EIGHT

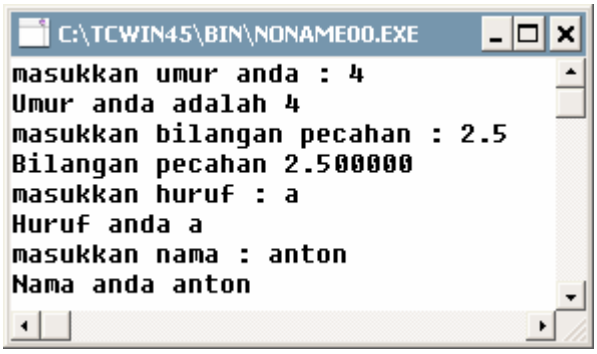
ADT (Abstract Data Type) dan Stack Array

ADT (Abstract Data Type) atau Tipe Data Bentukan

- Bahasa C memiliki tipe data numerik dan karakter (seperti integer, float, char dan lain-lain). Bagaimana jika kita ingin membuat tipe data baru?
- ADT adalah tipe data yang dibuat oleh programmer sendiri yang memiliki suatu nama tertentu.
- ADT dapat berupa tipe data dasar namun diberi nama baru atau berupa kumpulan tipe data berbeda yang diberi nama baru.
- Untuk pembuatan ADT digunakan keyword **typedef**

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.
- Berbeda dengan array yang berisi kumpulan variabel yang bertipe data sama, struct dapat memiliki variabel-variabel yang bertipe data sama atau berbeda, bahkan bisa menyimpan variabel yang bertipe data array atau struct
- Variabel-variabel yang menjadi anggota struct disebut dengan elemen struct

Ilustrasi Struct

Struct bisa diumpamakan sebagai sebuah class, misalnya: 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 hampir mirip dengan class pada Java, namun struct tidak memiliki method atau function.

Struct dapat digunakan dengan cara membuat variabel (analogikan dengan obyek pada Java)

Misalnya :
obyek anton bertipe struct Mahasiswa
obyek erick bertipe struct Mahasiswa

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

Pendeklarasian dan penggunaan Struct (1)

```
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)

