

Membaca Notasi Algoritma dalam tugas ini

→ : Hasil balik fungsi (return)
 ← : Assignment (=)
 = : Operator pembandingan (==)
 ↑ : Operator Dereferensi (*)
 ↓ : Operator Referensi (&)
 Input : Parameter Input
 Output : Parameter output
INPUT : Mengambil data yang di input (scanf)
OUTPUT : Menulis data ke layar (printf)

```

/*pdparray.h*/
#ifndef PDPARRAY_H
#define PDPARRAY_H
#include<stdio.h>
#include<stdlib.h>
/*
  Mendefinisikan tipe data boolean
*/
#define true 1
#define false 0
#define boolean unsigned char
/*
  Mendefinisikan maximum data
  kontainer pada array
*/
#define MAXDATA 100
/*Maksimal elemen yang di ijinakan <=100*/

#define ElmtNil -99
/*
  ElmtNil adalah nilai dari elemen yang tidak terpakai.
  Jika isi/nilai seluruh elemen dalam DATA= -99 berarti
  DATA dianggap Kosong/belum dipakai, selain itu pasti isi
  X
  |          +---+---+---+---+---+---+---+---+---+---+---+---+   +---+
  +---->|-99|-99|-99|-99|-99|-99|-99|-99|-99|-99|-99|...|-99|
          +---+---+---+---+---+---+---+---+---+---+---+---+   +---+
IdxEff   0   .....                               99
IdxEff=0, karena semua elemen X bernilai -99
  X
  |          +---+---+---+---+---+---+---+---+---+---+---+---+   +---+
  +---->|35 |40 |22 |50 |67 |-99|-99|-99|-99|-99|-99|...|-99|
          +---+---+---+---+---+---+---+---+---+---+---+---+   +---+
IdxEff   0   1   2   3   4   .....                               99
IdxEff=4, karena ada 4 elemen X yang TIDAK bernilai -99
*/

