1.1. **ABSTRACT DATA TYPE (ADT)**

ADT adalah sebuah unit struktur data yang terdiri atas tipe data bentukan serta fungsi/prosedur untuk mengolah data bentukan tersebut. Kumpulan fungsi/prosedur di dalam sebuah ADT dikenal dengan istilah primitif. Pada modul ini akan dibahas 3 ADT dasar yang penting yaitu ADT JAM, ADT POINT, dan ADT DATE.

- ADT JAM terdiri atas struct hh,mm,ss; dan memiliki prosedur/fungsi misalnya cetakJam(), ubahKeDetik(), selisihDetik(), dsb.
- ADT POINT terdiri atas struct x,y; dan memiliki prosedur/fungsi misalnya cetakTitik(), isOrigin(), kuadran(), jarakOrigin(), translasi, rotasi, dilatasi, mirror, dsb. ADT ini sangat penting dan akan banyak digunakan di kuliah lanjut seperti Analisis Perancangan Algoritma, Software Project 1 & 2, Komputasi Paralel, Komputasi Kinerja Tinggi, dsb.
- ADT DATE terdiri atas struct dd,mm,yy; dan memiliki prosedur/fungsi misalnya cetakDate(), isKabisat(), jumlahHari(), dsb.

Untuk mengingat kembali tentang tipe bentukan, lihat Modul 9 pada KUG1F1 (Praktikum Pemrograman Terstruktur 1). Berikut ini adalah contoh tipe bentukan tanpa primitif.

```
struct Jam {
        int hh;
        int mm;
        int ss;
} j1, j2, j3;
```

Cara pengaksesan hh, mm, dan ss dilakukan dengan menggunakan tanda titik, contohnya:

```
j1.hh
j1.mm
j1.ss
```

Definisi tipe bentukan sederhana tersebut kita tambahkan beberapa primitif (fungsi/prosedur) untuk membentuk sebuah ADT yang disebut ADT JAM.

Berikut ini adalah definisi ADT JAM dalam notasi algoritmik dan bahasa C.

```
Type Jam: < hh: integer,
            mm: integer,
            ss: integer >
```

```
// FUNGSI UNTUK VALIDASI DAN PEMBUATAN JAM
function isValidJam(h ,m, s: integer)→boolean
```

Function makeJam(h : integer; m : integer; s :integer)→Jam

// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Function bacaJam()→Jam
procedure tulisJam(input j :Jam)
// FUNGSI KONVERSI
Function jamToDetik(Jam j)→integer
Function detikToJam(s : integer)→Jam

// OPERASI ARITMETIK function durasi(jaw : Jam; jakh: Jam)→integer Function jMinus(j1 : Jam j1; j2 : Jam)→Jam Function jPlus(Jam j1, Jam j2)→Jam Function nextDetik(j : Jam)→Jam Function nextNDetik(Jam j, int n)→Jam Function prevDetik(J :Jam)→Jam Function prevNDetik(j : Jam j, n : integer)
// OPERASI RELATIONAL (PERBANDINGAN) Function jEqual(j1 :Jam; j2 :Jam)→boolean function jNotEqual(j1 :Jam; j2 :Jam)→boolean function jLessThan(j1 :Jam; j2 :Jam)→boolean function jGreaterThan(j1 :Jam; j2 :Jam)→boolean