```
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

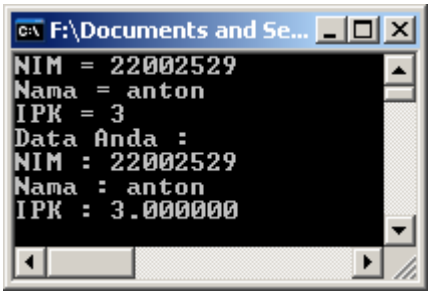
- Penggunaan struct dilakukan dengan membuat suatu variabel yang bertipe struct tersebut
- Pengaksesan elemen struct dilakukan secara individual dengan menyebutkan nama variabel struct diikuti dengan operator titik (.)
- Misalnya dengan struct mahasiswa seperti contoh di atas, kita akan akses elemen-elemennya 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();
}
```

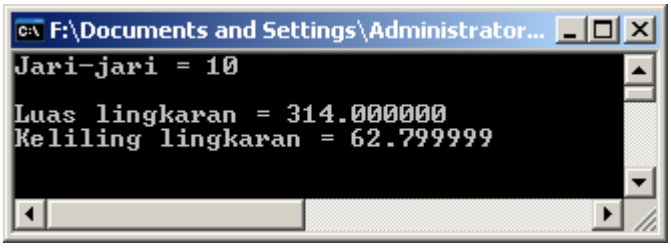
Hasilnya:



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:



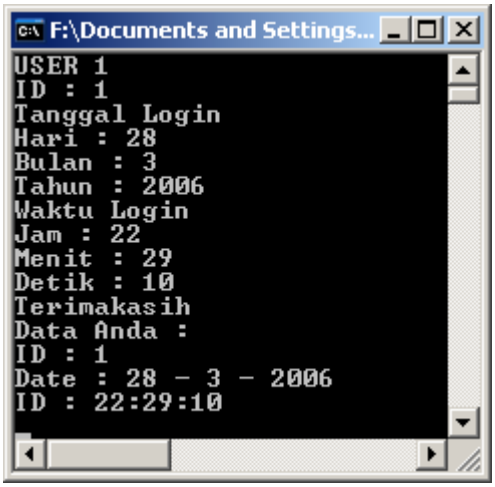
Struct yang berisi struct lain

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:



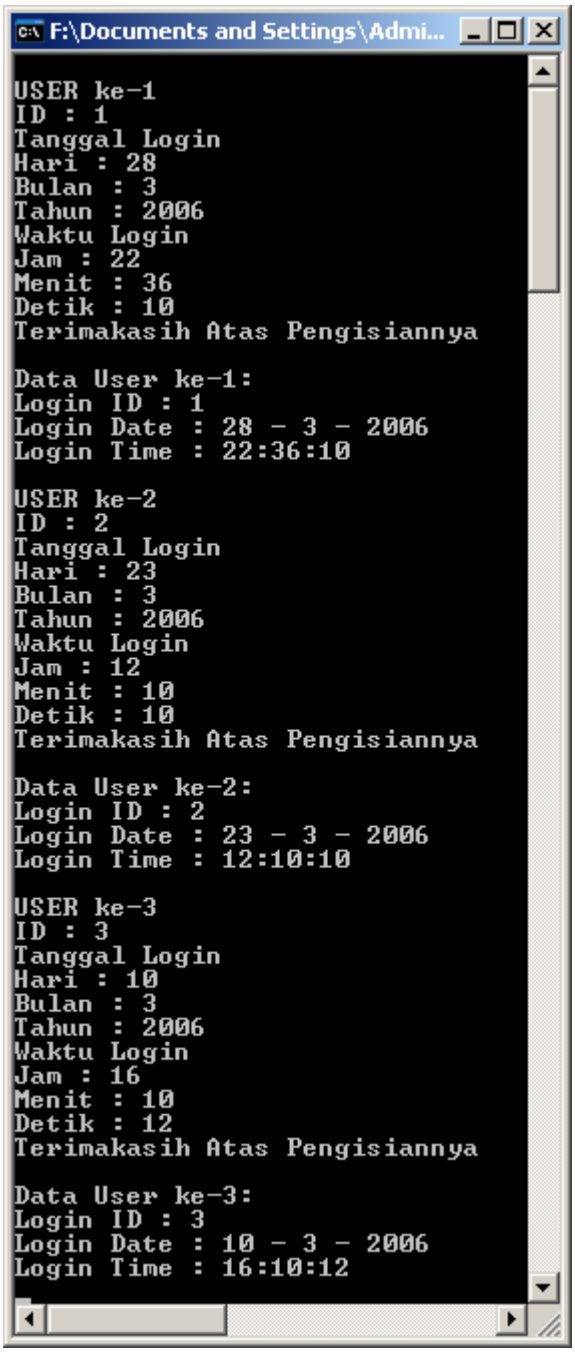
Array of Struct

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");
```