```

```
/*
  Definisi tipe data baru bernama
  DATA adalah array of integer dengan
  kapasitas 100 elemen integer
*/
typedef int DATA[MAXDATA];
/*untuk menangani hasil balik fungsi
  perlu pointer, shg dapat mengakses parameter
  array dan mengembalikan nilai dalam fungsi
*/
typedef int *PDATA;
/*
  IdxEff adalah di gunakan untuk membatasi jumlah
  elemen yang terpakai dalam DATA. Karena DATA
  tersedia 0..99, jika kita menginginkan hanya 20 elemen
  saja, maka variabel global ini kita set menjadi
  IdxEff=20, sehingga traversal dilakukan hingga
  maximal 20, dan bukan 100.
  keyword extern berarti variabel ini dapat di akses di luar
  file pdparray.h
  IdxEff dapat berubah-ubah, selama program berjalan (running)
  dengan definisi IdxEff<=MAXDATA
*/
extern int IdxEff;

/*Fungsi dan Prosedur pengolahan array*/
/*Konstruktor*/
void InitData(DATA D);
/*
  Menyetel IdxEff=0, dan mengisi semua elemen
  dari DATA ke ElmtNil(-99).
*/
/*Manajemen Memory*/
PDATA Alokasi(int besar);
/*
  Mengalokasikan data int* sebesar "besar"
  Dengan menggunakan (int*)malloc()
*/
void Dealokasi(PDATA P);
/*
  Menddealokasikan data int* yang sudah teralokasi
  dengan free
*/

/*Predikat*/
boolean IsEmpty(DATA D);
```

```
/*
  Memeriksa apakah D kosong, D kosong jika
  seluruh nilai elemen D=ElmtNil, dan mengembalikan
  true, jika tidak mengembalikan false
*/
boolean IsFull(DATA D);
/*
  Memeriksa apakah D sudah penuh berisi elemen.
  Dikatakan PENUH :
  Jika IdxEff=MAXDAT dan semua elemen D<>ElmtNil
  Maka mengembalikan true, selain itu false
*/
boolean IsElmtKosong(DATA D,int idx);
/*
  Mengembalikan true jika elemen ke idx bukan ElmtNil
  selain itu false
*/

/*Selektor (getter dan setter)*/
void SetIdxEff(int n);
/*
  Menyetel IdxEff dengan n pada D.
*/
int GetIdxEff();
/*
  Mengembalikan nilai IdxEff yang sedang berlaku
*/

void SetVFirst(DATA D,int value);
/*
  Mengisi Elemen terdepan dengan value pada D
  -Jika D masih kosong maka, mengisi nilai
  elemen ke-0 dengan value, dan IdxEff bertambah 1
  -Jika elemen ke-0 sudah berisi nilai, maka
  akan di ganti dengan value, IdxEff TIDAK BERUBAH
*/
int GetVFirst(DATA D);
/*
  Jika D Isi mengembalikan nilai elemen PERTAMA dari D
  Jika D kosong mengembalikan ElmtNil
*/

int GetVLast(DATA D);
/*
  Jika D Isi mengembalikan nilai elemen TERAKHIR dari D
  Jika D kosong mengembalikan ElmtNil
*/
```

```
*/

void SetVkeIdx(DATA D,int value,int idx);
/*
  Mengisi elemen D[idx]=value,
  jika D[idx] sudah isi maka akan diganti dengan value
  dan IdxEff TIDAK BERUBAH
  Jika D[idx] kosong, maka IdxEff berubah (bertambah 1)
*/

int GetVkeIdx(DATA D,int idx);
/*
  Mengembalikan nilai elemen dari D pada posisi ke idx
*/

/*Pemrosesan Elemen pada DATA*/
void InputData(DATA D, int n);
/*
  Melakukan entry data pada D,sebanyak n kali.
  Perlu dilakukan pengesetan IdxEff sebanyak n.
  Perlu dilakukan pemeriksaan terhadap D
  Jika D Penuh(IsFull), maka entry data tidak dapat
  dilanjutkan.
*/
void PrintData(DATA D);
/*
  Mencetak seluruh data berdasarkan IdxEff, yaitu mulai
  0..IdxEff, dalam format "[1,2,3,4,5,6]"
*/

void PrintValue(int value);
/*
  mencetak suatu nilai integer,
  sebagai pengganti printf
*/

int NbElmt(DATA D);
/*
  Menghitung Jumlah Elemen data array
*/
int MaxElmt(DATA D);
/*
  Mencari Nilai Elemen maximum dari data array
*/
int MinElmt(DATA D);
/*
  Mencari Nilai Elemen minimum dari data array
*/
```

```
*/
float Average(DATA D);
/*
Menentukan rata-rata nilai Elemen dari data array
*/
int ElmtGanjil(DATA D);
/*
Menghitung jumlah elemen ganjil data array
*/
int ElmtGenap(DATA D);
/*
Menghitung jumlah elemen genap data array
*/
int SumOfElmt(DATA D);
/*
Menjumlahkan seluruh nilai elemen data array
*/
int SumOfElmtGanjil(DATA D);
/*
Menjumlahkan seluruh nilai elemen data array yang Ganjil saja
*/
int SumOfElmtGenap(DATA D);
/*
Menjumlahkan seluruh nilai elemen data array yang Genap saja
*/
int ElmtPrima(DATA D);
/*
Menghitung jumlah elemen data array yang Prima
*/
int SumOfElmtPrima(DATA D);
/*
Menjumlahkan seluruh nilai elemen data array yang Prima saja
*/
float AvgPrima(DATA D);
/*
Menghitung rata-rata seluruh nilai elemen data array yang Prima
saja
*/

/*=====*/
/*Operasi terhadap 2 array atau lebih*/
/*=====*/
void PUrutData(DATA D);
/*
Prosedur Mengurutkan Nilai elemen dari kecil ke besar dalam
data array
*/
```

```
PDATA FUrutData(DATA D);
/*
  Fungsi Mengurutkan Nilai elemen dari kecil ke besar dalam data
  array
*/

boolean IsEqArray(DATA d1,DATA d2);
/*
  Membandingkan NILAI dan JUMLAH elemen dari d1 dan d2
  Jika sama mengembalikan true selain itu false
