

PERTEMUAN 11

Stack dan Queue

Hermanto, S.Kom. M.Kom

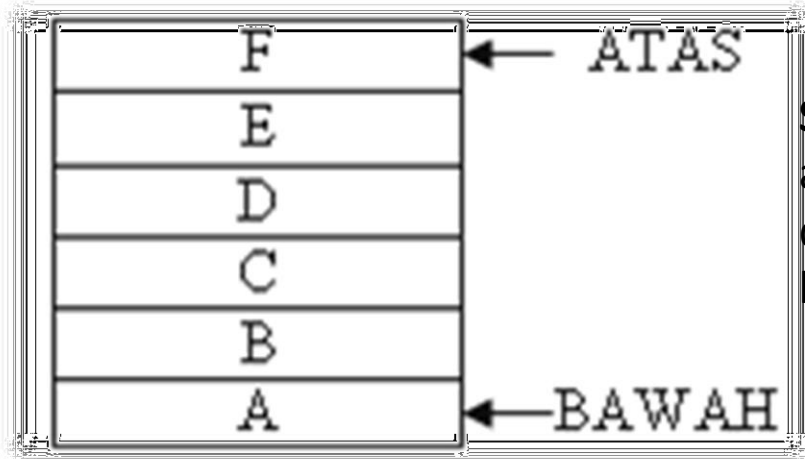


Pendahuluan

- ❑ Stack merupakan metode dalam menyimpan atau mengambil data ke dan dari memori. Secara sederhana, tumpukan bisa diartikan sebagai suatu kumpulan data yang seolah-olah ada data yang diletakan diatas data yang lain. Satu hal yang perlu kita ingat adalah bahwa kita bisa menambah data, dan mengambil (menghapus) data lewat ujung yang sama, yang disebut sebagai ujung atas tumpukan (top of stack).
- ❑ Untuk menjelaskan pengertian diatas kita ambil contoh sebagai berikut. Misalnya kita mempunyai dua buah kotak yang kita tumpuk, sehingga kotak kita letakkan diatas kotak yang lain. Jika kemudian tumpukan duah buah kotak itu kita tambah dengan kotak ketiga, keempat dan seterusnya, maka akan kita peroleh sebuah tumpukan kotak yang terdiri dari N kotak.
- ❑ Prinsip Stack adalah **First In Last Out (FILO)** atau **Last In First Out (LIFO)**.

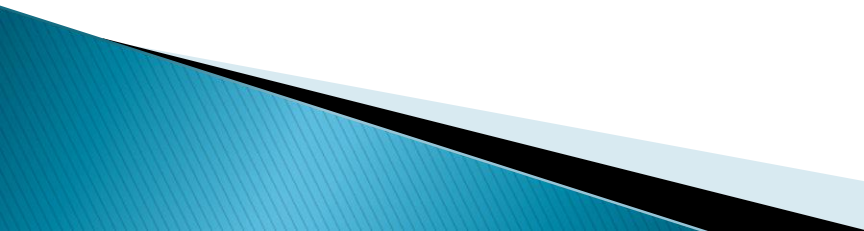
Stack is?

Stack pada **Struktur Data** sebagai tumpukan dari benda, sekumpulan data yang seolah-olah diletakkan di atas data yang lain, Suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (top) saja.

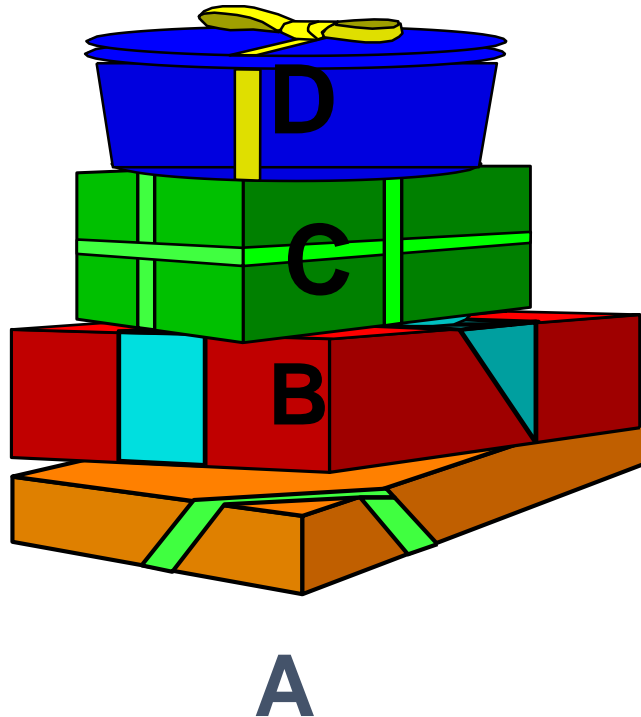


Stack bersifat LIFO (*Last In First Out*) artinya Benda yang terakhir masuk ke dalam stack akan menjadi yang pertama keluar dari stack.

Operasi-operasi pada Stack

- 1. Push:** digunakan untuk menambah item pada stack pada tumpukan paling atas
 - 2. Pop :** digunakan untuk mengambil item pada stack pada tumpukan paling atas
 - 3. Clear** : digunakan untuk mengosongkan stack
 - 4. IsEmpty :** fungsi yang digunakan untuk mengecek apakah stack sudah kosong
 - 5. IsFull :** fungsi yang digunakan untuk mengecek apakah stack sudah penuh
 - 6. Deklarasi :** proses pendeklarasian stack.
 - 7. Inisialisasi :** proses pembuatan stack kosong, biasanya dengan pemberian nilai untuk top.
- 

Mengenal Struktur Data Stack



Dari gambar ini kita bisa mengatakan bahwa kotak B ada diatas kotak A dan ada dibawah kotak C. Gambar dibawah ini hanya menunjukkan bahwa dalam tumpukan kita hanya bisa menambah atau mengambil sebuah kotak lewat satu ujung, yaitu ujung bagian atas

Contoh lain adalah ada sekumpulan perintah stack yaitu `push(5)`, `push(7)`, `pop`, `push(3)`, `pop`. Jika dijalankan, maka bagaimana proses operasi stack?