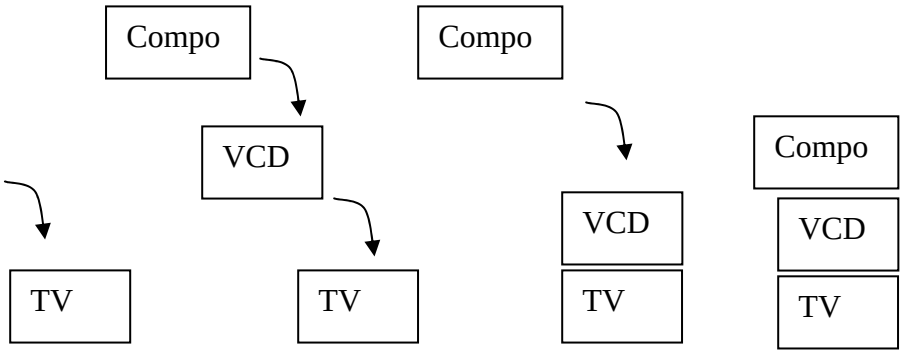
```
        printf("Jam : ");scanf("%d",&user[i].waktuLogin.h);
        printf("Menit : ");scanf("%d",&user[i].waktuLogin.m);
        printf("Detik : ");scanf("%d",&user[i].waktuLogin.s);
        printf("Terimakasih Atas Pengisiannya\n");

        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



STACK

- Stack atau tumpukan adalah suatu stuktur data yang penting dalam pemrograman
- Bersifat LIFO (Last In First Out)
- Benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari stack
- Contohnya, karena kita menumpuk Compo di posisi terakhir, maka Compo akan menjadi elemen teratas dalam tumpukan. Sebaliknya, karena kita menumpuk Televisi pada saat pertama kali, maka elemen Televisi menjadi elemen terbawah dari tumpukan. Dan jika kita mengambil elemen dari tumpukan, maka secara otomatis akan terambil elemen teratas, yaitu Compo juga.



Operasi-operasi/fungsi Stack

- **Push :** digunakan untuk menambah item pada stack pada tumpukan paling atas
- **Pop :** digunakan untuk mengambil item pada stack pada tumpukan paling atas
- **Clear :** digunakan untuk mengosongkan stack
- **IsEmpty :** fungsi yang digunakan untuk mengecek apakah stack sudah kosong
- **IsFull :** fungsi yang digunakan untuk mengecek apakah stack sudah penuh

Stack with Array of Struct

- Definisikan Stack dengan menggunakan struct
- Definisikan MAX_STACK untuk maksimum isi stack
- Buatlah variabel array data sebagai implementasi stack secara nyata
- Deklarasikan operasi-operasi/function di atas dan buat implemetasinya

Deklarasi MAX_STACK

```
#define MAX_STACK 10     //hati-hati mulai dari 0 jadi 0-9
```

Deklarasi STACK dengan struct dan array data

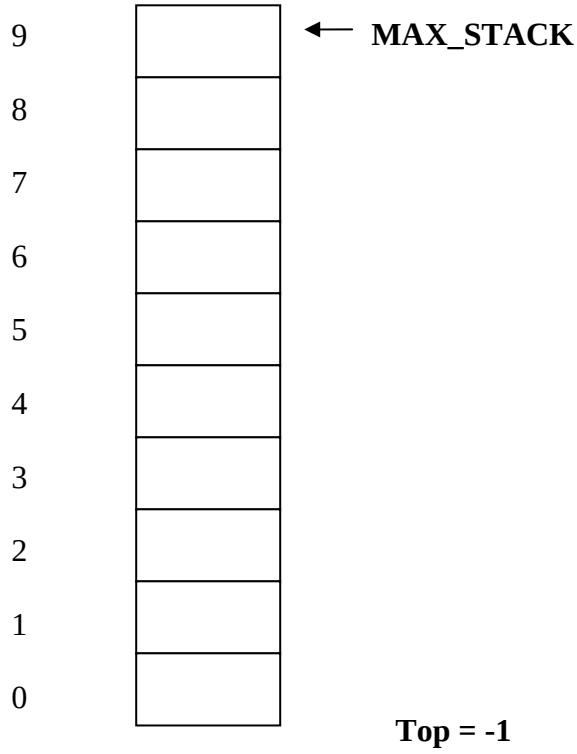
```
typedef struct STACK{
     int top;
     char data[10][10];  //misalkan : data adalah array of string
                         //berjumlah 10 data, masing-masing string
                         //menampung maksimal 10 karakter
};
```

Deklarasi/buat variabel dari struct

