

STRUKTUR DATA

TEKNIK INFORMATIKA

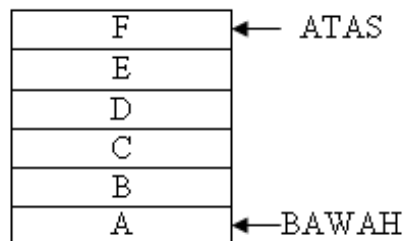
ANGGIT DWI LAKSONO, S.Kom

Pengertian Stack

- Stack dapat dianalogikan seperti suatu tumpukan dari benda
- Stack bersifat LIFO (*Last In First Out*) yaitu benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari stack
- Misalkan kita menumpuk 5 buah kotak apel dan 5 buah kotak jeruk, ketika kotak yang ditumpuk pertama kali adalah kotak apel dan kotak yang ditumpuk terakhir kali adalah kotak jeruk, maka kotak apel tersebut menjadi elemen terbawah dari tumpukan, dan kotak jeruk menjadi elemen teratas dalam tumpukan.
- Jika kita mengambil kotak dalam tumpukan tersebut, maka kotak jeruklah yang akan diambil pertama kali.

□ Ilustrasi Stack

Terdapat dua buah kotak yang ditumpuk, kotak yang satu akan ditumpuk diatas kotak yang lainnya. Jika kemudian stack 2 kotak tadi, ditambah kotak ketiga, keempat, kelima, dan seterusnya, maka akan diperoleh sebuah stack kotak yang terdiri dari N kotak.



Implementasi Stack

- Implementasi/representasi Stack bisa menggunakan **Array** atau **Linked list**.
- Stack yang diimplementasikan dengan array disebut **fixed-length stack**, karena ukurannya tidak bisa berubah.
- Stack yang diimplementasikan dengan linked list dengan pointer disebut **variable-length stack**, karena ukurannya bisa berubah-ubah sesuai dengan dinamika penambahan dan penghapusan elemen-elemennya.

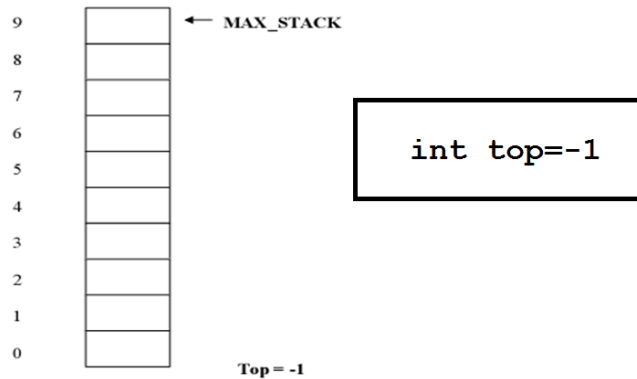
Operasi Stack

- **Create** : membuat stack baru yang masih kosong
- **Push** : untuk menambah/menyisipkan item pada tumpukan paling atas
- **Pop** : untuk mengambil item yang terakhir disisipkan, jika stack tidak kosong
- **Clear** : untuk mengosongkan stack
- **Empty/IsEmpty** : fungsi yang digunakan untuk mengecek apakah stack sudah kosong (mengembalikan nilai true jika stack kosong)
- **Full/IsFull** : fungsi yang digunakan untuk mengecek apakah stack sudah penuh (mengembalikan nilai true jika stack penuh)
- **Retreive/getTop** : untuk mendapatkan/ mengambil nilai yang terakhir disisipkan, jika stack tidak kosong