Representasi stack

Representasi stack dalam pemrograman, dapat dilakukan dengan 2 cara yaitu :

1. Representasi stack dengan array
2. Representasi stack dengan single linked list

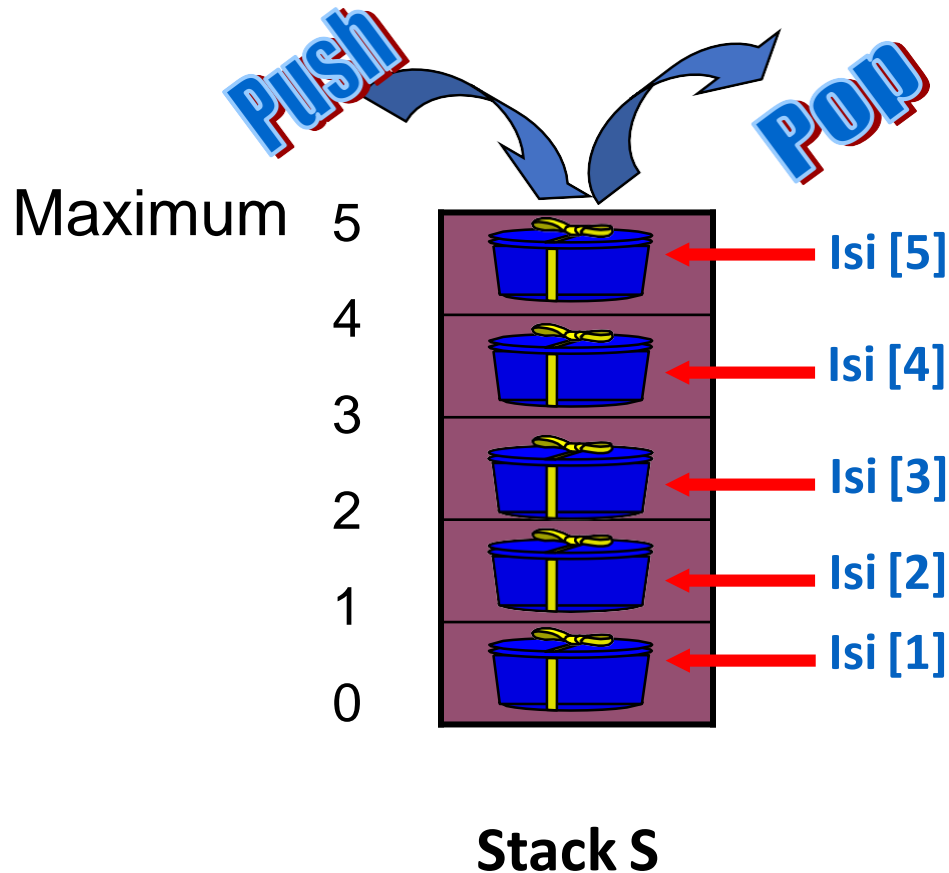
Sebagai contoh representasi kedua cara tersebut dengan operasi yang dilakukan adalah push(1), push(2), pop, push(5), push(8), pop. Untuk lebih detail, perhatikan gambar di bawah ini :

Representasi stack dengan menggunakan array dengan maksimal data 5 adalah

<div><div>?</div><div>?</div><div>?</div><div>?</div><div>?</div></div> <div>Top=0 Maks=5</div>	<div><div>?</div><div>?</div><div>?</div><div>?</div><div>1</div></div> <div>Top=1 Maks=5</div>	<div><div>?</div><div>?</div><div>?</div><div>2</div><div>1</div></div> <div>Top=2 Maks=5</div>	<div><div>?</div><div>?</div><div>?</div><div>2</div><div>1</div></div> <div>Top=3 Maks=5</div>	<div><div>?</div><div>?</div><div>?</div><div>5</div><div>1</div></div> <div>Top=2 Maks=5</div>	<div><div>?</div><div>?</div><div>8</div><div>5</div><div>1</div></div> <div>Top=3 Maks=5</div>	<div><div>?</div><div>?</div><div>8</div><div>5</div><div>1</div></div> <div>Top=2 Maks=5</div>
Kondisi awal	Push(1)	Push(2)	Pop	Push(5)	Push(8)	Pop

Elemen berisi ? berarti nilai elemen tidak diketahui.

Mengenal Struktur Data Stack



Deklarasi Struktur Data

Stack = Record

Isi : array[1..n] of Tipe Data

Atas : integer

End

Pendeklarasian stack

Proses pendeklarasian stack adalah proses pembuatan struktur stack dalam memori.

Pendeklarasian stack yang menggunakan array.

Suatu stack memiliki beberapa bagian yaitu :

- ❑ top yang menunjuk posisi data terakhir (top)
- ❑ elemen yang berisi data yang ada dalam stack. Bagian ini lah yang berbentuk array.
- ❑ maks_elemen yaitu variable yang menunjuk maksimal banyaknya elemen dalam stack.

```
Const max = 100; Type tstack = record  
    top = integer;  
    maks_elemen = integer;  
    elemen = array[1..max] of elemen;  
var stack = tstack;
```


Inisialisasi

Inisialisasi stack adalah proses pembuatan suatu stack kosong. Adapun langkah-langkah proses tersebut berdasarkan jenis penyimpanannya adalah :

Inisialisasi stack yang menggunakan array.

Proses inisialisasi untuk stack yang menggunakan array adalah dengan mengisi nilai field top dengan 0 (nol) jika elemen pertama diawali dengan nomor 1.

```
procedure inisialisasi(var stack : tstack);  
begin stack.top = 0;  
      stack.maks_elemen = max;  
end;
```

Operasi Push

Operasi Push berguna untuk menambah suatu elemen data baru pada stack dan disimpan pada posisi top yang akan mengakibatkan posisi top akan berubah. Penambahan dapat dilakukan jika stack itu belum penuh.

Stack dikatakan penuh Jika posisi atas sudah berada pada posisi N

(If $S.atas = n$ then stack penuh)

Push(x, S) adalah Memasukkan x kedalam Stack S

Langkah operasi push dalam array adalah dengan :

- ☐ Periksa apakah stack penuh (isfull). Jika bernilai false/0 (tidak penuh) maka proses push dilaksanakan dan jika pemeriksaan ini bernilai true/1 (stack penuh), maka proses push digagalkan.
- ☐ Proses push-nya sendiri adalah dengan menambah field top dengan 1, kemudian elemen pada posisi top diisi dengan elemen data baru.