```
STACK tumpuk;
```

Inisialisasi Stack

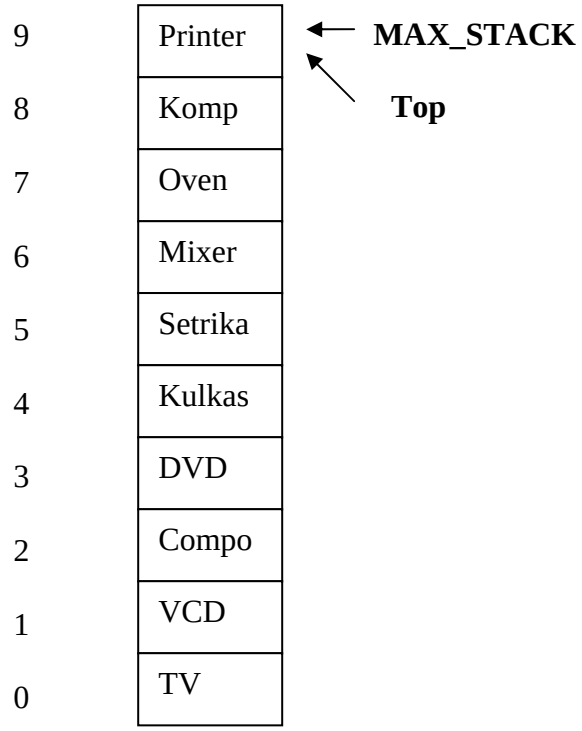
- Pada mulanya isi top dengan -1, karena array dalam C dimulai dari 0, yang berarti stack adalah KOSONG!
- Top adalah suatu variabel penanda dalam STACK yang menunjukkan elemen teratas Stack sekarang. Top Of Stack akan selalu bergerak hingga mencapai MAX of STACK sehingga menyebabkan stack PENUH!
- Ilustrasi stack pada saat inisialisasi:




```
void inisialisasi(){
     tumpuk.top = -1;
}
```

Fungsi IsFull

- Untuk memeriksa apakah stack sudah penuh?
- Dengan cara memeriksa top of stack, jika sudah sama dengan MAX_STACK-1 maka full, jika belum (masih lebih kecil dari MAX_STACK-1) maka belum full
- Ilustrasi:




```
int IsFull(){
      if(tumpuk.top == MAX_STACK-1)
            return 1;
      else
            return 0;
}
```

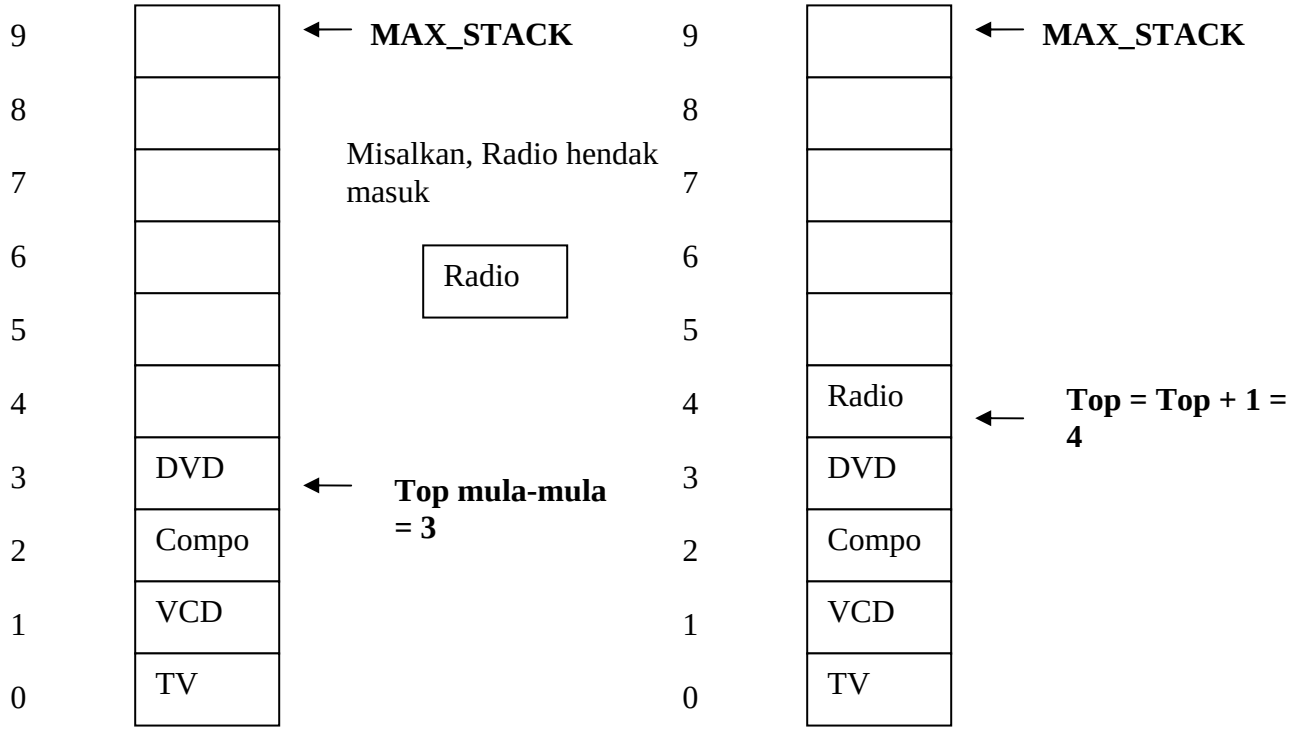
Fungsi IsEmpty

- Untuk memeriksa apakah stack masih kosong?
- Dengan cara memeriksa top of stack, jika masih -1 maka berarti stack masih kosong!
- Program:

