==========

==========

KUMPULAN PASCAL - PRAKTIKUM STRUKTUR DATA

==========

==========

- **Array 2D**

```
program array_multi;
uses crt;

var
  arrnilai : array[1..2, 1..3] of integer;
  baris,kolom : integer;
begin
clrscr;
     arrnilai[1, 1]:= 2;
     arrnilai[1, 2]:= 13;
     arrnilai[1, 3]:= 4;

     arrnilai[2, 1]:= 7;
     arrnilai[2, 2]:= 4;
     arrnilai[2, 3]:= 2;

for baris := 1 to 2 do
   begin
      for kolom := 1 to 3 do
      begin
       write(arrnilai[baris, kolom], ' ');

         if kolom = 3 then writeln;
   end;
  end;
readln;
end.
```

- **Array 3D**

```
program array_multi_3D;
uses crt;

const max = 3;
type arr3d = array [1..max, 1..max,
1..max] of integer;

var
  myarr : arr3d;
  baris,kolom,dimensi : integer;
begin
clrscr;
     myarr[1, 1, 1]:= 2;
     myarr[2, 1, 1]:= 2;
     myarr[3, 1, 1]:= 4;

     myarr[1, 2, 1]:= 3;
     myarr[2, 2, 1]:= 5;
     myarr[3, 2, 1]:= 9;

     myarr[1, 3, 1]:= 8;
     myarr[2, 3, 1]:= 7;
     myarr[3, 3, 1]:= 2;

{dimensi2}
     myarr[1, 1, 2]:= 10;
     myarr[2, 1, 2]:= 6;
     myarr[3, 1, 2]:= 3;

     myarr[1, 2, 2]:= 5;
     myarr[2, 2, 2]:= 7;
     myarr[3, 2, 2]:= 6;
     myarr[1, 3, 2]:= 2;
     myarr[2, 3, 2]:= 8;
     myarr[3, 3, 2]:= 2;

{dimensi3}
     myarr[1, 1, 3]:= 7;
     myarr[2, 1, 3]:= 7;
     myarr[3, 1, 3]:= 2;

     myarr[1, 2, 3]:= 2;
     myarr[2, 2, 3]:= 9;
```

```
    myarr[3, 2, 3]:= 3;

    myarr[1, 3, 3]:= 1;
    myarr[2, 3, 3]:= 4;
    myarr[3, 3, 3]:= 7;

    for dimensi := 1 to max do
    begin
       for baris := 1 to max do
       begin
          for kolom := 1 to max do
          begin
             write(myarr[baris, kolom,
dimensi], ' ');
              if kolom = max then
writeln('');
          end;
             if baris = max then
writeln('');
       end;
    end;
readln;
end.

for baris := 1 to 2 do
  begin
     for kolom := 1 to 3 do
     begin
      write(arrnilai[baris, kolom], ' ');

       if kolom = 3 then writeln;
  end;
 end;
readln;
end.
```

- **Bubble Sort**

```
Program Sorting_Bubble;
Uses Crt;
Const Max = 5;

Type Arr = Array[1..max] Of Byte;

Var
Data : Arr;
i : Byte;

Procedure Input;
Begin
    Clrscr;
    Writeln('Masukkan 5 Data ');

Writeln('=======================
=============');

    For I:=1 To Max Do
    Begin
        Write('Data Ke-',I,' :
');Readln(Data[i]);
    End;

    Clrscr;
    Write('Data Yang telah Diinput :
');

    For i:=1 to Max Do
        Write(Data[i],' ');

    Writeln;
End;

Procedure Change (Var a,b :Byte);
Var c:Byte;
Begin
C:=a; a:=b; b:=c;
End;

Procedure Asc_Bubble;
Var P,Q : Byte;
Flag: Boolean;
```

```
Begin
     Flag:=False;
     P:=2;

     While (P<Max) And (Not Flag) Do
     Begin
          Flag:=True;
          For Q:=Max Downto P Do
               If Data[Q]<Data[Q-1]
Then
               Begin

Change(Data[Q],data[Q-1]);
                    Flag:=False;
               End;
          Inc(i);
     End;

     Write(' Ascending : ');
End;

Procedure Desc_Bubble;
Var
     P,Q : Byte;
     Flag: Boolean;

Begin
     Flag:=False;
     P:=2;
     While (P<Max) And (Not Flag) Do
     Begin
          Flag:=True;
          For Q:=Max Downto P Do
               If Data[Q]>Data[Q-1]
Then
               Begin

Change(Data[Q],data[Q-1]);
                    Flag:=False;
               End;
               Inc(i);
          End;
     Write('Descending : ');

End;
```

```
Procedure Output;
Begin
     For I:=1 To Max Do
          Write(Data[I],' ');

     Writeln;
End;

Begin
     Input;
     Asc_Bubble;
     Output;
     Desc_Bubble;
     OutPut;
     Writeln;
     Write('Tekan Enter Untuk
Lanjut');
     Readln;
End.
```