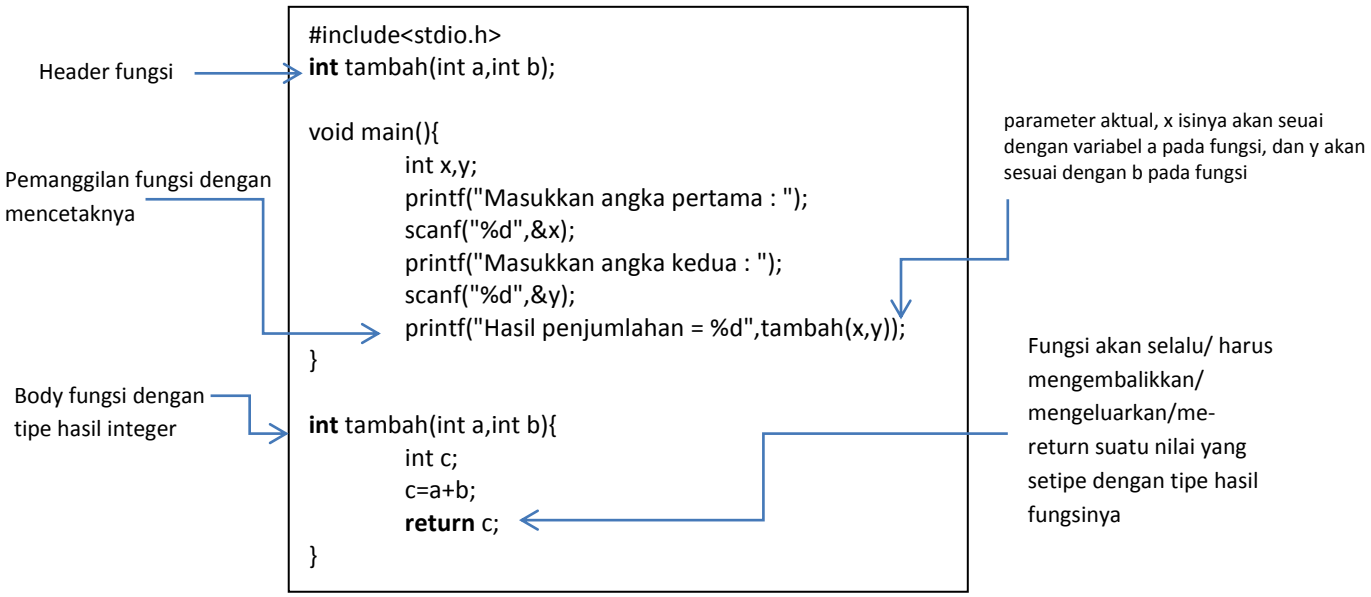
```
typedef struct {
        int hh;
        int mm;
        int ss;
} Jam;

// FUNGSI UNTUK VALIDASI DAN PEMBUATAN JAM
boolean isValidJam(int h, int m, int s);
Jam makeJam(int h, int m, int s);
// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Jam bacaJam();
void tulisJam(Jam j);
// FUNGSI KONVERSI
int jamToDetik(Jam j);
Jam detikToJam(int s);
// OPERASI ARITMETIK
int durasi(Jam jaw, Jam jakh);
Jam jMinus(Jam j1, Jam j2);
Jam jPlus(Jam j1, Jam j2);
Jam nextDetik(Jam j);
Jam nextNDetik(Jam j, int n);
Jam prevDetik(Jam j);
Jam prevNDetik(Jam j, int n);
// OPERASI RELATIONAL (PERBANDINGAN)
boolean jEqual(Jam j1, Jam j2);
```

```
boolean jNotEqual(Jam j1, Jam j2);
boolean jLessThan(Jam j1, Jam j2);
boolean jGreaterThan(Jam j1, Jam j2);
```

Perhatikan bahwa pada definisi di atas, hanya ada header dari fungsi/prosedur. Lalu di mana kita harus menaruh body algoritma dari fungsi/prosedur tersebut? Seperti telah dibahas di Modul 4 KUG1F1 (Praktikum Pemrograman Terstruktur 1) mengenai Fungsi dan Prosedur, kita ketahui bahwa header fungsi/prosedur ditaruh sebelum main(), sedangkan body algoritma fungsi/prosedur ditaruh setelah main(). Semuanya ditaruh dalam sebuah file. Contohnya adalah sebagai berikut:

Penulisan fungsi di bawah (setelah) main program



Pada sistem ADT, file tersebut akan dipecah menjadi 3 bagian yaitu:

- Header file: berisi tipe bentukan dan header fungsi/prosedur
- Body file: berisi body algoritma fungsi/prosedur (harus mengacu pada header file, dan juga harus meng-“include” header file.
- Driver file: berisi fungsi utama main() untuk memanggil dan menguji fungsi/prosedur apakah sudah betul algoritmanya atau belum.

