

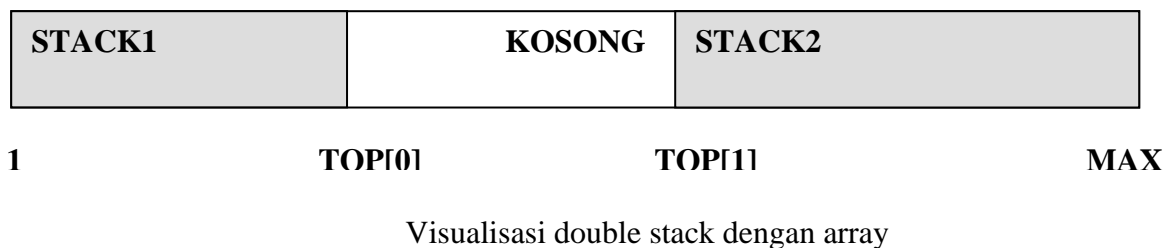
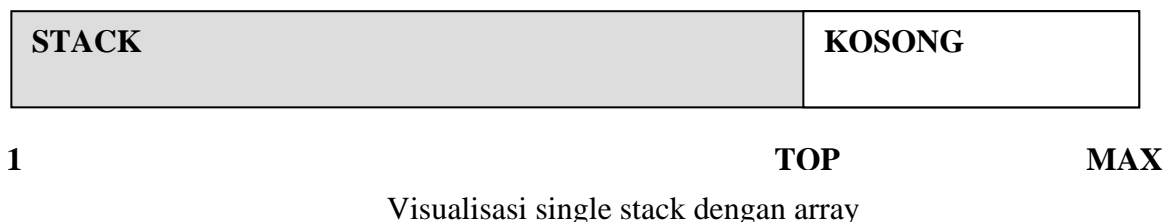
NINE

Queue dengan Array

LANJUTAN:

DOUBLE STACK dengan Array

- Adalah suatu teknik khusus yang dikembangkan untuk menghemat pemakaian memori.
- Intinya adalah penggunaan sebuah array untuk menampung dua stack



- Array dibagi dua tempat, dimana stack1 bergerak ke kanan dan stack2 bergerak ke kiri.
- Jika Top[0] (elemen teratas dari stack1) bertemu dengan Top[1] (elemen teratas dari stack2) maka stack akan penuh.

DEKLARASI DOUBLE STACK

```
#define MAX 8
typedef struct{
    int data[MAX];
} Stack;

Stack stack;
int top[2];
```

OPERASI-OPERASI DALAM DOUBLE STACK

- Create()

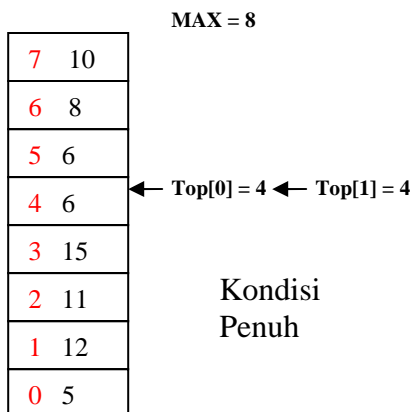
- o digunakan untuk menciptakan dan menginisialisasi double stack
- o dipanggil pada saat program pertama kali dijalankan.
- o dengan cara menginisialisasi top[0] dengan -1 karena elemen array pertama di C adalah 0, dan menginisialisasi top[1] dengan MAX karena elemen terakhir dari array MAX-1.



```
void Create(){  
    //kosongkan stack  
    top[0] = -1;  
    top[1] = MAX;  
    //top[1] = 8 karena elemen array 0-7  
}
```

- IsFull()

- o Digunakan untuk mengecek apakah semua elemen array yang disediakan telah terisi penuh semua atau belum.
- o Cara: periksa nilai top[0] dan top[1], jika nilai top[0]+1 >= top[1] maka berarti batas top[0] sudah melebihi top[1] yang berarti penuh.



```
int IsFull(){  
    int full = 0;  
    if(top[0]+1 >= top[1]) full=1;  
    return full;  
}
```

- IsEmpty(noStack)

- o Digunakan untuk mengecek apakah stack tertentu kosong atau tidak.
- o Dengan cara mengecek nilai top[0] untuk stack1 dan mengecek top[1] untuk stack yang kedua.