- **dynamic_pointer**

```
Program dynamic_pointer;
Uses crt;
Type PntEmployee = ^recEmployee;
      recEmployee = record
      nama : string [20];
      divisi : string [20];
      gaji : longint;
 End;
 Var
 Pemp : PntEmployee;
 Begin
          new (Pemp); //alokasi
memory
          Pemp^.nama := ' sukirman ';
          Pemp^.divisi := ' sales ';
          Pemp^.gaji := 2000;
          writeln (Pemp^.nama ,' - ',
Pemp^.divisi,' - ', Pemp^.gaji);
          dispose (Pemp); //membuang
memory
          readln;
 end.
```

- **getmemory_pointer**

```
Program getmemory_pointer;
Uses crt;
Var p : pointer;
Begin
          getmem (p, 8192);//alokasi
memory sebanyak 8192 byte
          freemem(p,
8192);//membuang alokasi
end.
```

- **linked_list**

```
Program linked_list_lifo;
uses crt;
type
point=^recMhs;
recMhs=record
id_Mhs:String[4];
nama:String[8];
prodi:String[10];
next,prev:point;
end;
var head,tail,now:point;
jawab:char;
no:byte;

procedure Insert;
begin
new(now);
if head=nil then
begin
head:=now;
tail:=now;
head^.prev:=nil;
tail^.prev:=nil;
end

else
begin
tail^.next:=now;
now^.prev:=tail;
tail:=now;
tail^.next:=nil;
end;
end;

{Program utama}
begin
clrscr;
writeln (' DATA MAHASISWA LIFO ');
writeln (' ----------------------------------
-- ');
writeln;
repeat
INSERT;
writeln('Id Mahasiswa
=');readln(now^.id_Mhs);
write('Nama =');readln(now^.nama);
write('Prodi =');readln(now^.prodi);
writeln;
write('Apakah ingin nambah data
lagi?');readln(jawab);
writeln;
until upcase(jawab)='T';
clrscr;
writeln ;
writeln ('DATA MAHASISWA LIFO');
writeln;
no:=1;
now:=tail;
while now <>nil do
begin
gotoxy(1,3+no);write(now^.id_Mhs);
gotoxy(7,3+no);write(now^.nama);
gotoxy(22,3+no);write(now^.prodi);
writeln;
inc(no);//menambahkan 1
now:=now^.next;
end;
readln;
end.
```

- **memory_pointer**

```
Program memory_pointer;
Uses crt;
Var nilai : integer;
    ptr : ^integer;
    memory : ^word;
Begin
        nilai := 100;
        ptr := @nilai;//memanfaatkan
variable nilai
        Writeln (' Nilai Saat Ini = ' ,
ptr^);
        ptr^ := 200; //pointer
mengubah nilai
        writeln (' Nilai Sekarang = ',
ptr^);
        memory := addr(ptr); //ambil
ukuran pointer
        writeln (memory^);
        readln;
End.
```