Tujuan pemecahan tersebut adalah supaya lebih rapi/terstruktur, dan supaya lebih mudah dibaca. Seorang programmer yang ingin mengetahui garis besar sebuah ADT dapat melihat header file-nya saja, tanpa perlu melihat body filenya. Body file biasanya panjang dan rumit karena berisi body algoritma dari semua prosedur/fungsi yang telah didefinisikan headernya di header file.

Setiap ADT terdiri atas 3 file yaitu header file, body file, dan driver file. Sebagai contoh, ADT JAM akan terdiri atas file header **jam.h**, file body **jam.c**, dan file driver **mjam.c.** Untuk lebih lengkapnya, pelajari Subbab 1.2 tentang ADT JAM.

1.2. **ADT JAM**

ADT JAM adalah ADT paling sederhana yang tipe bentukannya terdiri atas jam (hh), menit (mm), dan detik (ss). Berikut ini adalah notasi algoritmik untuk pendefinisian ADT JAM.

1.2.1. Header File: jam.h

```
Type Jam: < hh: integer,
            mm: integer,
            ss: integer >
```

```
// FUNGSI VALIDASI
function isValidJam(h ,m, s: integer)→boolean
{I.S.: h, m, s adalah masukan integer sebarang
 F.S.: return true bila h, m, s berada dalam range yang benar untuk jam (0..23, 0..59, 0..59) }
```

```
// FUNGSI KONSTRUKTOR
Function makeJam(h : integer;  m : integer; s :integer)→Jam
{I.S.: h, m, s adalah masukan integer sebarang
F.S.: lakukan pengecekan dengan memanggil fungsi isValid, kemudian bentuklah sebuah Jam dari
komponen h, m, s}
```

```
// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Function bacaJam()→Jam
{ membaca komponen H,M,S sehingga membentuk J yang valid}
procedure tulisJam(input j :Jam)
{menampilkan jam ke layar dengan format [hh:mm:ss]}


// FUNGSI KONVERSI
Function jamToDetik(Jam j)→integer
{Rumus : detik = 3600*hour+menit*60+detik}
Function detikToJam(s : integer)→Jam
```

```
// OPERASI ARITMETIK
function durasi(jaw : Jam; jakh: Jam)→integer
Function  jMinus(j1 : Jam j1; j2 : Jam)→Jam
Function jPlus(Jam j1, Jam j2)→Jam
Function nextDetik(j : Jam)→Jam
Function nextNDetik(Jam j, int n)→Jam
Function  prevDetik(J :Jam )→Jam
Function prevNDetik(j : Jam j, n : integer)
```

```
// OPERASI RELATIONAL (PERBANDINGAN)
Function jEqual(j1 :Jam; j2 :Jam )→boolean
function jNotEqual(j1 :Jam; j2 :Jam )→boolean
function jLessThan(j1 :Jam; j2 :Jam)→boolean
function jGreaterThan(j1 :Jam; j2 :Jam)→boolean
```