```
int IsEmpty(){
  if(tumpuk.top == -1)
      return 1;
  else
      return 0;
}
```

Fungsi Push

- Untuk memasukkan elemen ke stack, selalu menjadi elemen teratas stack
- Tambah satu (increment) nilai top of stack terlebih dahulu setiap kali ada penambahan elemen stack, asalkan stack masih belum penuh, kemudian isikan nilai baru ke stack berdasarkan indeks top of stack setelah ditambah satu (diincrement)
- Ilustrasinya:

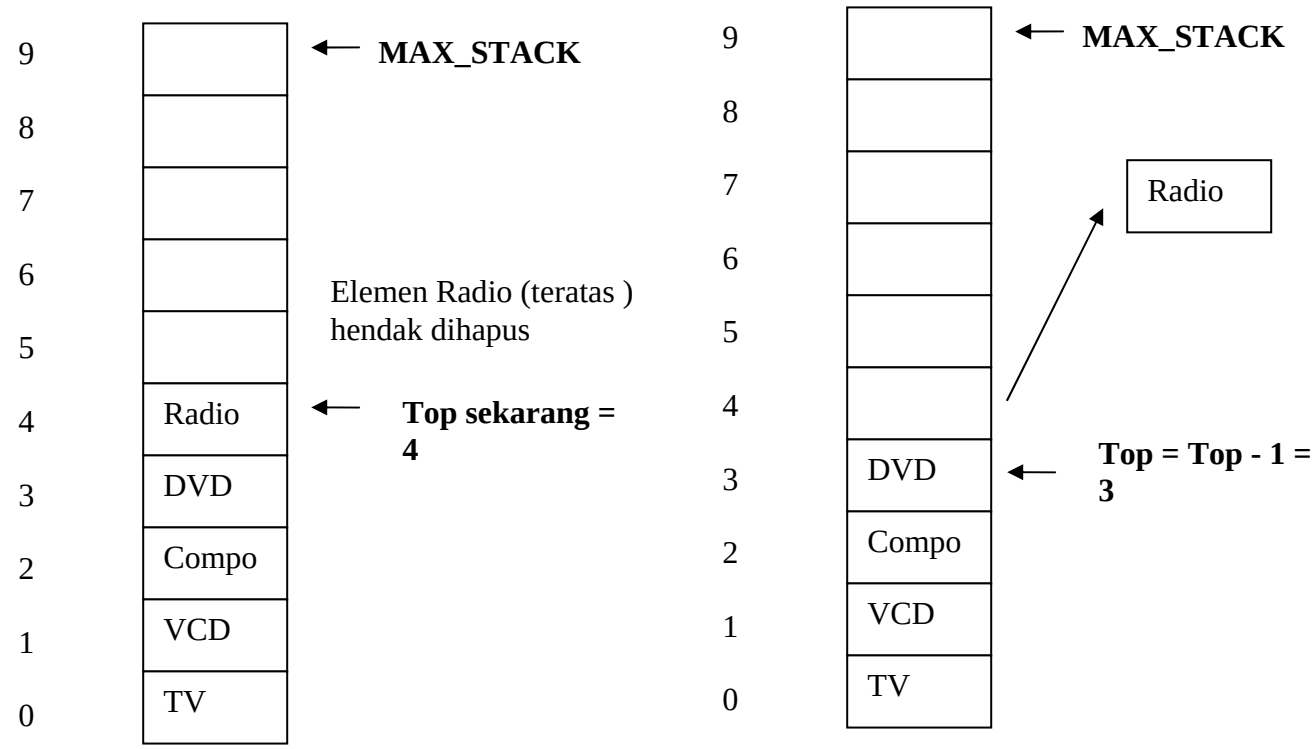