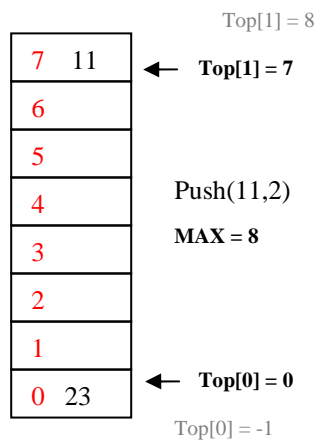
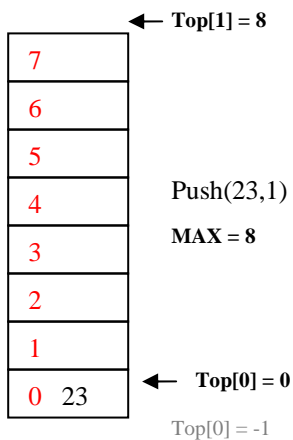
- o Jika $top[0]$ sama dengan -1 dan $top[1] = MAX-1$ maka kosong.

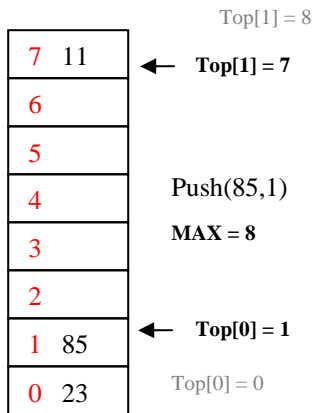


```
int IsEmpty(int noStack){
    int empty=0;
    if(noStack==1){
        if(top[0] == 0) empty=1;
    } else
    if(noStack==2){
        if(top[1] == MAX) empty=1;
    }
    return empty;
}
```

- **Push(data,noStack)**

- o Untuk memasukkan data ke suatu stack tertentu.
- o Untuk setiap stack yang ada, tambah/kurangi counter untuk top yang dimaksud.
- o Untuk stack1 maka naikan counter $top[0]$
- o Untuk stack2 maka kurangi counter $top[1]$

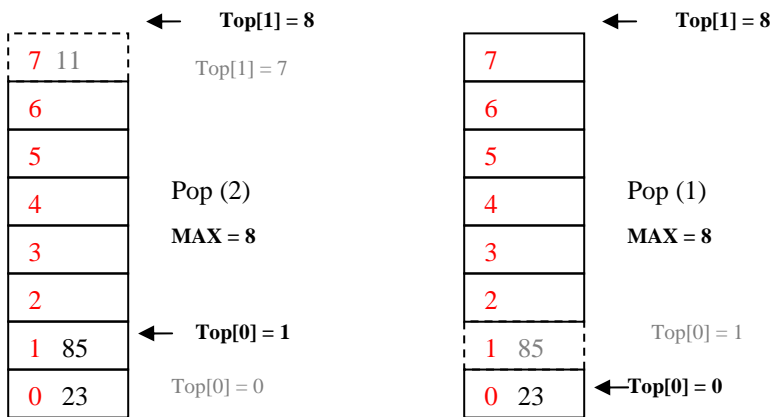




```
void Push(int d, int noStack){
    if(IsFull()==0){
        if(noStack == 1) top[0]++; else
        if(noStack == 2) top[1]--;
        stack.data[top[noStack-1]] = d;
        printf("masuk %d
        !",stack.data[top[noStack-1]]);
    }
}
```