1.2.2. Body File: jam.c

```
// FUNGSI VALIDASI
function isValidJam(h ,m, s: integer)→boolean
Kamus
Algoritma :
        if((h>=0) and (h<24) and (m>=0) and (m<60) and (s>=0) and (s<60)) then
                →true
        else
                →false
        endif

// FUNGSI KONSTRUKTOR
Function makeJam(h : integer;  m : integer; s :integer)→Jam
Kamus :
        j:Jam
Algoritma :
        If (isValidJam(h,m,s)=true) then
                j.hh←h;
                j.mm←m;
                j.ss←s;
        else
                j.hh←0;
                j.mm←0;
                j.ss←0;
        endif
        →j;


// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Function bacaJam()→Jam
Kamus :
        h,m,s : integer
        stat : boolean
        j : Jam

Algoritma :
        stat ←false
        while (stat=false) do
                input("Jam: ", h);
                input("Menit: ", m);
                input("Detik : ", s)
                if(isValidJam(h,m,s)=true) then
                        j.hh←h;
                        j.mm←m;
                        j.ss←s;
```

```
                        stat←true;
                else
                        output("Jam yang anda masukkan tidak valid. Silahkan masukkan kembali jam.
\n");
                        stat=false;
                endif
        endwhile
        →j;


procedure tulisJam(input j :Jam)
Kamus :
Algoritma :
        output(" : ",j.hh, " : " ,j.mm, " : " ,j.ss);


// FUNGSI KONVERSI
Function jamToDetik(Jam j)→integer
Kamus :
        jum : integer
algoritma :
        jum←(j.hh*3600)+(j.mm*60)+j.ss;
        →jum;

Function detikToJam(s : integer)→Jam
Kamus :
        jam, menit, detik : integer
        j : Jam
Algoritma :
        Jam←s/3600;
        menit←(s mod 3600)/60;
        detik←(s mod 6000)%60;
        if (isValidJam(jam,menit,detik)=true) then
                j.hh←jam;
                j.mm←menit;
                j.ss←detik;
        else
                j.hh←0;
                j.mm←0;
                j.ss←0;
        endif
        → j


// OPERASI ARITMETIK
function durasi(jaw : Jam; jakh: Jam)→integer
Kamus :
        jam1, jam2, selisih : integer
```

```
Algoritma :
        jam1←jamToDetik(jaw);
        jam2←jamToDetik(jakh);
        selisih←abs(jam1-jam2);
        →selisih;



Function  jMinus(j1 : Jam j1; j2 : Jam)→Jam
Kamus :
        selisih : integer
        j : Jam
Kamus :
        if((j1.hh<j2.hh) or ((j1.hh=j2.hh) and (j1.mm<=j2.mm))) then
                selisih←durasi(j2,j1);
                j←detikToJam(selisih);
        else
                j.hh←0;
                j.mm←0;
                j.ss←0;
        endif
        →j;


Function jPlus(Jam j1, Jam j2)→Jam
Kamus
        s1, s2, jum : integer
         j : Jam
Algoritma :
        s1←jamToDetik(j1);
        s2←jamToDetik(j2);
        jum←s1+s2;
        j←detikToJam(jum);
        → j;


Function nextDetik(j : Jam)→Jam
Kamus :
Algoritma :
        if((j.ss>=0) and (j.ss<59)) then
                j.ss←j.ss+1
        else
                if((j.mm>=0) and (j.mm<59)) then
                        j.ss←0;
                        j.mm←j.mm+1;
                else if(j.mm=59) then
                        j.ss←0;
                        j.mm←0;
```

```
                        j.hh←j.hh+1
                endif
        endif
        → j;


Function  prevDetik(J :Jam )→Jam
Kamus :
Algoritma :
        if(j.ss=0) then
                if(j.mm=0) then
                        j.ss←59;
                        j.mm←59;
                        j.hh←j.hh-1;
                else
                        j.ss←59;
                        j.mm←j.mm-1;
                endif
        else
                j.ss←j.ss-1;
        endif

// OPERASI RELATIONAL (PERBANDINGAN)
Function jEqual(j1 :Jam; j2 :Jam )→boolean
Kamus :
Algoritma :
        if((j1.hh=j2.hh) and (j1.mm=j2.mm) and (j1.ss=j2.ss)) then
                → true;
        else
                → false;
        endif

function jNotEqual(j1 :Jam; j2 :Jam )→boolean
Kamus :
Algritma :
        if(jEqual(j1,j2)=true) then
                → false;
        else
                → true;
        endif

function jLessThan(j1 :Jam; j2 :Jam)→boolean
Kamus:
        jam1,jam2 : integer
Algoritma :
        jam1←jamToDetik(j1);
        jam2←jamToDetik(j2);
        if(jam1<jam2) then
```

```
                → true;
        end
                → false;
        endif

function jGreaterThan(j1 :Jam; j2 :Jam)→boolean
Kamus :
Algoritma :
        if(jLessThan(j1,j2)=true) then
                →false;
        else
                → true;
        endif
```