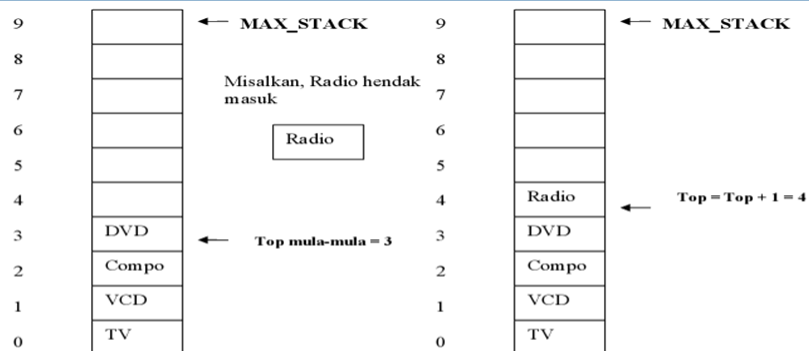
Inisialisasi Stack

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

Ilustrasi Inisialisasi Stack

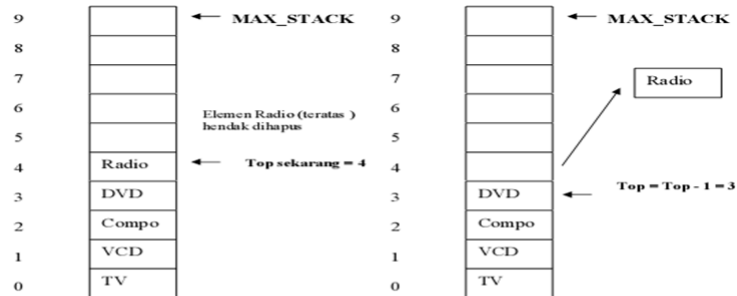


Ilustrasi Push pada Stack



```
void push(int stack[], int top, int value)
{
    top=top+1;
    stack[top]=value;
}
```

Ilustrasi Pop pada Stack



Programnya:

```
void pop(int stack[], int top, int value)
{
    value=stack[top];
    top=top-1;
}
```

Contoh

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
#define MAX_STACK 10
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(){
    cout<<"Data yang terambil = \n"<<tumpuk.data[tumpuk.top];
    tumpuk.top--;
}
void Clear(){
    tumpuk.top=-1;
}

void TampilStack(){
    for(int i=tumpuk.top;i>=0;i--){
        cout<<"Data : "<<tumpuk.data[i]<<endl;
    }
}
```

Contoh lanjutan

```

int main(){
    int pil;
    inisialisasi();
    char dt[10];
    do{
        cout<<"1. push\n";
        cout<<"2. pop\n";
        cout<<"3. print\n";
        cout<<"4. clear\n";
        cout<<"5. exit\n";
        cout<<"Pilihan : ";
        cin>>pil;
        switch(pil){
            case 1: if(IsFull() != 1){ 1)
                cout<<"Data = ";
                cin>>dt;
                Push(dt);
                } else
                cout<<"\nSudah penuh!\n";
                break;
            case 2: if(IsEmpty() != 1)
                Pop();
                else
                cout<<"\nMasih kosong!\n";
                break;
            case 3: if(IsEmpty() != 1)
                cout<<"\nSudah penuh!\n";
                break;
            case 4: Clear();
                cout<<"\nSudah kosong!\n";
                break;
            case 5: exit(0);
        }
        getch();
    }while(pil != 5);
    getch();
}

```

Pengertian Queue

- Struktur data antrian atau queue adalah suatu bentuk khusus dari linear list, dengan operasi penyisipan (insertion) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (REAR), dan operasi penghapusan (deletion) hanya diperbolehkan pada sisi lainnya, yang disebut sisi depan (FRONT), dari list.
- Contoh : kita lihat antrean (Q₁, Q₂, ..., Q_N).
- Notasikan bagian depan dari antrean Q sebagai FRONT(Q) dan bagian belakang sebagai REAR(Q), maka untuk antrean Q = [Q₁, Q₂, ..., Q_N] :

$$\text{FRONT}(Q) = Q_1 \text{ dan } \text{REAR}(Q) = Q_N$$

Implementasi Queue

- Queue dalam dunia nyata :
 - ▣ Barisan bahan atau komponen yang akan diproses suatu mesin
 - ▣ Barisan bahan atau komponen yang akan diproses manusia
- Queue dalam ilmu komputer diimplementasikan dalam :
 - ▣ Antrian data yang akan diproses dalam prosesor
 - ▣ Antrian data yang akan ditulis dalam memori
 - ▣ Pemrosesan banyak job (tugas) pada sistem multiprogramming
 - ▣ Pemrosesan banyak paket yang datang dari banyak koneksi pada suatu host, bridge, gateway.

Prinsip Kerja Queue

- Buatlah struct untuk menampung data pelanggan toko, dimana struct tsb akan digunakan untuk menyimpan nama [char], alamat [char], umur [int], jenis kelamin [char], no telp [int], .
- Buatlah aplikasi untuk mengakses struct tersebut.

Prinsip Kerja Queue

- Misalkan kita mulai dengan antrean hampa Q. Antrean hampa Q, atau $Q[]$ dapat disajikan seperti terlihat pada gambar berikut :

.....

- Misalkan notasi Jumlah(Q) untuk menyatakan jumlah elemen di dalam antrean Q, maka :
 - Jumlah(Q) = 0
 - FRONT(Q) = tidak terdefinisi
 - REAR(Q) = tidak terdefinisi

Prinsip Kerja Queue

- Lalu kita INSERT elemen A, diperoleh $Q = [A]$, seperti terlihat di Gambar

.....
 A

- Maka disini :
 - Jumlah(Q) = 1
 - FRONT(Q) = A
 - REAR(Q) = A

Prinsip Kerja Queue

- Dilanjutkan dengan INSERT elemen B, sehingga diperoleh $Q = [A, B]$, seperti terlihat di Gambar