*/
PDATA FAddArray(DATA d1,DATA d2);
/*
  Fungsi Menjumlahkan tiap NILAI ELEMEN dari d1 dan d2
  Syarat Jumlah elemen d1=d2
  Mengembalikan suatu array baru hasil dari d1+d2
*/
void PAddArray(DATA d1,DATA d2,DATA d3);
/*
  Prosedur Menjumlahkan tiap NILAI ELEMEN dari d1 dan d2
  dan hasilnya di simpan di d3.
  d3 di definisikan di luar prosedur
  Syarat Jumlah elemen d1=d2=d3
*/
PDATA FSambungArray(DATA d1,DATA d2);
/*
  Fungsi menyambung tiap ELEMEN dari d1 dan d2
  Syarat Jumlah elemen d1=d2
  Mengembalikan suatu array baru hasil penyambungan d1 dan d2
  dimana jumlah elemen array hasil penyambungan adalah
  jum elmt d1 + jum elmt d2
*/
void PSambungArray(DATA d1,DATA d2,DATA d3);
/*
  Prosedur menyambung tiap ELEMEN dari d1 dan d2
  Syarat Jumlah elemen d1=d2
  Hasil sambungan di simpan di d3
  Jumlah elemen d3=jum elmt d1 + jum elmt d2
*/
void PecahArray(DATA d,DATA d1,DATA d2);
/*
  Prosedur memecah elemen d menjadi 2 data array, misal d1 dan d2
  jika jumlah d ganjil maka berlaku syarat Jumlah elemen d1>=d2
*/
#endif
```

```
/*pdparray.c*/  
  
/*Fungsi dan Prosedur pengolahan array*/  
/*Konstruktor*/  
Procedure InitData(Input D:DATA)  
/*  
Menyetel IdxEff=0, dan mengisi semua elemen  
dari DATA ke ElmtNil(-99).  
*/  
    i:Integer  
    SetIdxEff(0)  
    for i←0 to MAXDATA do  
        D[i]←ElmtNil  
  
/*Manajemen Memori*/  
Function Alokasi(input besar:Integer)→ PDATA  
    Hasil: PDATA  
    hasil ←(int*)malloc(besar*sizeof(int))  
    →hasil  
  
Procedure Dealokasi(Input P: PDATA)  
    free(P)  
  
/*Predikat*/  
Function IsEmpty(Input D:DATA)→ boolean  
/*  
Memeriksa apakah D kosong, D kosong jika  
seluruh nilai elemen D=ElmtNil, dan mengembalikan  
true, jika tidak mengembalikan false  
*/  
    →(GetIdxEff(D)=0);  
  
Function IsFull(Input D:DATA)→ boolean  
/*  
Memeriksa apakah D sudah penuh berisi elemen.  
Dikatakan PENUH :  
Jika IdxEff=MAXDAT dan semua elemen D<>EmltNil  
Maka mengembalikan true, selain itu false  
*/  
    →(GetIdxEff(=)MAXDATA)  
  
Function IsElmtKosong(Input D:DATA,Input idx:Integer)→ boolean  
/*  
Mengembalikan true jika elemen ke idx bukan ElmtNil  
selain itu false
```

```
*/
    →(D[idx]=ElmtNil)

/*Selektor (getter dan setter)*/
Procedure SetIdxEff(Input n:Integer)
/*
    Menyetel IdxEff dengan n pada D.
*/
    IdxEff←n;

Function GetIdxEff()→Integer
/*
    Mengembalikan nilai IdxEff yang sedang berlaku
*/
    →IdxEff;

Procedure SetVFirst(Input D:DATA,Input value:Integer)
/*
    Mengisi Elemen terdepan dengan value pada D
    -Jika D masih kosong maka, mengisi nilai
      elemen ke-0 dengan value, dan IdxEff bertambah 1
    -Jika elemen ke-0 sudah berisi nilai, maka
      akan di ganti dengan value, IdxEff TIDAK BERUBAH
*/
    if IsElmtKosong(D,0) then
        D[0]←value
        IdxEff++
    else D[0]←value

Function GetVFirst(Input D:DATA)→Integer
/*
    Jika D Isis mengembalikan nilai elemen PERTAMA dari D
    Jika D kosong mengembalikan ElmtNil
*/
    if IsEmpty(D) →ElmtNil
    else → D[0]

Function GetVLast(Input D:DATA)→Integer
/*
    Jika D Isi mengembalikan nilai elemen TERAKHIR dari D
    Jika D kosong mengembalikan ElmtNil
*/
    →D[GetIdxEff(D)-1]

Procedure SetVkeIdx(Input D:DATA,Input value,idx:Integer)
/*
    Perlu dilakukan pemeriksaan terhadap elemen D[idx],
    dengan IsElmtKosong();
```

```

Mengisi elemen D[idx]=value,
jika D[idx] sudah isi maka akan diganti dengan value
dan IdxEff TIDAK BERUBAH
Jika D[idx] kosong, maka IdxEff berubah (bertambah 1)
*/
    if (IsElmtKosong(D,idx)) then
        D[idx]←value;
        IdxEff++;
    else
        D[idx]←value;

Function GetVkeIdx(Input D:DATA, Input idx:Integer)→Integer
/*
Mengembalikan nilai elemen dari D pada posisi ke idx,
jika elemen ke-idx tidak kosong, jika kosong mengembalikan
ElmtNil
*/
    if (IsElmtKosong(D,idx-1)) → ElmtNil
    else →(D[idx-1])

/*Pemrosesan Elemen pada DATA*/
Procedure InputData(Input D:DATA,Input n:Integer)
/*
Melakukan entry data pada D,sebanyak n kali.
Perlu dilakukan pengesetan IdxEff sebanyak n.
Perlu dilakukan pemeriksaan terhadap D penuh atau tidak
Jika D Penuh(IsFull), maka entry data tidak dapat
dilanjutkan.
*/
    I:Integer;
    If IsFull(D) then
        Output("Data sudah penuh !")
        →
    else
        OUTPUT("Entry %d elemen di D\n",n)
        OUTPUT("=====\n")
        i←0
        while(i<n)
            OUTPUT("elemen ke-%d :",i+1)
            INPUT("%d",&D[i])
            i++
            IdxEff++

Procedure PrintData(Input D:DATA)
/*
Mencetak seluruh data berdasarkan IdxEff, yaitu mulai
0..IdxEff, dalam format "[1,2,3,4,5,6]"

