ONE

STRUKTUR DATA, ADT, dan STRUCT

PENGANTAR STRUKTUR DATA

- Bagaimana cara mengatasi masalah implementasi data dengan komputer?
    - Pemahaman masalah secara menyeluruh dan persiapan data
    - Keputusan operasi-operasi yang dilakukan terhadap data
    - Penyimpanan data-data pada memori sehingga tersimpan dan terstruktur secara logis, operasinya efisien
    - Pengambilan keputusan terhadap bahasa pemrograman mana yang paling cocok untuk jenis data yang ada

Perbedaan antara Tipe Data, Obyek Data dan Struktur Data

- Tipe data adalah jenis data yang ditangani oleh suatu bahasa pemrograman pada komputer.
- Tiap-tiap bahasa pemrograman memiliki tipe data yang memungkinkan:
    - Deklarasi terhadap variabel tipe data tersebut
    - Menyediakan kumpulan operasi yang mungkin terhadap variabel bertipe data tersebut
    - Contoh tipe data di C? Java? Pascal? .NET?
- Obyek Data adalah kumpulan elemen yang mungkin untuk suatu tipe data tertentu.
    - Mis: integer mengacu pada obyek data -32768 s/d 32767, byte 0 s/d 255, string adalah kumpulan karakter maks 255 huruf

- Struktur Data adalah cara penyimpanan dan pengorganisasian data-data pada memori komputer maupun file pada media penyimpanan secara efektif sehingga dapat digunakan secara efisien, termasuk operasi-operasi di dalamnya.
- Di dalam struktur data kita berhubungan dengan 2 aktivitas:
    - Mendeskripsikan kumpulan obyek data yang sah sesuai dengan tipe data yang ada
    - Menunjukkan mekanisme kerja operasi-operasinya
    - Contoh: integer (-32768 s/d 32767) dan jenis operasi yang diperbolehkan adalah +, -, *, /, mod, ceil, floor, <, >, != dsb.
    - Struktur data = obyek data + [operasi manipulasi]
- Dengan pemilihan struktur data yang baik, maka problem yang kompleks dapat diselesaikan dengan algoritma yang dapat digunakan secara efisien, operasi-operasi penting dapat dieksekusi dengan sumber daya yang lebih kecil, memori lebih kecil, dan waktu eksekusi yang lebih cepat.
- Ciri algoritma yang baik menurut Donald E.Knuth:
    - Input: ada minimal 0 input atau lebih
    - Ouput: ada minimal 1 output atau lebih
    - Definite: ada kejelasan apa yang dilakukan
    - Efective: langkah yang dikerjakan harus efektif
    - Terminate: langkah harus dapat berhenti (stop) secara jelas
- Tidak semua struktur data baik dan sesuai. Contoh untuk problem data bank, problem pengurutan dan pencarian data berbeda.

ADT (Abstract Data Type) atau Tipe Data Bentukan

- Bahasa pemrograman bisa memiliki tipe data:
    - Built-in : sudah tersedia oleh bahasa pemrograman tersebut
        - Tidak berorientasi pada persoalan yang dihadapi.

- UDT : User Defined Type, dibuat oleh pemrogram.
  - Mendekati penyelesaian persoalan yang dihadapi
  - Contoh: record pada Pascal, struct pada C, class pada Java
- ADT : Abstract Data Type
  - memperluas konsep UDT dengan menambahkan pengkapsulan atau enkapsulasi, berisi sifat-sifat dan operasi-operasi yang bisa dilakukan terhadap kelas tersebut.
  - Contoh: class pada Java

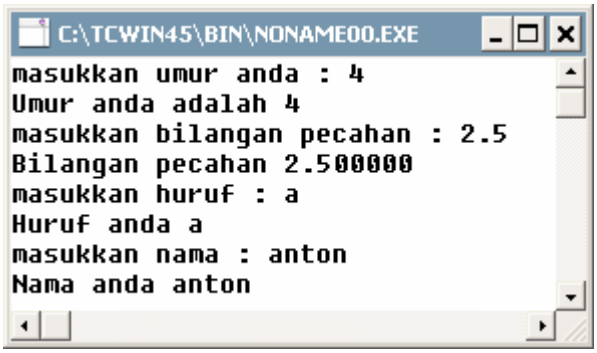
- Bahasa C memiliki tipe data numerik dan karakter (seperti int, float, char dan lain-lain). Disamping itu juga memiliki tipe data enumerasi dan structure. Bagaimana jika kita ingin membuat tipe data baru?
- Untuk pembuatan tipe data baru digunakan keyword **typedef**
- Bentuk umum:

```
typedef <tipe_data_lama> <ama_tipe_data_baru>
```