Push(x,s)

Procedure Push(xbaru : Tipe data, S : Stack)

If $S.atas < n$ then

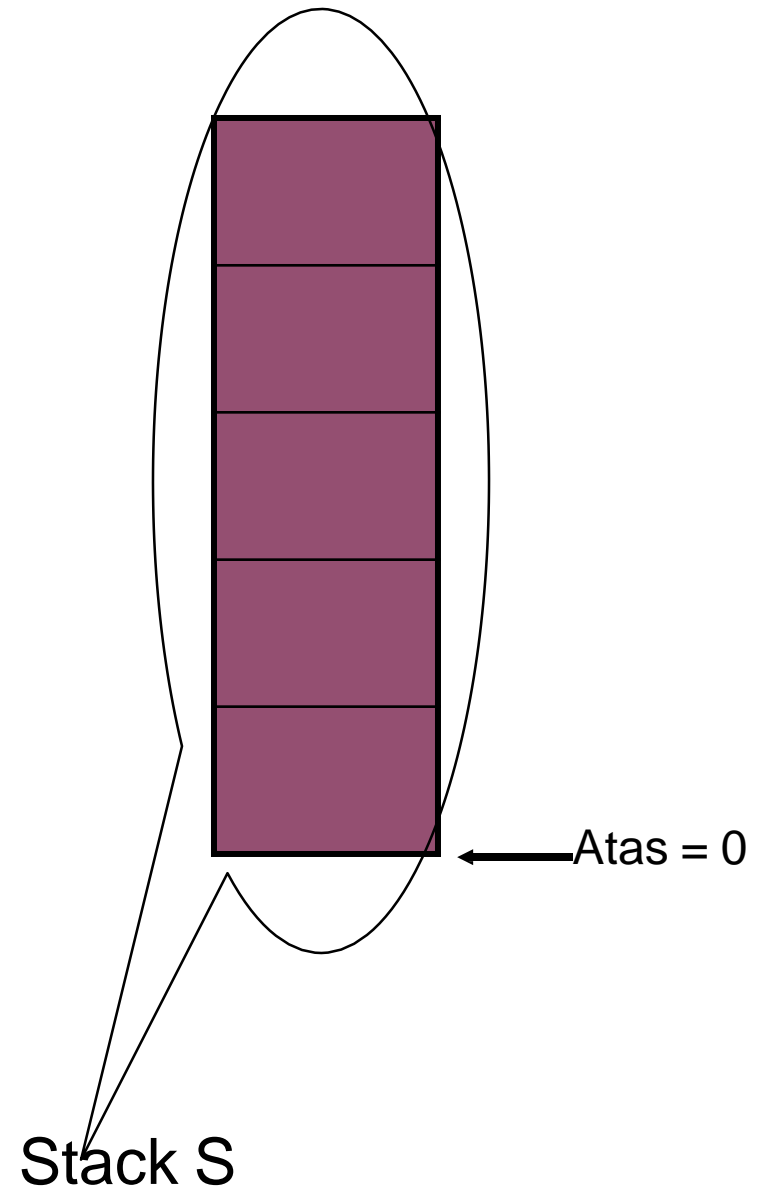
$S.atas = S.atas + 1$

$S.isi[S.atas] = xbaru$

Else

 stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

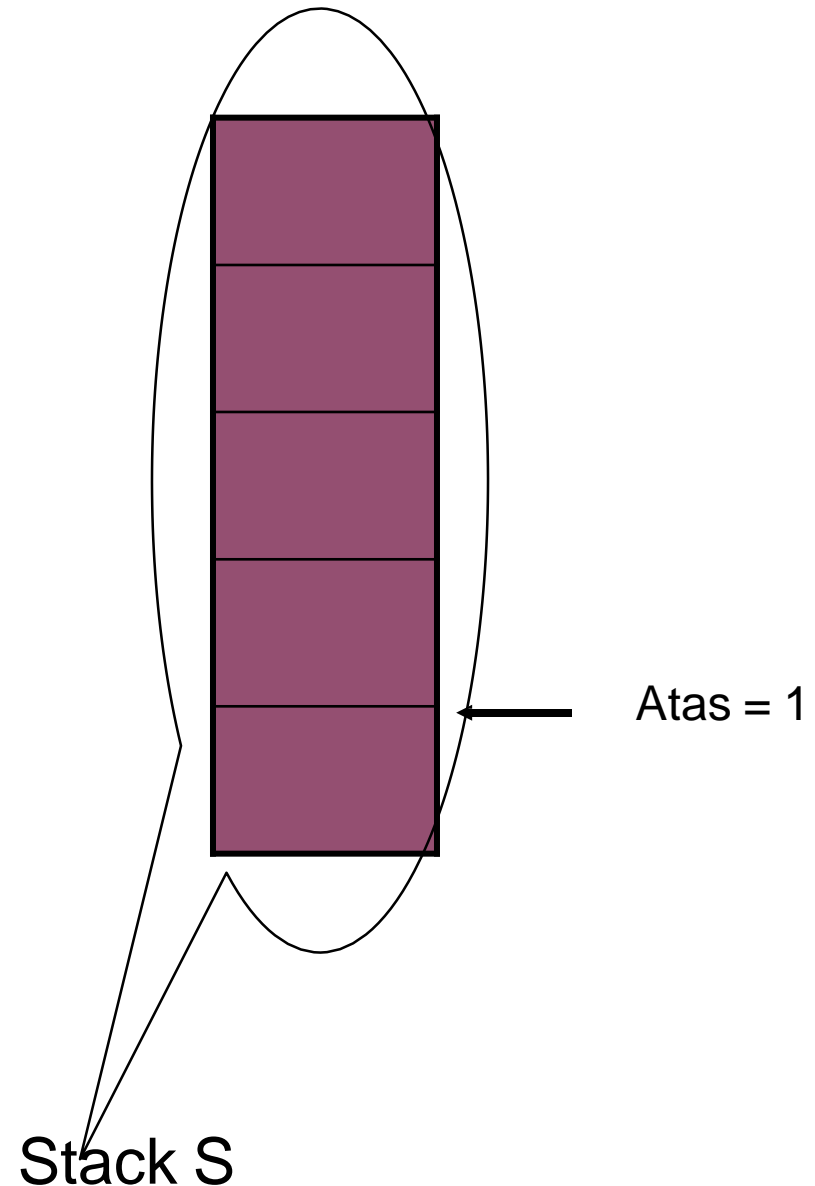
 S.atas = S.atas + 1

 S.isi[S.atas] = xbaru

Else

 stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

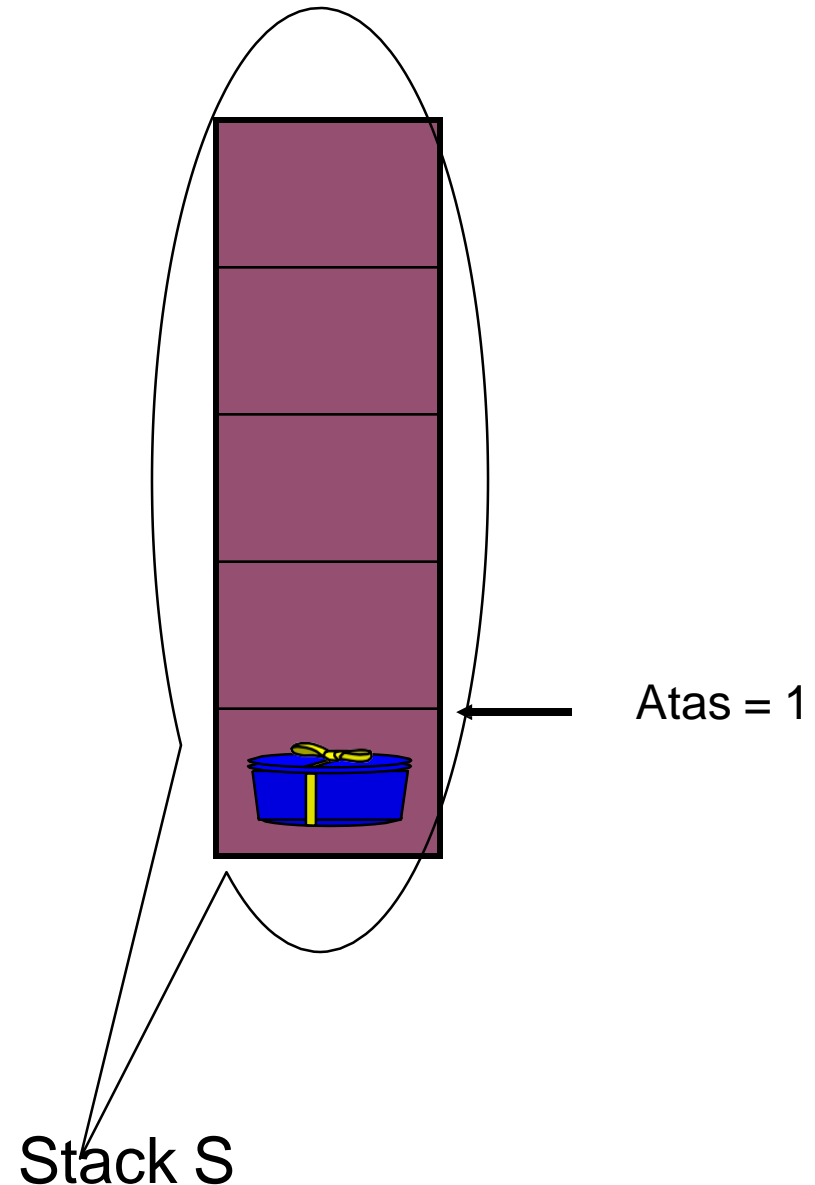
 S.atas = S.atas + 1

 S.isi[S.atas] = xbaru

Else

 stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

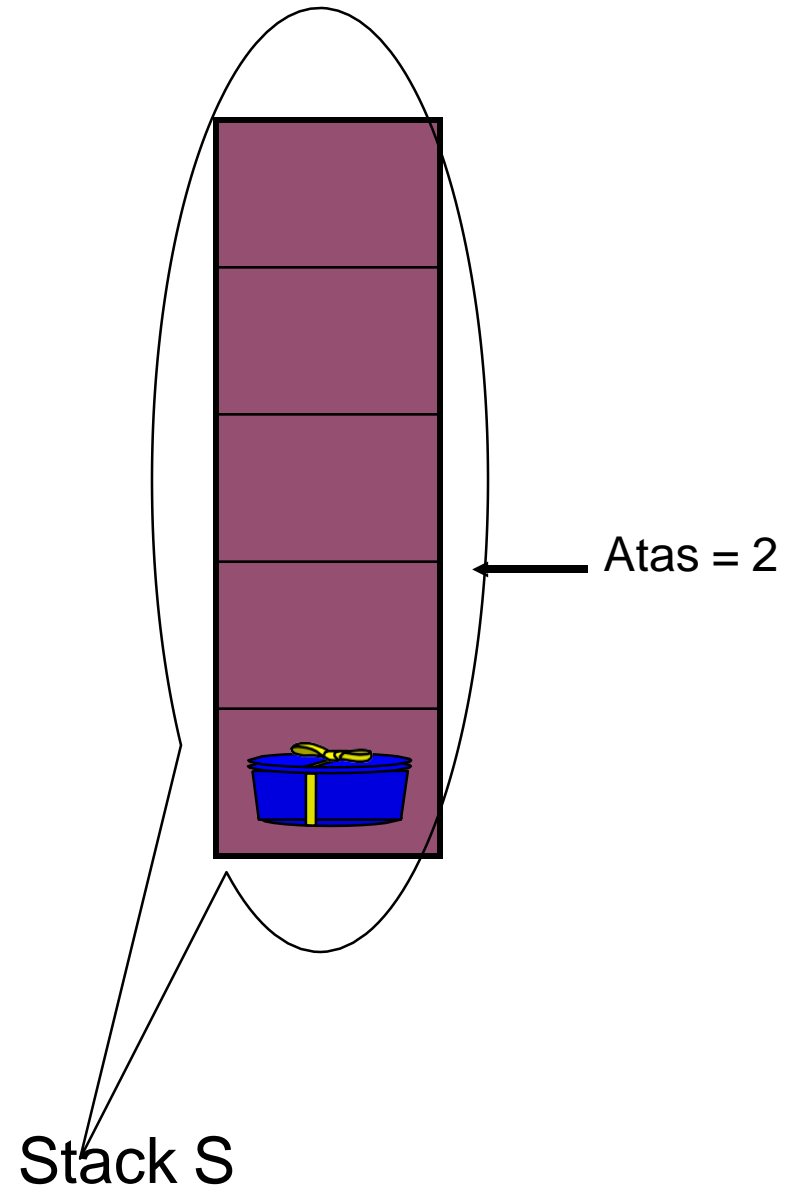
S.atas = S.atas + 1

S.isi[S.atas] = xbaru

Else

stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

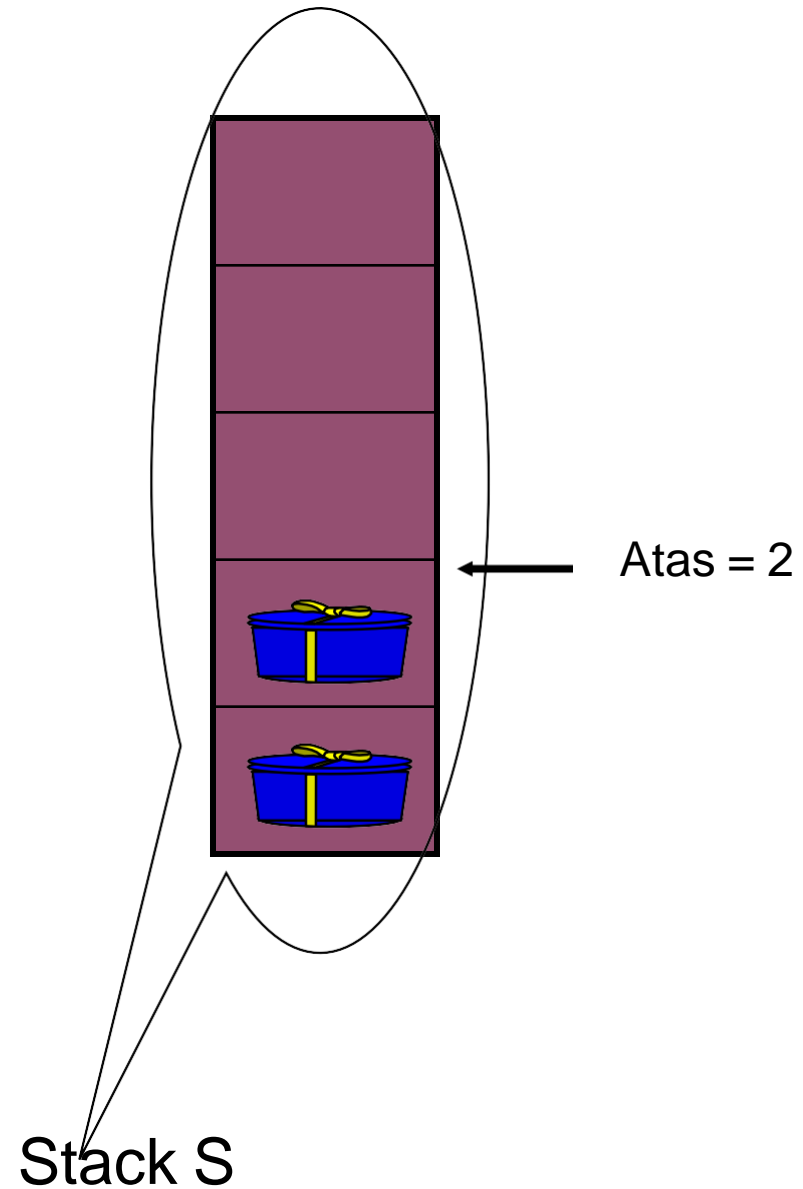
 S.atas = S.atas + 1

 S.isi[S.atas] = xbaru

Else

 stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

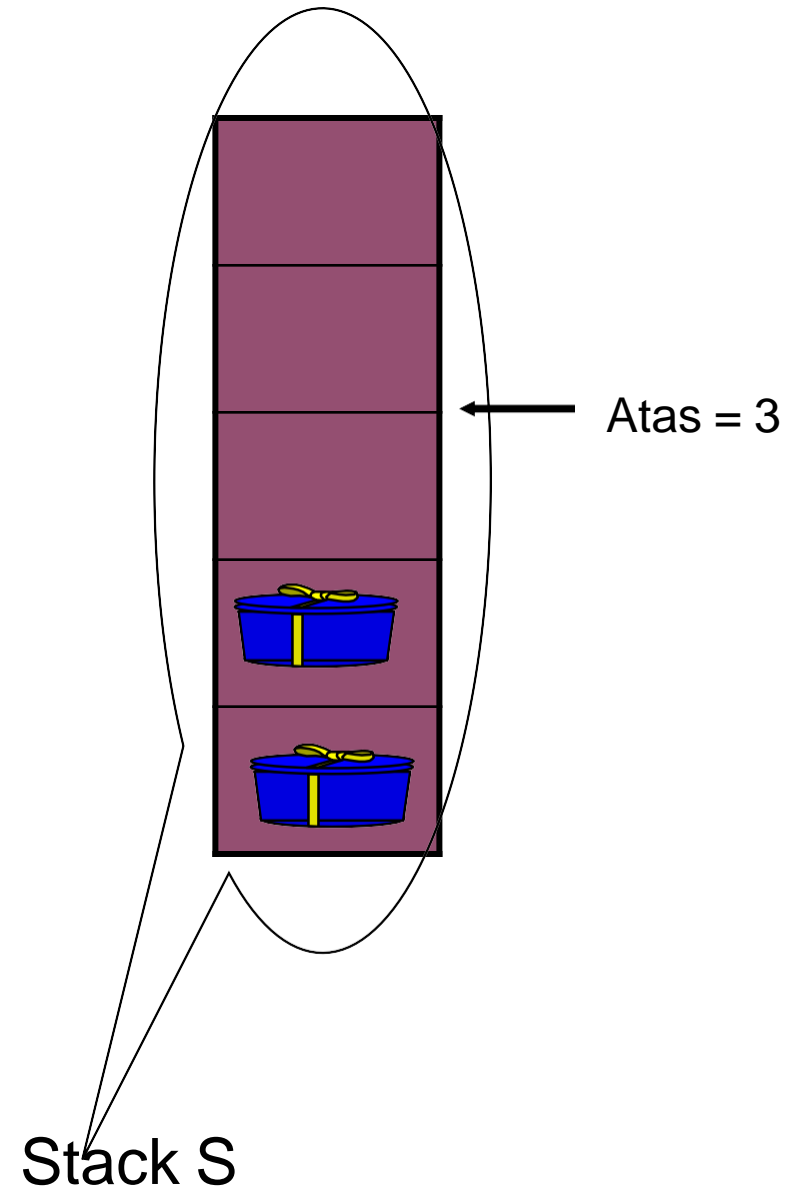
S.atas = S.atas + 1

S.isi[S.atas] = xbaru

Else

stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