- **Merge Sort**

```
program MergeSort;
uses crt;
type arr = array [1..100] of integer;

var
    ArrMain, ArrUrut : arr;
    n,m : integer;

function merge(Left:arr; pjgL:integer;
Right:arr; pjgR:integer):arr;
var
    i,j,k,m,panjang: integer;
    hasil : arr;
begin
    i:=1;
    j:=1;
k:=1;
panjang:=pjgL+pjgR;
while ((pjgL>0) and (pjgR>0)) do
begin
if(Left[i]<= Right[j]) then
begin
hasil[k]:=Left[i];
i:=i+1;
k:=k+1;
pjgL:=pjgL-1;
end
else
begin
hasil[k]:=Right[j];
j:=j+1;
k:=k+1;
pjgR:=pjgR-1;
end;
end;
while (pjgL>0) do
begin
hasil[k]:=Left[i];
i:=i+1;
k:=k+1;
pjgL:=pjgL-1;
end;
while (pjgR>0) do
```

```
begin
hasil[k]:=Right[j];
j:=j+1;
k:=k+1;
pjgR:=pjgR-1;
end;
merge:=hasil;
for m:= 1 to panjang do
writeln('Array Hasil ke-',m,' :
',hasil[m]);
end;

function mergesort(pjg:integer;A :
arr):arr;
var
middle,i,pjgLeft,pjgRight : integer;
ArrLeft,ArrRight,ArrHasil : arr;
begin
if pjg <= 1 then
mergesort := A
else
begin
middle := pjg div 2;
for i:=1 to middle do
ArrLeft[i]:=A[i];

for i:=(middle+1) to pjg do
ArrRight[i-middle]:=A[i];
pjgLeft := pjg div 2;
pjgRight := (pjg+1) div 2;
for m:= 1 to pjgLeft do
writeln('ArrayLeft ke-',m,' :
',ArrLeft[m]);
for m:= 1 to pjgRight do
writeln('ArrayRight ke-',m,' :
',ArrRight[m]);
ArrLeft:=mergesort(pjgLeft,ArrLeft);
ArrRight:=mergesort(pjgRight,ArrRigh
t);
mergesort:=merge(ArrLeft,pjgLeft,Arr
Right,pjgRight);
end;
end;

begin
clrscr;
write('Jumlah array : ');readln(n);
for m := 1 to n do
begin
write('Array ke-',m,' : ');
readln(ArrMain[m]);
end;
writeln;
ArrUrut := mergesort(n,ArrMain);

for m:= 1 to n do
writeln('Array Urut ke-',m,' :
',ArrUrut[m]);
readln;
end.
```

- **queue_array**

```
Program queue_array;
Const QUEUE_SIZE = 20;
Var
myQueue : Array [1..QUEUE_SIZE] of
integer;
topPointer, i : integer;

Procedure CreateQueue;
Begin
        topPointer := 0;
end;
Function IsEmpty : boolean;
Begin
        IsEmpty := false;
        if topPointer = 0 then
        IsEmpty := true;
end;
Function IsFull : boolean;
Begin
        IsFull := false;
        if ((topPointer + 1) =
QUEUE_SIZE) then
        IsFull := true;
end;
Procedure enQueue (item : integer);
Begin
        if not IsFull then
        begin
                myQueue [topPointer
+ 1] := item;
                topPointer :=
topPointer + 1;
        end;
end;
Function deQueue : integer;
Begin
        if not IsEmpty then
        begin
                deQueue := myQueue
[1];
                topPointer :=
topPointer - 1;
        end;
        //geser posisi
        for i := 1 to topPointer do
        begin
                myQueue [i] :=
myQueue [i + 1];
        End;
end;
Function Getsize : integer;
Begin
        Getsize := topPointer;
end;
 Procedure display;
 Begin
        writeln (' Tampilan data : ');
        for i := 1 to topPointer do
        writeln (myQueue[i]);
 end;

{program utama}
Begin
        CreateQueue;
        enQueue (4);
        enQueue (25);
        enQueue (18);
        enQueue (25);
        enQueue (43);
        display;
        writeln (' Ukuran saat ini ',
Getsize);
        writeln (' deQueue = ',
deQueue);
        writeln (' deQueue = ',
deQueue);
        writeln (' deQueue = ',
deQueue);
        writeln (' Ukuran saat ini ',
Getsize);
        display;
        CreateQueue ; //reset
        writeln (' Ukuran saat ini ',
Getsize, ' setelah reset ');
        display;
        readln;
end.
```

- **Quick Sort**

```
Program quicksort;

Type
 TipeArray = string[20];
 ArrayUrut = array[1..1000] of
TipeArray;

 Procedure QuickSort(var x      :
ArrayUrut;
                    Bawah, Atas : word);

 var
  I, J   : word;
  Sementara : TipeArray;
 Begin
  While Atas > bawah Do
  begin
  I := Bawah;
  J := Atas;
  Sementara := X[Bawah];


  {Memecah Array menjadi 2 bagian}
  While I < J Do Begin
    While X[J] > Sementara Do J := J - 1;
    X[I] := X[J];
    While (I<J) And (X[I] <= Sementara)
Do I := I + 1;
    X[J] := x[I];
  end;

  X[I] := Sementara;

  {Urutkan rekursi}
  QuickSort(X, Bawah, I-1);
  Bawah := I + 1;
 end;
end;

Var
  Nama : ArrayUrut;
  N, I : word;
Begin
  Write('Jumlah data yang akan
diurutkan ?'); ReadLn(N);
  WriteLn;
  WriteLn('Masukkan data :');
  For I:=1 to N Do Begin
    Write('Data ke ',I,' ? ');
ReadLn(Nama[I]);
 end;

 {urutkan dengan prosedur QuickSort}
 QuickSort(Nama,1,N);

 {Tampilkan Data yang telah diurut}
 WriteLn;

 WriteLn('Data yang telah di urut :');
 WriteLn('-------------------------');

 For I := 1 To N Do
   WriteLn(Nama[I]);

 end.
```