Contoh:

```
#include <stdio.h>
#include <conio.h>

typedef int angka;
typedef float pecahan;
typedef char huruf;

void main(){
      clrscr();
      angka umur;
      pecahan pecah;
      huruf h;

      huruf nama[10];

      printf("masukkan umur anda : ");scanf("%d",&umur);
      printf("Umur anda adalah %d",umur);

      printf("\nmasukkan bilangan pecahan : ");scanf("%f",&pecah);
      printf("Bilangan pecahan %f",pecah);

      printf("\nmasukkan huruf : ");h=getche();
      printf("\nHuruf anda %c",h);

      printf("\nmasukkan nama : ");scanf("%s",nama);
      printf("Nama anda %s",nama);
```

```
      getch();
}
```



Struct

- Struct adalah tipe data bentukan yang berisi kumpulan variabel-variabel yang bernaung dalam satu nama yang sama dan memiliki kaitan satu sama lain.
- Berbeda dengan array hanya berupa kumpulan variabel yang bertipe data sama, struct bisa memiliki variabel-variabel yang bertipe data sama atau berbeda, bahkan bisa menyimpan variabel yang bertipe data array atau struct itu sendiri.
- Variabel-variabel yang menjadi anggota struct disebut dengan elemen struct.
- Bentuk umum:

```
typedef struct{
      tipe_data <nama_var>;
      tipe_data <nama_var>;
      ....
}
```

Ilustrasi Struct

Struct bisa diumpamakan sebagai sebuah obyek, misalnya: obyek Mahasiswa

Struct Mahasiswa memiliki property atau atribut atau variabel yang melekat padanya:

- NIM yaitu karakter sejumlah 8
- Nama yaitu karakter
- IPK yaitu bilangan pecahan

Struct tidak memiliki operasi (method) atau function.

Struct dapat digunakan dengan cara membuat variabel yang bertipe struct tersebut.

Misalnya :
variabel anton bertipe struct Mahasiswa
variabel erick bertipe struct Mahasiswa

Dengan demikian variabel anton dan erick memiliki NIM, Nama, dan IPK masing-masing

Pendeklarasian dan penggunaan Struct (1) (menggunakan *typedef*)

```
typedef struct Mahasiswa {
     char NIM[8];
     char nama[50];
     float ipk;
};

//untuk menggunakan struct Mahasiswa dengan membuat variabel mhs dan mhs2
Mahasiswa mhs,mhs2;

//untuk menggunakan struct Mahasiswa dengan membuat variabel array m;
Mahasiswa m[100];
```

Pendeklarasian dan penggunaan Struct (2) (tanpa menggunakan *typedef*)

```
struct {
     char NIM[8];
     char nama[50];
     float ipk;
} mhs;
```

Berarti kita sudah mempunyai **variabel** mhs yang bertipe data struct seperti diatas.

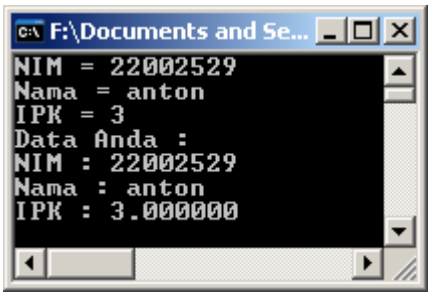
Cara penggunaan struct dan pengaksesan elemen-elemennya

- Penggunaan/pemakaian tipe data struct dilakukan dengan membuat suatu variabel yang bertipe data struct tersebut
- Pengaksesan elemen struct dilakukan secara individual dengan menyebutkan nama variabel struct diikuti dengan operator titik (.)
- Misalnya dengan struct mahasiswa seperti contoh di atas, kita akan akses elemen-elemennya seperti contoh berikut:

Contoh 1

```
#include <stdio.h>
#include <conio.h>

//Pendeklarasian tipe data baru struct Mahasiswa
typedef struct Mahasiswa{
         char NIM[9];
         char nama[30];
         float ipk;
        };

void main(){
      //Buat variabel mhs bertipe data Mahasiswa
      Mahasiswa mhs;
      clrscr();

      printf("NIM = ");scanf("%s",mhs.NIM);
      printf("Nama = ");scanf("%s",mhs.nama);
      printf("IPK = ");scanf("%f",&mhs.ipk);

      printf("Data Anda : \n");
      printf("NIM : %s\n",mhs.NIM);
      printf("Nama : %s\n",mhs.nama);
      printf("IPK : %f\n",mhs.ipk);
      getch();
}
```

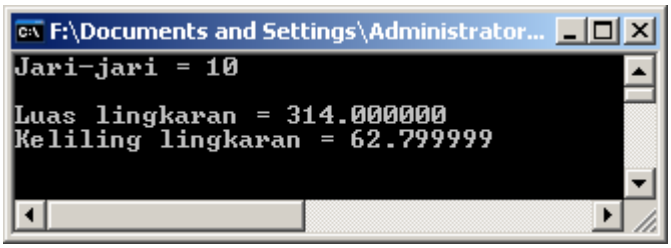
Hasilnya:



Contoh 2

```
#include <stdio.h>
#include <conio.h>
#define phi 3.14

//langsung dianggap variabel 'lingkaran'
struct {
  float jari2;
  float keliling;
  float luas;
} lingkaran;

//fungsi void untuk menghitung luas ingkaran
void luasLingkaran(){
//langsung menggunakan luas lingkaran asli
     lingkaran.luas = lingkaran.jari2 * lingkaran.jari2 * phi;
     printf("\nLuas lingkaran = %f",lingkaran.luas);
}

//fungsi yang mengembalikan nilai float untuk menghitung keliling lingkaran
float kelLingkaran(float j){
      return 2*phi*lingkaran.jari2;
}

int main(){
     clrscr();
     printf("Jari-jari = ");scanf("%f",&lingkaran.jari2);
//panggil fungsi luasLingkaran
     luasLingkaran();
//panggil fungsi keliling, nilai kembaliannya dikirim ke keliling lingkaran asli
     lingkaran.keliling = kelLingkaran(lingkaran.jari2);
//tampilkan keliling lingkaran asli
     printf("\nKeliling lingkaran = %f",lingkaran.keliling);
     getch();
}
```

Hasilnya:

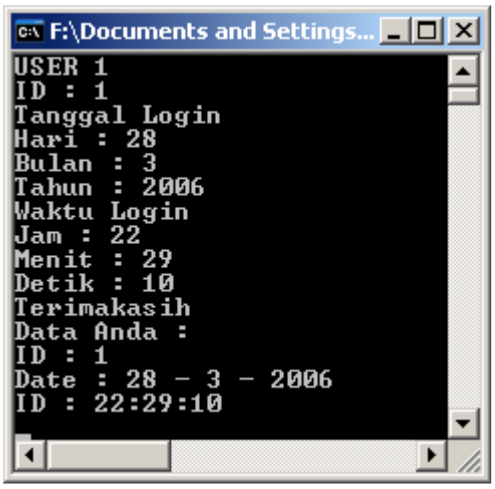


Struct yang berisi struct lain

Contoh:

```
#include <stdio.h>
#include <conio.h>

typedef struct Date{
          int dd;
          int mm;
          int yyyy;
        };

typedef struct Time{
          int h;
          int m;
          int s;
        };

typedef struct Login{
          int ID;
          Date tglLogin;
          Time waktuLogin;
        };

int main(){
     Login user1;

     printf("USER 1\n");
     printf("ID : ");scanf("%d",&user1.ID);
     printf("Tanggal Login\n");
     printf("Hari : ");scanf("%d",&user1.tglLogin.dd);
     printf("Bulan : ");scanf("%d",&user1.tglLogin.mm);
     printf("Tahun : ");scanf("%d",&user1.tglLogin.yyyy);
     printf("Waktu Login\n");
     printf("Jam : ");scanf("%d",&user1.waktuLogin.h);
     printf("Menit : ");scanf("%d",&user1.waktuLogin.m);
     printf("Detik : ");scanf("%d",&user1.waktuLogin.s);
     printf("Terimakasih\n");

     printf("Data Anda :\n");
     printf("ID : %d\n",user1.ID);
     printf("Date : %d - %d -
%d\n",user1.tglLogin.dd,user1.tglLogin.mm,user1.tglLogin.yyyy);
     printf("ID :
%d:%d:%d\n",user1.waktuLogin.h,user1.waktuLogin.m,user1.waktuLogin.s);

     getch();
}
```