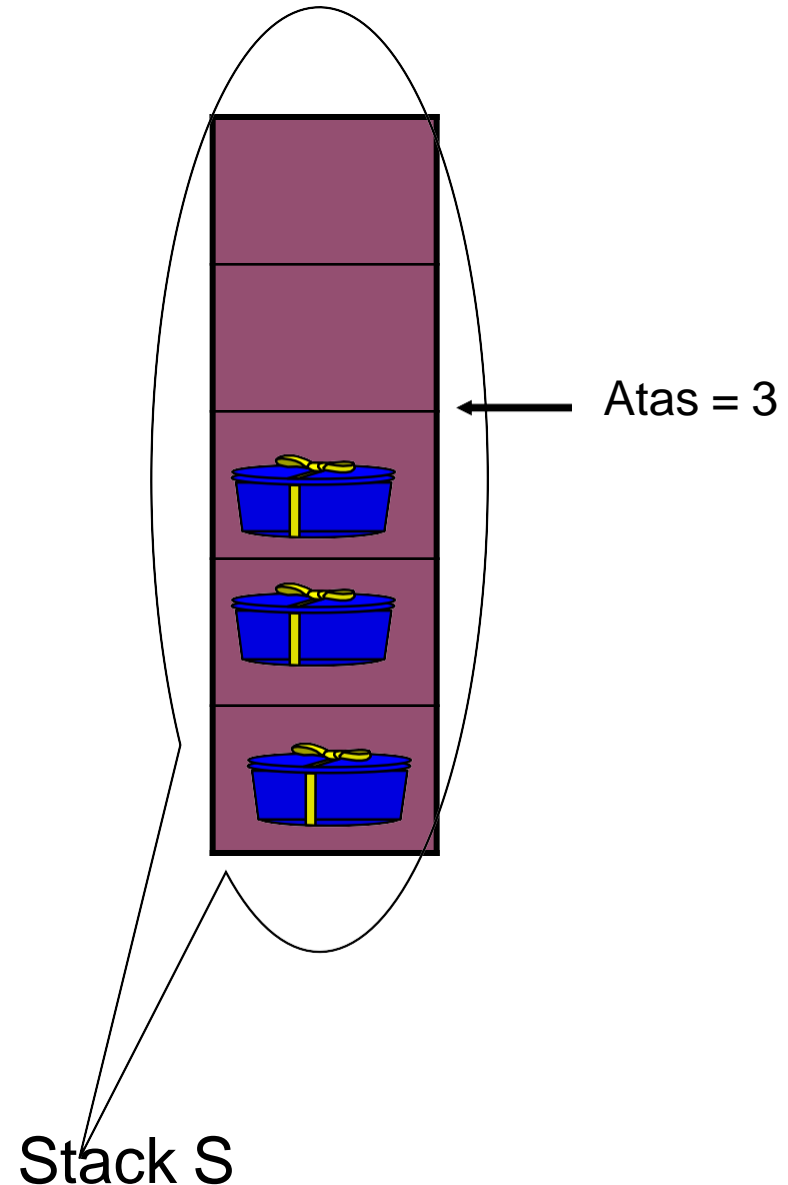
 S.atas = S.atas + 1

 S.isi[S.atas] = xbaru

Else

 stack sudah penuh

End if



Push(x,s)

Procedure Push(xbaru :Tipe data, S : Stack)

If S.atas < n then

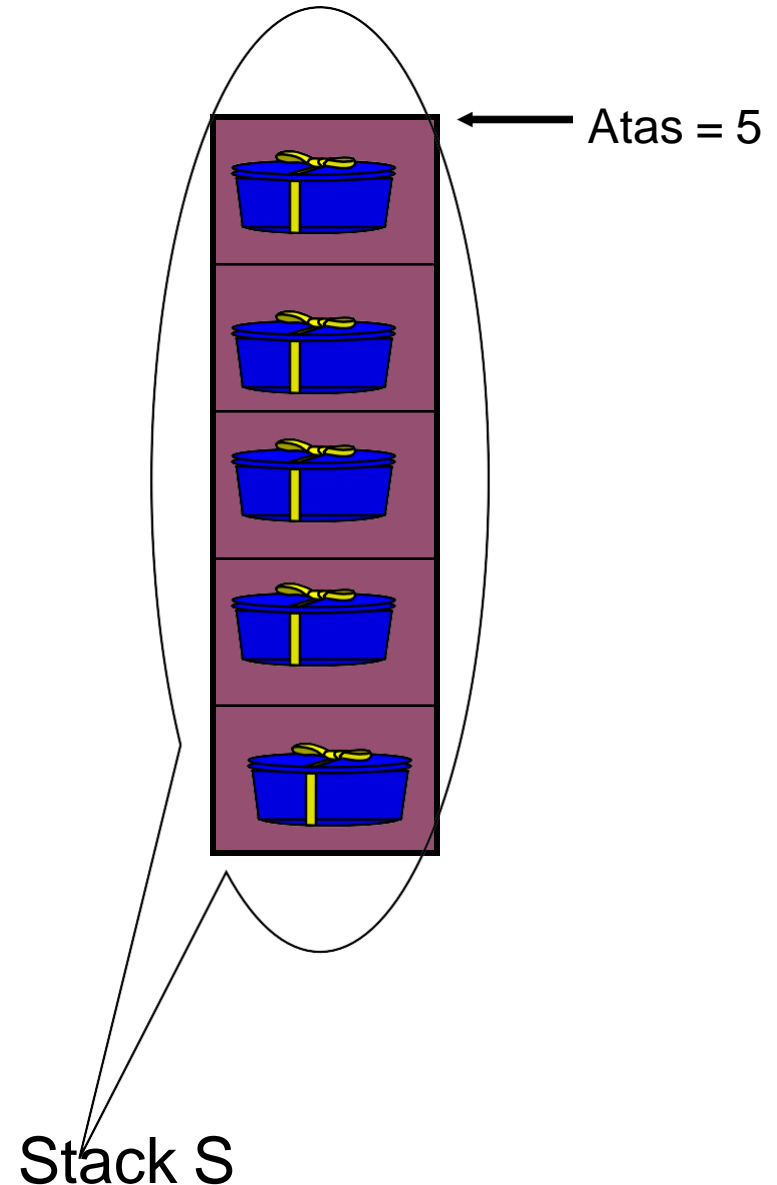
S.atas = S.atas + 1

S.isi[S.atas] = xbaru

Else

stack sudah penuh

End if



Pop(S)

Pop(s) berguna untuk mengambil elemen terakhir (top) dan kemudian menghapus elemen tersebut sehingga posisi top akan berpindah (LIFO