Pop(noStack)

- o Untuk menghapus elemen teratas dari tumpukan stack yang dipilih berdasarkan noStack.
- o Jika stack1 maka kurangi counter untuk top[0], sebelumnya tampilkan nilai teratas untuk stack1 tersebut.
- o Jika stack2 maka tambah counter untuk top[1], sebelumnya tampilkan nilai teratas untuk stack2 tersebut.



```
int Pop(int noStack){
    int e;
    if(IsEmpty(noStack)==0){
        e=stack.data[top[noStack-1]];
        if(noStack==1) top[0]--; else
        if(noStack==2) top[1]++;
    }
    return e;
}
```

- **Clear()**

- o Untuk menghapus semua elemen yang ada di suatu stack sesuai dengan noStack yang ada
- o Dengan cara mengeset top[0] = -1 jika noStack = 1 dan top[1] = MAX jika noStack = 2

```
void Clear(int noStack){
    if(noStack==1) top[0]=-1; else
    if(noStack==2) top[1]=MAX;
}
```

- **Tampil()**

- o Digunakan untuk menampilkan elemen-elemen yang ada di stack1 dan stack2
- o Untuk stack1 : buatlah looping dari 0 s/d top[0]
- o Untuk stack2 : buatlah looping dari MAX-1 downto top[1]

```
void Tampil() {
    int i;
    printf("Stack 1\n");
    for(i=0;i<=top[0];i++){
        printf("%d ",stack.data[i]);
    }
    printf("\nStack 2\n");
    for(i=MAX-1;i>=top[1];i--){
        printf("%d ",stack.data[i]);
    }
}
```

Program selengkapnya adalah sebagai berikut:

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

#define MAX 8
typedef struct {
    int data[MAX];
} Stack;

Stack stack;
int top[2];

void Create(){
    //kosongkan stack
    top[0] = -1;
    top[1] = MAX; //top[1] = 8 karena elemen array 0-7
}
```

```

int IsFull(){
    int full = 0;
    if(top[0]+1 >= top[1]) full=1;
    return full;
}

void Push(int d, int noStack){
    if(IsFull()==0){
        if(noStack == 1) top[0]++; else
        if(noStack == 2) top[1]--;
        stack.data[top[noStack-1]] = d;
        printf("masuk %d !",stack.data[top[noStack-1]]);
    }
}

int IsEmpty(int noStack){
    int empty=0;
    if(noStack==1){
        if(top[0] == 0) empty=1;
    } else
    if(noStack==2){
        if(top[1] == MAX) empty=1;
    }
    return empty;
}

int Pop(int noStack){
    int e;
    if(IsEmpty(noStack)==0){
        e=stack.data[top[noStack-1]];
        if(noStack==1) top[0]--; else
        if(noStack==2) top[1]++;
    }
    return e;
}

void Clear(int noStack){
    if(noStack==1) top[0]=-1; else
    if(noStack==2) top[1]=MAX;
}

void Tampil(){
    int i;
    printf("Stack 1\n");
    for(i=0;i<=top[0];i++){
        printf("%d ",stack.data[i]);
    }
    printf("\nStack 2\n");
}

```

```

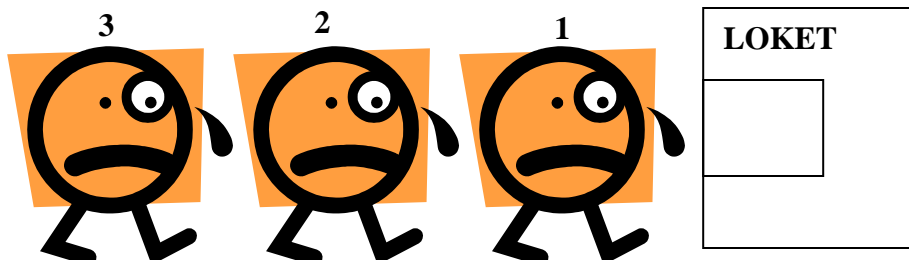
        for(i=MAX-1;i>=top[1];i--){
            printf("%d ",stack.data[i]);
        }
    }