Hasil:



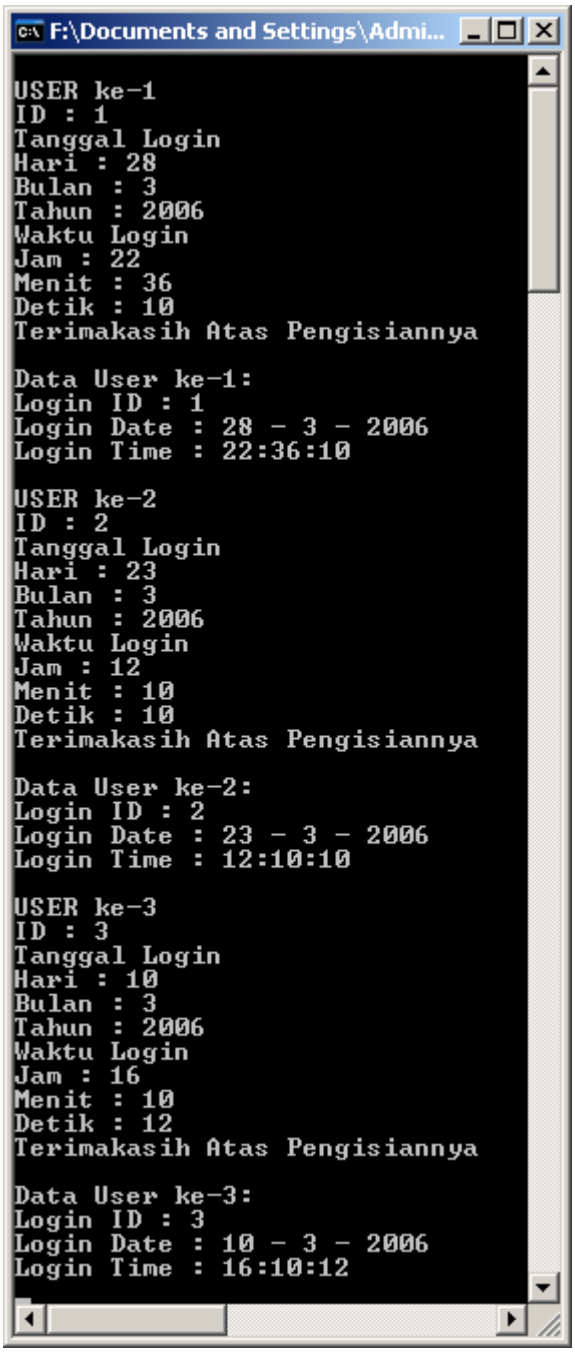
Array of Struct

Contoh

```
#include <stdio.h>
#include <conio.h>

typedef struct Date{
          int dd;
          int mm;
          int yyyy;
        };

typedef struct Time{
          int h;
          int m;
          int s;
        };

typedef struct Login{
          int ID;
          Date tglLogin;
          Time waktuLogin;
        };

int main(){
     Login user[3];

     //3 user
     for(int i=0;i<3;i++){
         printf("\nUSER ke-%d\n",i+1);
         printf("ID : ");scanf("%d",&user[i].ID);
         printf("Tanggal Login\n");
         printf("Hari : ");scanf("%d",&user[i].tglLogin.dd);
         printf("Bulan : ");scanf("%d",&user[i].tglLogin.mm);
         printf("Tahun : ");scanf("%d",&user[i].tglLogin.yyyy);
```

```
        printf("Waktu Login\n");
        printf("Jam : ");scanf("%d",&user[i].waktuLogin.h);
        printf("Menit : ");scanf("%d",&user[i].waktuLogin.m);
        printf("Detik : ");scanf("%d",&user[i].waktuLogin.s);
        printf("Terimakasih Atas Pengisiannya\n");

        printf("\nData User ke-%d:\n",i+1);
        printf("Login ID : %d\n",user[i].ID);
        printf("Login Date : %d - %d -
%d\n",user[i].tglLogin.dd,user[i].tglLogin.mm,user[i].tglLogin.yyyy);
        printf("Login Time :
%d:%d:%d\n",user[i].waktuLogin.h,user[i].waktuLogin.m,user[i].waktuLogin.s);
    }

    getch();
}
```