Last In First Out). Proses penghapusan dapat dilakukan jika stack tidak dalam keadaan Kosong.

If $S.Atas > 0$ then stack tidak kosong

Dimana Setiap melakukan penghapusan, maka posisi yang paling atas akan berkurang 1 ($S.Atas = S.Atas - 1$)

Procedure Pop(S: Stack)

If $S.atas > 0$ then

$Pop = S.isi[S.atas];$

$S.Atas = S.Atas - 1$

Else

Stack Kosong

End if

Pop(S)

Procedure Pop(S: Stack)

If S.atas>0 then

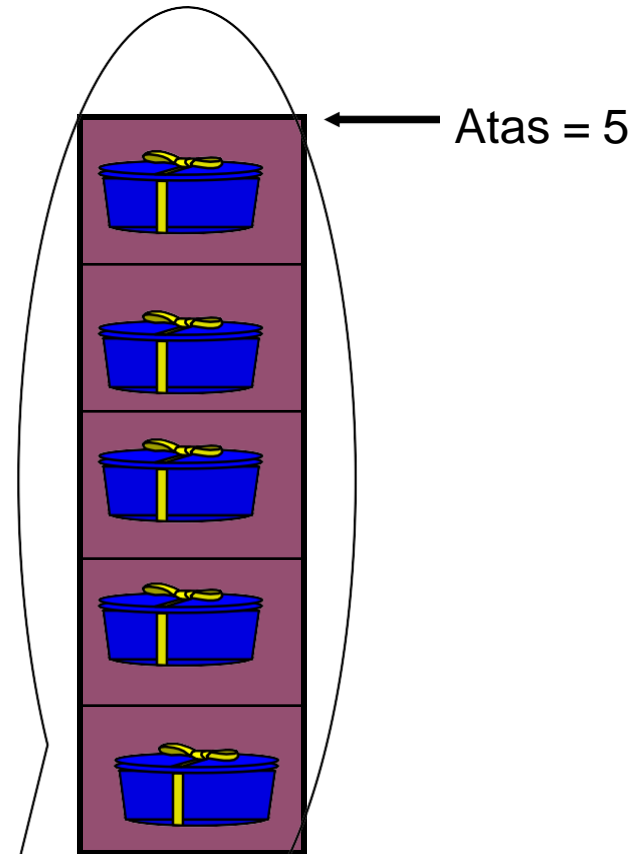
 Pop = S.isi[S.atas];