1.2.3. Header File Dalam Bahasa C: jam.h

```
#include <stdio.h>
#include "boolean.h"

#ifndef jam_H
#define jam_H

typedef struct {
        int hh;
        int mm;
        int ss;
} Jam;

// FUNGSI VALIDASI
boolean isValidJam(int h, int m, int s);
/* return true bila h, m, s berada dalam range yang benar untuk jam (0..23, 0..59, 0..59) */

// FUNGSI KONSTRUKTOR
Jam makeJam(int h, int m, int s);
/* lakukan pengecekan dengan memanggil fungsi isValid, kemudian bentuklah sebuah Jam dari
komponen h, m, s */

// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Jam bacaJam();
void tulisJam(Jam j);
// Prosedur tulisJam: menampilkan jam ke layar dengan format [hh:mm:ss]

// FUNGSI KONVERSI
int jamToDetik(Jam j);
Jam detikToJam(int s);

// OPERASI ARITMETIK
int durasi(Jam jaw, Jam jakh);
```

```
Jam jMinus(Jam j1, Jam j2);
// Fungsi jMinus: mengecek j1<=j2, lalu menghitung j2-j1, outputnya berupa Jam juga
Jam jPlus(Jam j1, Jam j2);
// Fungsi jPlus: menjumlahkan j1+j2, outputnya berupa Jam juga
Jam nextDetik(Jam j);
Jam nextNDetik(Jam j, int n);
Jam prevDetik(Jam j);
Jam prevNDetik(Jam j, int n);

// OPERASI RELATIONAL (PERBANDINGAN)
boolean jEqual(Jam j1, Jam j2);
boolean jNotEqual(Jam j1, Jam j2);
boolean jLessThan(Jam j1, Jam j2);
boolean jGreaterThan(Jam j1, Jam j2);
#endif
```