Hasil

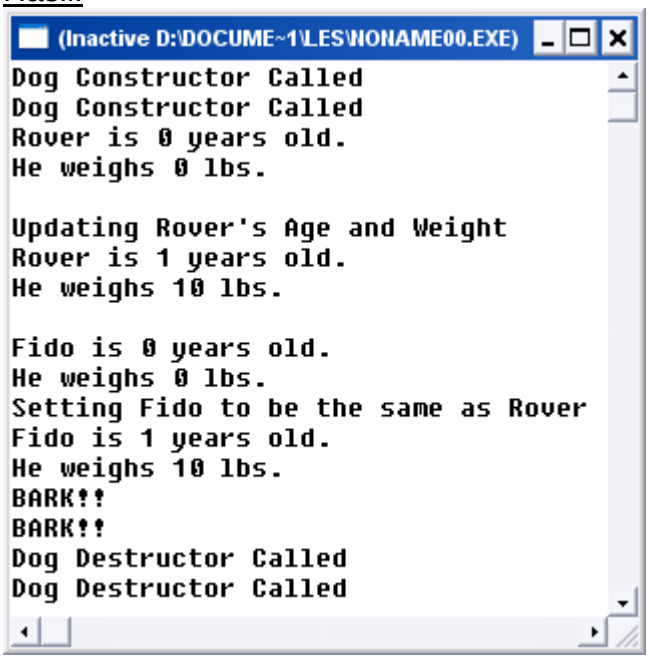


Contoh penggunaan class pada C++:

```
#include <iostream.h>

class Dog {
private:
	int age;
	int weight;
public:
	Dog();	//Constructor
	~Dog();	//Destructor
	void setAge(int age);
	int getAge();
	void setWeight(int weight);
	int getWeight();
	void speak();
};

Dog::Dog()
{
	age = 0;
	weight = 0;
	cout << "Dog Constructor Called" << endl;
}

Dog::~Dog()
{
	cout << "Dog Destructor Called" << endl;
}

void Dog::setAge(int age)
{
	this->age = age;
}

int Dog::getAge()
{
	return age;
}

void Dog::setWeight(int weight)
{
	this->weight = weight;
}

int Dog::getWeight()
{
    return weight;
}

void Dog::speak()
{
    cout << "BARK!!" << endl;
}

int main()
{
	Dog fido;
```

```
    Dog rover;

       cout << "Rover is " << rover.getAge() << " years old." << endl;
    cout << "He weighs " << rover.getWeight() << " lbs." << endl;
    cout << endl;

    cout << "Updating Rover's Age and Weight" << endl;
       rover.setAge(1);
       rover.setWeight(10);

    cout << "Rover is " << rover.getAge() << " years old." << endl;
       cout << "He weighs " << rover.getWeight() << " lbs." << endl;
       cout << endl;

    cout << "Fido is " << fido.getAge() << " years old." << endl;
       cout << "He weighs " << fido.getWeight() << " lbs." << endl;

       cout << "Setting Fido to be the same as Rover" << endl;
    fido = rover;

    cout << "Fido is " << fido.getAge() << " years old." << endl;
    cout << "He weighs " << fido.getWeight() << " lbs." << endl;

       rover.speak();
    fido.speak();

    return 0;
}
```

Hasil:



SOAL-SOAL:

1. Buatlah program menu yang berisi data-data KTP penduduk yang disimpan dalam array struct 1 dimensi dan dapat dilakukan penambahan data, pencarian data, penampilan data dan penghapusan data.

NEXT

Sorting Array