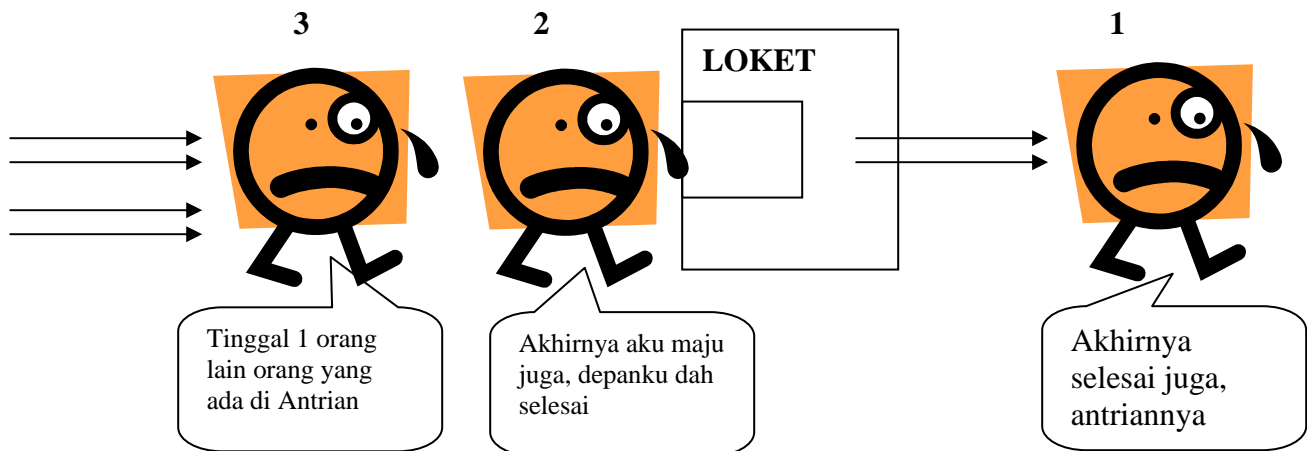
void main(){
    int pil;
    int no;
    int data;
    Create();
    do{
        clrscr();
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Tampil\n");
        printf("4. Clear\n");
        printf("5. Exit\n");
        printf("Pilihan = ");scanf("%d",&pil);
        switch(pil){
            case 1:    printf("stack ke (1/2) : ");scanf("%d",&no);
                    printf("Data = ");scanf("%d",&data);
                    Push(data,no);
                    break;
            case 2:    printf("stack ke (1/2) : ");scanf("%d",&no);
                    printf("Elemen yang keluar : %d",Pop(no));
                    break;
            case 3:    Tampil();
                    break;
            case 4:    printf("stack ke (1/2) : ");scanf("%d",&no);
                    Clear(no);
                    break;
        }
        getch();
    } while(pil!=5);
}

```

QUEUE DENGAN MENGGUNAKAN ARRAY

- Queue = Antrian
- Elemen yang pertama kali masuk ke antrian akan keluar pertama kalinya

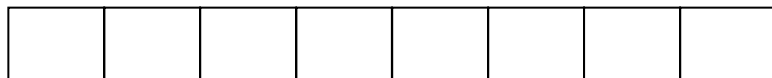




- DEQUEUE adalah mengeluarkan satu elemen dari suatu Antrian
- Antrian dapat dibuat dengan menggunakan: Linier Array dan Circular Array

QUEUE DENGAN LINIEAR ARRAY

- Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya
- Sehingga membutuhkan variabel Head dan Tail



↑ 0 1 2 3 4 5 6 7 MAX = 8

Head = -1

Tail = -1

DEKLARASI QUEUE

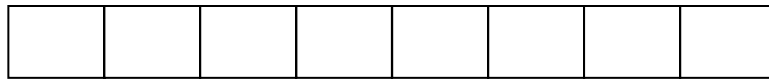
```
#define MAX 8
typedef struct{
    int data[MAX];
    int head;
    int tail;
} Queue;