1.2.4. Body File Dalam Bahasa C: jam.c

```
#include <stdio.h>
#include "jam.h"

// FUNGSI VALIDASI
boolean isValidJam(int h, int m, int s){
/* Fungsi isValidJam:
I.S.: h, m, s adalah masukan integer sebarang
F.S.: return true bila h, m, s berada dalam range yang benar untuk jam (0..23, 0..59, 0..59)
*/
        if((h>=0)&&(h<24)&&(m>=0)&&(m<60)&&(s>=0)&&(s<60)){
                return true;
        }else{
                return false;
        }
}


// FUNGSI KONSTRUKTOR
Jam makeJam(int h, int m, int s){
/* Fungsi makeJam:
I.S.: h, m, s adalah masukan integer sebarang
F.S.: lakukan pengecekan dengan memanggil fungsi isValid, kemudian bentuklah sebuah Jam dari
komponen h, m, s
*/
        Jam j;

        if(isValidJam(h,m,s)==true){
                j.hh=h;
```

```
                j.mm=m;
                j.ss=s;
        }else{
                j.hh=0;
                j.mm=0;
                j.ss=0;
        }
        return j;
}


// FUNGSI UNTUK MEMBACA INPUT DARI KEYBOARD DAN MENULIS KE LAYAR
Jam bacaJam(){
        int h,m,s;
        boolean stat;
        Jam j;

        stat=false;
        while(stat==false){
                printf("Jam: ");
                scanf("%d",&h);
                printf("Menit: ");
                scanf("%d",&m);
                printf("Detik: ");
                scanf("%d",&s);
                if(isValidJam(h,m,s)==true){
                        j.hh=h;
                        j.mm=m;
                        j.ss=s;
                        stat=true;
                }else{
                        printf("Jam yang anda masukkan tidak valid. Silahkan masukkan kembali jam.
\n");
                        stat==false;
                }
        }
        return j;
}

void tulisJam(Jam j){
// Prosedur tulisJam: menampilkan jam ke layar dengan format [hh:mm:ss]
        printf("%d : %d : %d \n",j.hh,j.mm,j.ss);
}

// FUNGSI KONVERSI
int jamToDetik(Jam j){
        int jum;
```

```
	jum=(j.hh*3600)+(j.mm*60)+j.ss;
	return jum;
}

Jam detikToJam(int s){
	int jam, menit, detik;
	Jam j;

	jam=s/3600;
	menit=(s%3600)/60;
	detik=(s%6000)%60;
	if(isValidJam(jam,menit,detik)==true){
		j.hh=jam;
		j.mm=menit;
		j.ss=detik;
	}else{
		j.hh=0;
		j.mm=0;
		j.ss=0;
	}
	return j;
}


// OPERASI ARITMETIK
int durasi(Jam jaw, Jam jakh){
	int jam1, jam2, selisih;
	jam1=jamToDetik(jaw);
	jam2=jamToDetik(jakh);
	selisih=abs(jam1-jam2);
	return selisih;
}

Jam jMinus(Jam j1, Jam j2){
// Fungsi jMinus: mengecek j1<=j2, lalu menghitung j2-j1, outputnya berupa Jam juga
	int selisih;
	Jam j;
	if((j1.hh<j2.hh)||((j1.hh==j2.hh)&&(j1.mm<=j2.mm))){
		selisih=durasi(j2,j1);
		j=detikToJam(selisih);
	}else{
		j.hh=0;
		j.mm=0;
		j.ss=0;
	}
	return j;
}
```

```
Jam jPlus(Jam j1, Jam j2){
// Fungsi jPlus: menjumlahkan j1+j2, outputnya berupa Jam juga
	int s1, s2, jum;
	Jam j;

	s1=jamToDetik(j1);
	s2=jamToDetik(j2);
	jum=s1+s2;
	j=detikToJam(jum);
	return j;
}

Jam nextDetik(Jam j){
	if((j.ss>=0)&&(j.ss<59)){
		j.ss++;
	}else{
		if((j.mm>=0)&&(j.mm<59)){
			j.ss=0;
			j.mm++;
		}else if(j.mm==59){
			j.ss=0;
			j.mm=0;
			j.hh++;
		}
	}
	return j;
}
Jam prevDetik(Jam j){
	if(j.ss==0){
		if(j.mm==0){
			j.ss=59;
			j.mm=59;
			j.hh=j.hh-1;
		}else{
			j.ss=59;
			j.mm=j.mm-1;
		}
	}else{
		j.ss=j.ss-1;
	}
}

// OPERASI RELATIONAL (PERBANDINGAN)
boolean jEqual(Jam j1, Jam j2){
	if((j1.hh==j2.hh)&&(j1.mm==j2.mm)&&(j1.ss==j2.ss)){
		return true;
	}else{
		return false;
```

```
	}
}

boolean jNotEqual(Jam j1, Jam j2){
	if(jEqual(j1,j2)==true){
		return false;
	}else{
		return true;
	}
}

boolean jLessThan(Jam j1, Jam j2){
	int jam1,jam2;

	jam1=jamToDetik(j1);
	jam2=jamToDetik(j2);
	if(jam1<jam2){
		return true;
	}else{
		return false;
	}
}
boolean jGreaterThan(Jam j1, Jam j2){
	if(jLessThan(j1,j2)==true){
		return false;
	}else{
		return true;
	}
}
```

1.2.5. Driver: mjam.c

```
#include <stdio.h>
#include "jam.h"

int main()
{
	Jam ja = makeJam(6,30,45);
	tulisJam(ja);

	Jam jb = makeJam(17,59,59);
	tulisJam(jb);

	printf("ja diubah ke detik = %i\n",jamToDetik(ja));
	printf("jb diubah ke detik = %i\n",jamToDetik(jb));

	Jam jc = detikToJam(7285);
	tulisJam(jc);
```

```
	printf("Durasi atau selisih dari ja-jb= %i\n",durasi(ja,jb));

	printf("[0=No,1=Yes] jEqual(ja,jb) = %i\n",jEqual(ja,jb));
	printf("[0=No,1=Yes] jNotEqual(ja,jb) = %i\n",jNotEqual(ja,jb));
	printf("[0=No,1=Yes] jLessThan(ja,jb) = %i\n",jLessThan(ja,jb));
	printf("[0=No,1=Yes] jGreaterThan(ja,jb) = %i\n",jGreaterThan(ja,jb));

	return 0;
}
```