```

```

*/
{
  i:Integer
  if IsEmpty(D) then
    OUTPUT("Data Kosong !")
    →
  else
    i←0
    OUTPUT("[" )
    while(i<GetIdxEff(D)) do
      OUTPUT("%d",D[i])
      if(i=GetIdxEff(D)-1) then
        break;
      else
        OUTPUT(" , ")
        i++
    OUTPUT(" ]\n")

```

```

Procedure PrintValue(value:Integer)
  OUTPUT("%d\n",value)

```

```

Function FAddArray(Input d1,d2:DATA)→ PDATA

```

```

/*
Fungsi Menjumlahkan tiap NILAI ELEMEN dari d1 dan d2
Syarat Jumlah elemen d1=d2
Mengembalikan suatu array baru hasil dari d1+d2
*/

```

```

  i:Integer, hasil:PDATA
  hasil←Alokasi(GetIdxEff(d1))
  for i←0 to GetIdxEff(d1) do
    *(hasil+i)←(d1[i]+d2[i])
  → hasil

```

```
/*Contoh test driver (driver.c)*/

#include "pdparray.h"

DATA data1;
DATA data2={1,2,3,4,5};
PDATA hasil;

int main()
{
    InitData(data1);
    InputData(data1,5);
    printf("data1: ");PrintData(data1);
    printf("Nilai Elemen Pertama : ");
    PrintValue(GetVFirst(data1));
    printf("Nilai Elemen Terakhir : ");
    PrintValue(GetVLast(data1));
    printf("Nilai Elemen ke-%d : ",2);
    PrintValue(GetVkeIdx(data1,2));
    printf("Nilai Elemen ke-%d : ",7);
    PrintValue(GetVkeIdx(data1,7));

    printf("data1: ");PrintData(data1);
    printf("data2: ");PrintData(data2);
    hasil=Alokasi(GetIdxEff(data1));
    hasil = FAddArray(data1,data2);
    printf("Hasil Penambahan data1 dan data2 adalah :\n");
    printf("hasil: ");PrintData(hasil);
    getch();
    return 0;
}
```