Queue antrian;
```

OPERASI-OPERASI PADA QUEUE

- Create()

- o Untuk menciptakan dan menginisialisasi Queue
- o Dengan cara membuat Head dan Tail = -1



↑ 0 1 2 3 4 5 6 7 MAX = 8

Head = -1

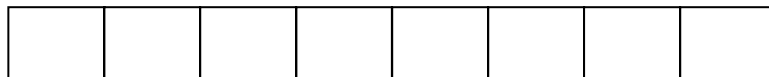
Antrian pertama kali

Tail = -1

```
void Create(){
    antrian.head=antrian.tail=-1;
}
```

- IsEmpty()

- o Untuk memeriksa apakah Antrian sudah penuh atau belum
- o Dengan cara memeriksa nilai Tail, jika Tail = -1 maka empty
- o Kita tidak memeriksa Head, karena Head adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah
- o Pergerakan pada Antrian terjadi dengan penambahan elemen Antrian kebelakang, yaitu menggunakan nilai Tail



↑ 0 1 2 3 4 5 6 7 MAX = 8

Head = -1

Antrian Kosong

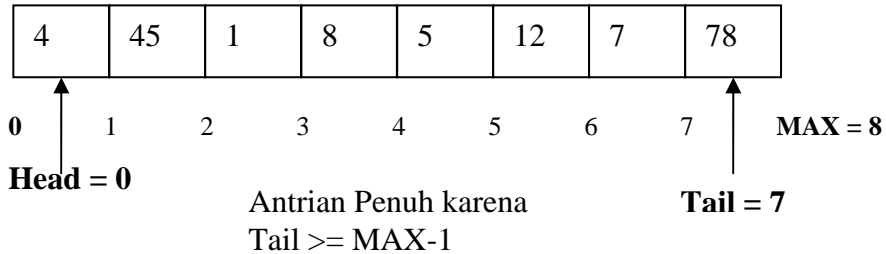
karena Tail = -1

Tail = -1

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

- IsFull()

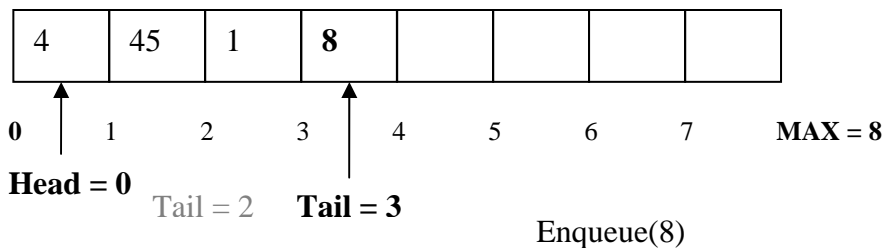
- o Untuk mengecek apakah Antrian sudah penuh atau belum
- o Dengan cara mengecek nilai Tail, jika Tail \geq MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh



```
int IsFull(){
    if(antrian.tail==MAX-1) return 1;
    else return 0;
}
```

- Enqueue(data)

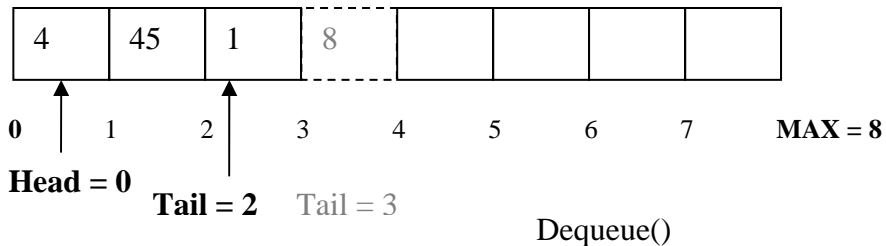
- o Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang
- o Penambahan elemen selalu menggerakkan variabel Tail dengan cara increment counter Tail