- **record_array**

```
Program record_array;
Uses crt;
Type recBrg = record
      IdBrg,nama,satuan : string;
      qty : integer;
      harga : longint;
End;
Var
  brg : array [1..5] of recBrg;
  i : integer;
Begin
        clrscr;
        for i := 1 to 5 do
        begin
                clrscr;
                writeln (' Data ke -', i);
                with brg [i] do
                begin
                        write ('id
Barang = ');readln(idBrg);
                        write ('Nama
= ');readln(nama);
                        write ('Satuan
= ');readln(satuan);
                        write ('Qty =
');readln(qty);
                        write ('Harga
= ');readln(harga);
                end;
        end;
        //tampilkan
writeln (' #Brg     Nama     Qty
Satuan   Harga   Jumlah ');
        writeln ('-----------------------------
-----------------------------------------------');

        for i := 1 to 5 do
        begin
                with brg [i] do
                begin
                        gotoxy
(0,2+i); write (idBrg);
                        gotoxy
(10,2+i); write (nama);
                        gotoxy
(12,2+i); write (qty);
                        gotoxy
(18,2+i); write (satuan);
                        gotoxy
(28,2+i); write (harga);
                        gotoxy
(37,2+i); writeln (qty * harga);
                end;
        end;
        readln;
End.

//nusron wahid
```

- **record_dasar**

```
Program record_dasar;
uses crt;
Type recTanggal = record
        hari,bulan,tahun : integer;
 End;
 var tanggal : recTanggal;
 Begin
          clrscr ;
          //masukkan data
    tanggal.hari := 15;
          tanggal.bulan := 10;
          tanggal.tahun := 2016;

          //tampilkan
          writeln ('Hari ini tanggal
',tanggal.hari,' bulan ',tanggal.bulan,'
tahun ',tanggal.tahun);
          readln;
 End.
```

- **record_pointer**

```
Program record_pointer;
Uses crt;
Type PtrMhs = ^RecMhs;
RecMhs = record
npm, nama, kelas : string;
End;
Var mhs : array [1..50] of RecMhs;
     pmhs : PtrMhs;
     i, n : integer;
 Begin
          clrscr;
          write (' Masukkan Jumlah
Data = '); readln (n);
          clrscr;
          for i := 1 to n do
          begin
                    pmhs := @mhs[i];
//menunjuk pada array
                    writeln (' Data ke - ',
i);
                    write (' NPM '); readln
(pmhs^.npm);
                    write (' Nama ');
readln (pmhs^.nama);
                    write (' Kelas ');
readln (pmhs^.kelas);
                    writeln;
          End;
          clrscr;
          //tampilkan data
          for i := 1 to n do
 Begin
          pmhs := @mhs[i];
          writeln (' Data ke - ', i);
          writeln (' NPM = ',
pmhs^.npm);
          writeln (' Nama = ',
pmhs^.nama);
          writeln (' Kelas ',
pmhs^.kelas);
          writeln ('  ');
 end;
 Readln;    End.
```

- **record_with**

```
Program record_with ;
Uses crt ;
Type recMhs = record
      NPM,nama : string ;
      Usia : integer ;
 End ;
 Var mhs : recMhs ;
 begin
          clrscr ;
          with mhs do
          begin
                    write ('NPM = ') ;
readln (npm) ;
                    write ('Nama = ') ;
readln (nama) ;
                    write ('Usia = ') ;
readln (usia) ;
          end ;

          //tampilkan saja

          clrscr ;
          with mhs do
          begin
                    writeln ('NPM = ',
npm) ;
                    writeln ('Nama = ',
nama) ;
                    writeln ('Usia = ',
usia);
          end;
          readln;
 End.
```