```

-----
A B
-----

```

- Maka disini :
 - $Jumlah(Q) = 2$
 - $FRONT(Q) = A$
 - $REAR(Q) = B$

Prinsip Kerja Queue

- Dilanjutkan dengan INSERT elemen C, sehingga diperoleh $Q = [A, B, C]$, seperti terlihat di Gambar

```

-----
A B C
-----

```

- Maka disini :
 - $Jumlah(Q) = 3$
 - $FRONT(Q) = A$
 - $REAR(Q) = C$

Prinsip Kerja Queue

- Dilanjutkan dengan DELETE satu elemen dari Q, sehingga diperoleh $Q = [B, C]$, seperti terlihat di Gambar

```

-----
B C
-----

```

- Maka disini :
 - Jumlah(Q) = 2
 - FRONT(Q) = B
 - REAR(Q) = C
- Dan seterusnya, kita dapat melakukan serangkaian INSERT dan DELETE yang lain.
- Suatu kesalahan underflow dapat terjadi, yakni apabila kita melakukan penghapusan pada antrean hampa.

Contoh

```

#include<stdio.h>           t=q[front];           for(i=front;i!=rear;i=(i+1)%QSIZE
#include<iostream.h>        if(front!=rear)        E)
#include<conio.h>           front=(front+1)%QSIZE;    cout<<q[i]<<"\t";
#define QSIZE 5             else{                  cout<<q[i]<<"\n";
int front=0,rear=-1,q[QSIZE]; front=0;                }}
void insert(int x)          rear=-1;          void destroy(){
{                             }}                int i=0;
rear=(rear+1)%QSIZE;        return t;          if(rear<=-1)
q[rear]=x;                  }                cout<<"\nAntrian Kosong.\n\n";
}                             void display(){        else
int del(){                  int i;                for(i=front;i<=rear;i=(front+1)%
int t=0;                     if(rear<=-1)        QSIZE)
if(rear<=-1)                cout<<"\nAntrian Kosong.\n\n"; del();
cout<<"\nAntrian kosong.\n\n"; else{            if(i!=0)
else{                        cout<<"\nGilirannya adalah\n\n"; cout<<"\nAntrian selesai.\n\n";
                                }

```

Contoh lanjutan

```

void main(){
    int ch,in,d;
    clrscr();
    do{
        cout<<"\nmain menu
        bioskop\n";
        cout<<"1.no.kursi\n";
        cout<<"2.hapus salah satu
        antrian\n";
        cout<<"3.urutan masuk \n"; cin>>in;
        cout<<"4.hapus seluruh
        antrian\n";
        cout<<"5.keluar menu\n";
        cout<<"tentukan pilihan:";
        cin>>ch;

        switch(ch){
            case 1:
                if(front==(rear+1)%QSIZE
                && rear>=QSIZE-1)
                    break;
                case 3:
                    display();
                    break;
                case 4:
                    destroy();
                    break;
                case 5:break;
                default:clrscr();
                cout<<"\npilihan anda
                tepat.\n\n"; }}
            while(ch!=5); }

        cout<<"\n kursi terpakai
        nomor="<<d<<"\n\n";
    }
}

```