```
void Push(char d[10]){
  tumpuk.top++;
  strcpy(tumpuk.data[tumpuk.top],d);
}
```

Fungsi Pop

- Untuk mengambil elemen teratas dari stack.
- Ambil dahulu nilai elemen teratas stack dengan mengakses top of stack, tampilkan nilai yang akan diambil terlebih dahulu, baru didecrement nilai top of stack sehingga jumlah elemen stack berkurang

- Ilustrasinya:

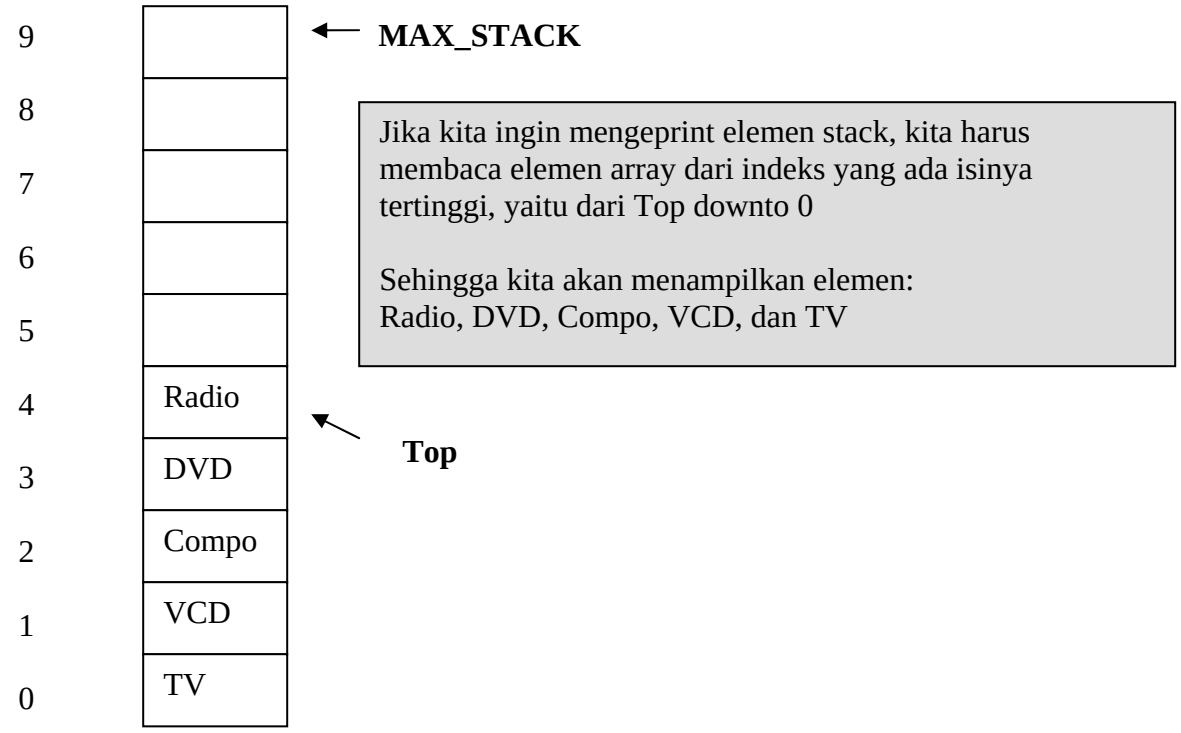


Programnya:

```
void Pop(){
   printf("Data yang terambil = %s\n",tumpuk.data[tumpuk.top]);
   tumpuk.top--;
}
```

Fungsi Print

- Untuk menampilkan semua elemen-elemen stack
- Dengan cara looping semua nilai array secara terbalik, karena kita harus mengakses dari indeks array tertinggi terlebih dahulu baru ke indeks yang kecil!



Program:

```
void TampilStack(){
     for(int i=tumpuk.top;i>=0;i--){
             printf("Data : %s\n",tumpuk.data[i]);
     }
}
```

Program Selengkapnya:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX_STACK 10

typedef struct STACK {
          int top;
          char data[10][10];
        };

STACK tumpuk;

void inisialisasi(){
     tumpuk.top = -1;
```

```
}

int IsFull(){
    if(tumpuk.top == MAX_STACK-1) return 1; else return 0;
}

int IsEmpty(){
    if(tumpuk.top == -1) return 1; else return 0;
}

void Push(char d[10]){
     tumpuk.top++;
     strcpy(tumpuk.data[tumpuk.top],d);
}

void Pop(){
     printf("Data yang terambil = %s\n",tumpuk.data[tumpuk.top]);
     tumpuk.top--;
}

void Clear(){
     tumpuk.top=-1;
}

void TampilStack(){
     for(int i=tumpuk.top;i>=0;i--){
             printf("Data : %s\n",tumpuk.data[i]);
     }
}

int main(){
    int pil;
    inisialisasi();
    char dt[10];
    do{
       printf("1. push\n");
       printf("2. pop\n");
       printf("3. print\n");
       printf("4. clear\n");
       printf("5. exit\n");
       printf("Pilihan : ");scanf("%d",&pil);
       switch(pil){
         case 1: if(IsFull() != 1){
                   printf("Data = ");scanf("%s",dt);
                   Push(dt);
                 } else printf("\nSudah penuh!\n");
                 break;
         case 2: if(IsEmpty() != 1)
                   Pop();
                 else
                   printf("\nMasih kosong!\n");
                 break;
         case 3: if(IsEmpty() != 1)
                   TampilStack();
                 else
                   printf("\nMasih kosong!\n");
                 break;
         case 4: Clear();
                 printf("\nSudah kosong!\n");
```

```
                break;
        }
        getch();
    }while(pil != 5);
    getch();
}
```

PR

1. Tambahkanlah function untuk menghapus salah satu elemen stack
2. Tambahkanlah function untuk mencari suatu elemen dalam stack
3. Tambahkan function untuk mengedit suatu elemen dalam stack
4. Carilah nilai total, rata-rata, terbesar dan terkecil dari elemen-elemen stack dalam function tersendiri

Dikumpul maksimal satu minggu dari sekarang ke email saya: anton@ukdw.ac.id maksimal pukul 13.00 WIB dalam file zip

NEXT WEEK: Queue dengan Array