' Data Yang Dicari : '); readln (data);
    ada := false;
    i   := 1;
    j   := 0;
    repeat
        if a[i] = data then
            begin
                posisi := i;
                ada     := true;
            end
        else if a[i] > data then
            begin
                j := i;
                i := n;
            end;
    end;

```

```

        inc (i);
until (i > n) or ada;
if not ada then
begin
    if j=0 then
        a[n+1] := data
    else
        begin
            for i:= n downto j do
                a[i+1] := a[i];
            a[j] := data;
        end;
        inc (n);
    end;
end;

end;

procedure isi;
var i : integer;
begin
    clrscr;
    write (' Jumlah Data : '); readln (m);
    for i:= 1 to m do
        begin
            write (' A[' ,i, ' ] : '); readln (a[i]);
        end;
        n := m;
    end;

procedure info;
var i : integer;
begin
    if ada then
        write (' Data Ditemukan ')
    else
        begin
            write (' Data Tidak Ditemukan ');
            for i:= 1 to n do
                write (a[i], ' ');
            end;
        end;
    end;

begin
    clrscr;
    isi;
    cari_vektor_urut_naik (a);
    info;
    readln;
end.

```

=====

{ Program Searching3 }

```

Program Searching3;
uses crt;
type larik = array [1..1000] of integer;
var a      : larik;
    posisi,atas,bawah,tengah : integer;
    n,m    : integer;
    ada    : boolean;

procedure cari_biner (var a : larik);
var i, data : integer;
begin
    write (' Data Yang Dicari : '); readln (data);
    ada := false;

```

```

    atas := n;
    bawah := 1;
    while atas >= bawah do
    begin
        tengah := (atas+bawah) div 2;
        if data < a[tengah] then
            atas := tengah - 1
        else if data > a[tengah] then
            bawah := tengah + 1
        else
            begin
                ada := true;
                posisi := tengah;
                bawah := atas + 1;
            end;
        end;
        if ada then
            write (' Data Ditemukan Pada Posisi ', posisi)
        else
            write (' Data Tidak Ditemukan ');
    end;

```

```

procedure isi;
var i : integer;
begin
    clrscr;
    write (' Jumlah Data : '); readln (m);
    for i:= 1 to m do
    begin
        write (' A[' ,i, ' ] : '); readln (a[i]);
    end;
    n := m;
end;

```

```

begin
    clrscr;
    isi;
    cari_biner (a);
    readln;
end.

```

```

=====
{ Program Pohon <tree> Non_Rekursif }

```

```

Program Pohon_Non_Rekursif;
uses crt;
type tree = ^simpul;
    simpul = record
        info : char;
        kiri,kanan : tree;
    end;
var
    bantu,p,q,b : tree;

function baru (hrf:char):tree;
var
    b : tree;
begin
    new(b);
    b^.info := hrf;
    b^.kanan := nil;
    b^.kiri := nil;
    baru := b;
end;

```

```

procedure sisip_non_rekursif (var akar : tree; hrf : char);
begin
  b := baru(hrf);
  if akar = nil then akar := b
  else
    begin
      q := akar;
      p := akar;
      while (hrf<>p^.info) and (q<>nil) do
        begin
          p := q;
          if hrf < p^.info then
            q := p^.kiri
          else
            q := p^.kanan;
          end;
          if hrf = p^.info then write ('Data Sudah Ada')
          else
            if hrf < p^.info then p^.kiri := b
            else p^.kanan := b;
          end;
        end;
      end;
end;

begin
  clrscr;
  sisip_non_rekursif (bantu,'D');
  sisip_non_rekursif (bantu,'B');
  sisip_non_rekursif (bantu,'C');
  sisip_non_rekursif (bantu,'E');
  write (b^.info);
  readln;
end.

```

{ Program Pohon <tree> Rekursif }

```

Program Pohon_Rekursif;
uses crt;
type tree = ^simpul;
      simpul = record
        info : char;
        kiri,kanan : tree;
      end;
var
  pohon,bantu,p,q,b : tree;

function baru (hrf:char):tree;
var
  b : tree;
begin
  new(b);
  b^.info := hrf;
  b^.kanan := nil;
  b^.kiri := nil;
  baru := b;
end;

procedure sisip_rekursif (var pohon : tree; hrf : char);
begin
  if pohon = nil then pohon := baru(hrf)
  else
    if pohon^.info > hrf then
      sisip_rekursif (pohon^.kiri,hrf)
    else if pohon^.info < hrf then
      sisip_rekursif (pohon^.kanan,hrf)

```

```
        else
            write ('karakter ',hrf,' sudah ada');
end;

begin
    clrscr;
    sisip_rekursif (bantu,'D');
    sisip_rekursif (bantu,'B');
    sisip_rekursif (bantu,'C');
    sisip_rekursif (bantu,'A');
    sisip_rekursif (bantu,'F');
    sisip_rekursif (bantu,'E');
    write (b^.info);
    readln;
end.
```

```
=====
Eksekusi Searching 1 :
Jumlah Data : 5
A[1] : 5
A[2] : 6
A[3] : 8
A[4] : 9
A[5] : 10
Data Yang Dicari : 1
Data Awal
5 6 8 9 10
Data Tidak Ditemukan
Data Akhir
5 6 8 9 10 1
=====
```

```
Eksekusi Searching 2 :
Jumlah Data : 5
A[1] : 1
A[2] : 3
A[3] : 5
A[4] : 7
A[5] : 9
Data Yang Dicari : 8
Data Tidak Ditemukan 1 3 5 7 8 9
=====
```

```
Eksekusi Searching 3 :
Jumlah Data : 5
A[1] : 1
A[2] : 4
A[3] : 7
A[4] : 9
A[5] : 10
Data Yang Dicari : 11
Data Tidak Ditemukan
=====
```