 S.atas = S.atas - 1

Else

Stack Kosong

End if



Stack S

Pop(S)

Procedure Pop(S: Stack)

If $S.atas > 0$ then

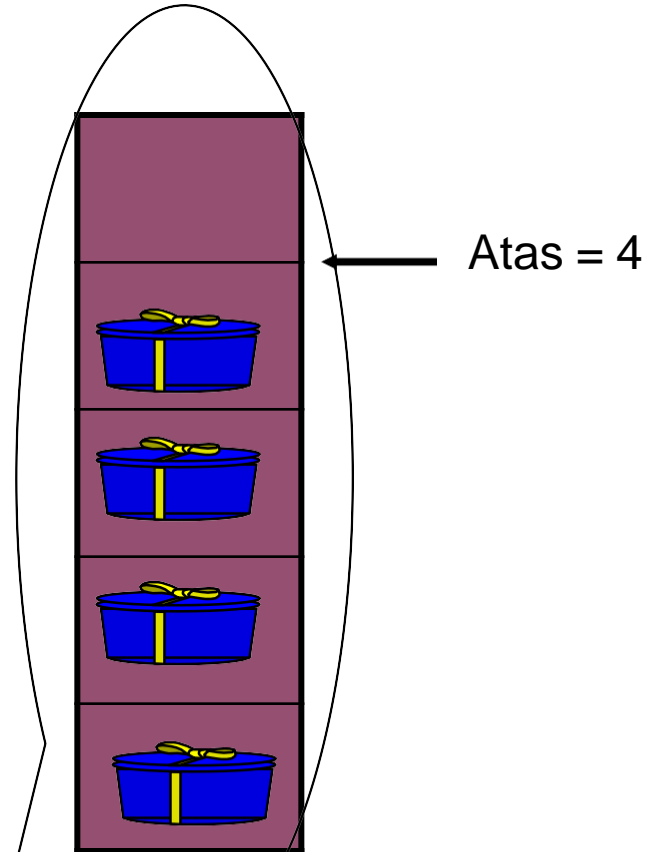
Pop = $S.isi[S.atas]$;

$S.atas = S.atas - 1$

Else

Stack Kosong

End if



Stack S

Pop(S)

Procedure Pop(S: Stack)

If S.atas>0 then

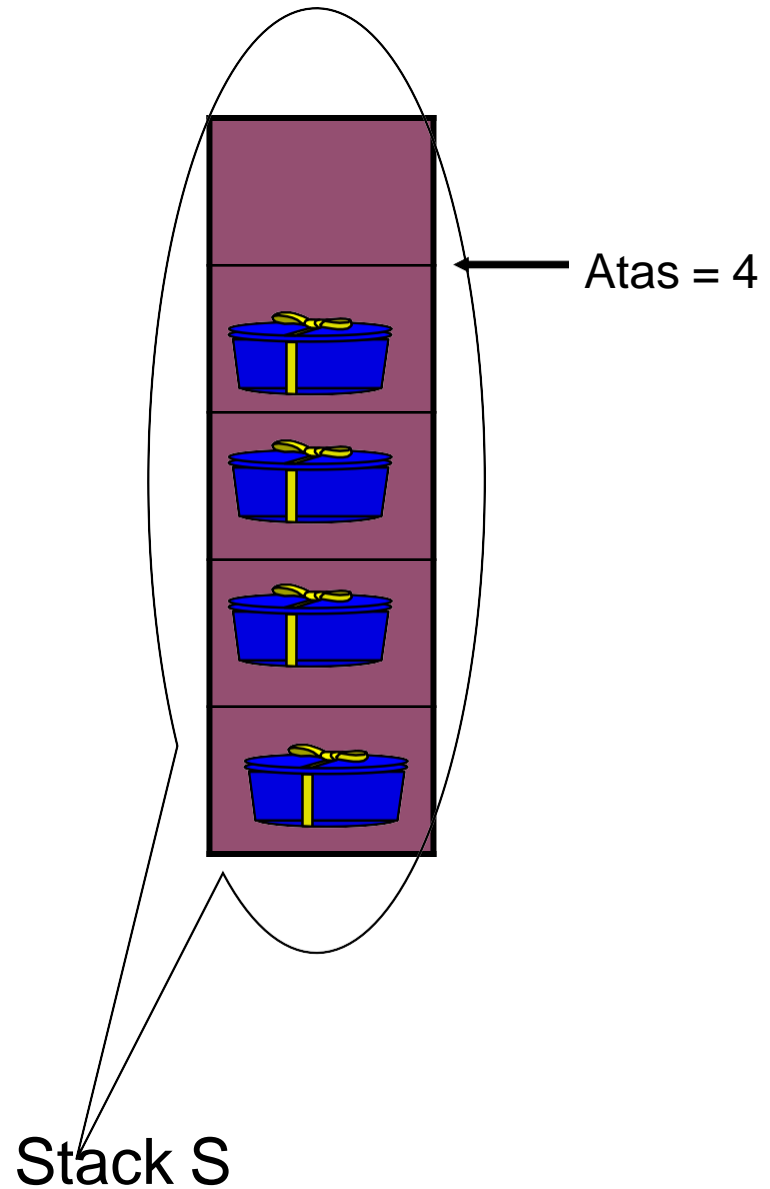
 Pop = S.isi[S.atas];

 S.atas= S.atas – 1

Else

Stack Kosong

End if



Pop(S)

Procedure Pop(S: Stack)