- **stack_array**

```
Program stack_array;
Const STACK_SIZE = 20;
Var
myStack : Array [1..STACK_SIZE] of
integer;
topPointer, i : integer;
Procedure CreateStack;
Begin
          topPointer := 0;
end;
Function IsEmpty : boolean;
Begin
          IsEmpty := false;
          if topPointer = 0 then
          IsEmpty := true;
end;
Function IsFull : boolean;
Begin
          IsFull := false;
          if ((topPointer + 1) =
STACK_SIZE) then
          IsFull := true;
end;
Procedure push (item : integer);
Begin
          if not IsFull then
          begin
                    myStack [topPointer +
1] := item;
                    topPointer :=
topPointer + 1;
          end;
end;
Function Pop : integer;
Begin
          if not IsEmpty then
          begin
                    Pop := myStack
[topPointer];
                    topPointer :=
topPointer - 1;
          end;
end;
```

```
Function Getsize : integer;
Begin
        Getsize := topPointer;
end;
 Procedure display;
 Begin
        writeln (' Tampilan data : ');
        for i := 1 to topPointer do
        writeln (myStack[i]);
 end;

{program utama}
Begin
        CreateStack;
        Push (4);
        Push (25);
        Push (18);
        Push (25);
        Push (43);
        display;
        writeln (' Ukuran saat ini ',
Getsize);
        writeln (' Pop = ', Pop);
        writeln (' Pop = ', Pop);
        writeln (' Pop = ', Pop);
        writeln (' Ukuran saat ini ',
Getsize);
        display;
        CreateStack ; //reset
        writeln (' Ukuran saat ini ',
Getsize, ' setelah reset ');
        display;
        readln;
end.
```

- **Tree Order**

```
Program tree_order;
uses Crt;

type
PTR = ^Data;
Data = record
Info: Integer;
Leftson,Rightson: PTR;
end;

type
angkaa = array [1..100]of integer;
var
BinaryTree: PTR;

Menu : char;
Selesai : char;
Angka : integer;

procedure InOrder(bt: PTR);
begin
if (bt <> nil) then
begin
InOrder(bt^.Leftson);
Write(bt^.Info:6);
InOrder(bt^.Rightson);
end;
end;

procedure PreOrder(bt: PTR);
begin
if (bt <> nil) then
begin
Write(bt^.Info:6);
PreOrder(bt^.Leftson);
PreOrder(bt^.Rightson);
end;
end;

procedure PostOrder(bt: PTR);
begin
if (bt <> nil) then
begin
```

```
PostOrder(bt^.Leftson);
PostOrder(bt^.Rightson);
Write(bt^.Info:6);
end;
end;

Procedure Insert(var bt: PTR; Info:
Integer);
var Baru: PTR;
begin
if (bt = nil) then
begin
new(baru);
baru^.Info := Info;
baru^.Leftson := nil;
baru^.Rightson := nil;
bt := baru;
end
else if (Info <= bt^.Info) then
Insert(bt^.Leftson, Info)
else
Insert(bt^.Rightson, Info);
end;

procedure MenuUtama;
begin
WriteLn(' Deretan Angka Dengan
Binary Tree ');
WriteLn('----------------------------------
-');
WriteLn('1. Isi data!');
WriteLn('2. Tampilkan data secara In
order');
WriteLn('3. Tampilkan data secara Pre
order');
WriteLn('4. Tampilkan data secara
Post order');
WriteLn('5. Keluar');
WriteLn('----------------------------------
-');
Write('Silahkan pilih (1-5): ');
end;

{Program utama}
var i,n : integer;
begin
clrscr;
write('Masukkan Jumlah Data Deretan
Angka! :');readln(n);
selesai:='N';
repeat

clrscr;
menuUtama;
readln(menu);

case menu of
'1' :
begin
clrscr;
for i := 1 to n do
begin
write('input data ke ',i,'
:');readln(angka);
insert(binarytree,angka);
end;
end;

'2' :
begin
writeln;
Writeln('In Order');
Writeln('========================
==============');
inorder(binarytree);
writeln;
end;

'3' :
begin
writeln;
Writeln('Pre Order');
Writeln('========================
==============');
preorder(binarytree);
writeln;
end;

'4' :
begin
```

```
writeln;
Writeln('Tampilkan Deretan Angka
Secara Post Order');
Writeln('=======================
================');
postorder(binarytree);
writeln;
end;

'5' :
begin
selesai:='Y';
end;
end;
readln;
until (selesai='Y') or (selesai='y');
end.
```