```
void Enqueue(int data){
    if(IsEmpty()==1){
        antrian.head=antrian.tail=0;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    } else
    if(IsFull()==0){
        antrian.tail++;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    }
}
```

- Dequeue()

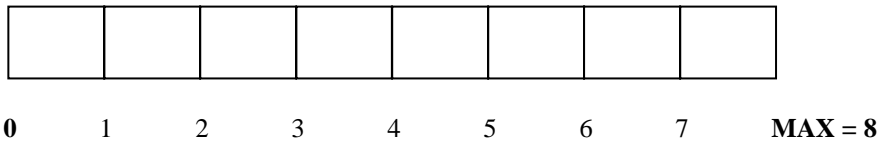
- o Digunakan untuk menghapus elemen terdepan/pertama dari Antrian
- o Dengan cara mengurangi counter Tail dan menggeser semua elemen antrian kedepan.
- o Penggeseran dilakukan dengan menggunakan looping



```
int Dequeue(){
    int i;
    int e = antrian.data[antrian.head];
    for(i=antrian.head;i<=antrian.tail-1;i++){
        antrian.data[i] = antrian.data[i+1];
    }
    antrian.tail--;
    return e;
}
```

- Clear()

- o Untuk menghapus elemen-elemen Antrian dengan cara membuat Tail dan Head = -1
- o Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen-elemen Antrian tidak lagi terbaca



Head = -1

Tail = -1

```
void Clear(){
    antrian.head=antrian.tail=-1;
    printf("data clear");
}
```

- Tampil()

- o Untuk menampilkan nilai-nilai elemen Antrian
- o Menggunakan looping dari head s/d tail

```
void Tampil(){
    if(IsEmpty()==0){
        for(int i=antrian.head;i<=antrian.tail;i++){
            printf("%d ",antrian.data[i]);
        }
    }else printf("data kosong!\n");
}
```

Program selengkapnya:

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

#define MAX 8
typedef struct{
    int data[MAX];
    int head;
    int tail;
} Queue;

Queue antrian;

void Create(){
    antrian.head=antrian.tail=-1;
}

int IsEmpty(){
    if(antrian.tail===-1)
        return 1;
    else
        return 0;
}

int IsFull(){
    if(antrian.tail==MAX-1) return 1;
    else return 0;
}

void Enqueue(int data){
    if(IsEmpty()==1){
        antrian.head=antrian.tail=0;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    }
}
```

```

    } else
    if(IsFull()==0){
        antrian.tail++;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    }
}

int Dequeue(){
    int i;
    int e = antrian.data[antrian.head];
    for(i=antrian.head;i<=antrian.tail-1;i++){
        antrian.data[i] = antrian.data[i+1];
    }
    antrian.tail--;
    return e;
}

void Clear(){
    antrian.head=antrian.tail=-1;
    printf("data clear");
}

void Tampil(){
    if(IsEmpty()==0){
        for(int i=antrian.head;i<=antrian.tail;i++){
            printf("%d ",antrian.data[i]);
        }
    }else printf("data kosong!\n");
}

void main(){
    int pil;
    int data;
    Create();
    do{
        clrscr();
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Tampil\n");
        printf("4. Clear\n");
        printf("5. Exit\n");
        printf("Pilihan = ");scanf("%d",&pil);
        switch(pil){
            case 1: printf("Data = ");scanf("%d",&data);
                    Enqueue(data);
                    break;
            case 2: printf("Elemen yang keluar : %d",Dequeue());
                    break;

```

```
        case 3:  Tampil();
                break;
        case 4:  Clear();
                break;
    }
    getch();
} while(pil!=5);
}
```