If S.atas>0 then

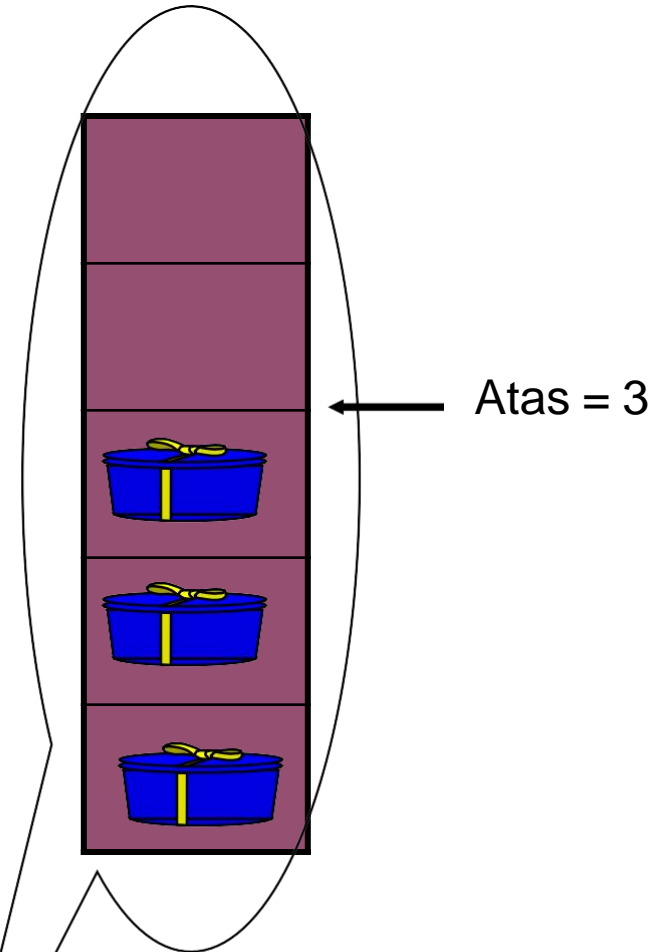
Pop = S.isi[S.atas];

S.atas = S.atas - 1

Else

Stack Kosong

End if



Stack S

Pop(S)

Procedure Pop(S: Stack)

If $S.atas > 0$ then

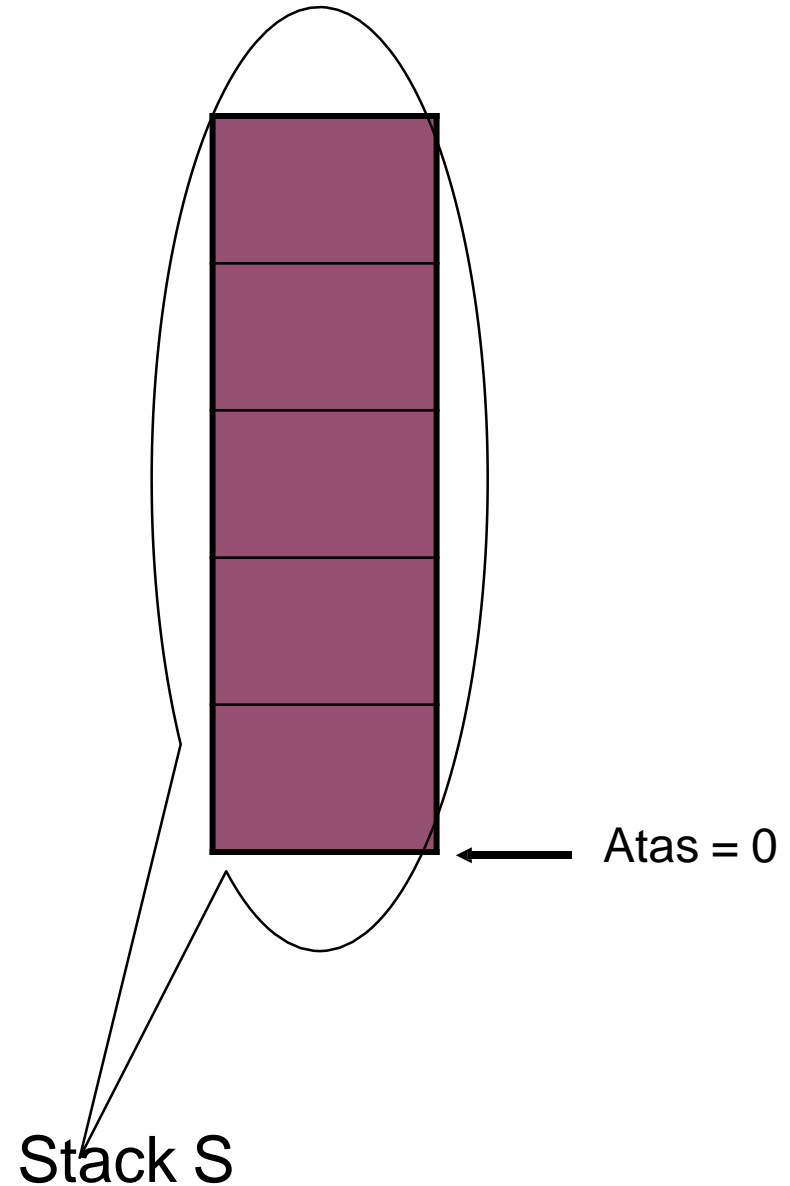
$Pop = S.isi[S.atas];$

$S.atas = S.atas - 1$

Else

 Stack Kosong

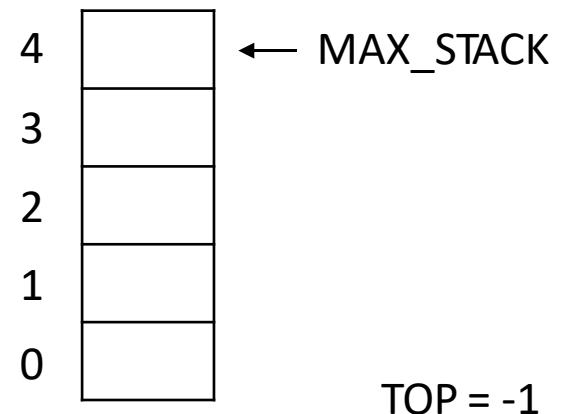
End if



Fungsi IsEmpty

- Digunakan untuk memeriksa apakah stack masih dalam kondisi kosong.
- Dengan cara memeriksa TOP of STACK Jika TOP masih = -1 maka berarti stack masih kosong.

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



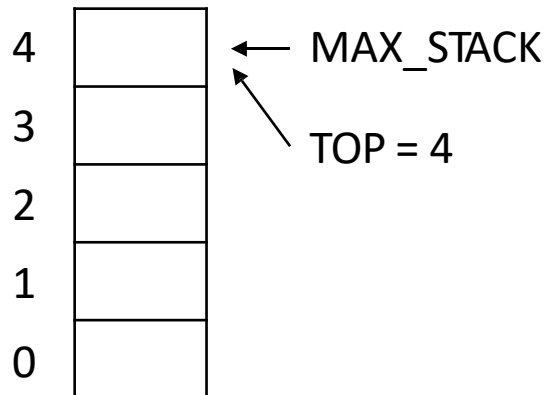
Fungsi IsFull

- Digunakan untuk memeriksa apakah stack dalam kondisi sudah penuh.
- Dengan cara memeriksa TOP of STACK

Jika TOP of STACK = MAX_STACK-1 maka FULL (Penuh).

Jika TOP of STACK < MAX_STACK-1 maka Belum Penuh.

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

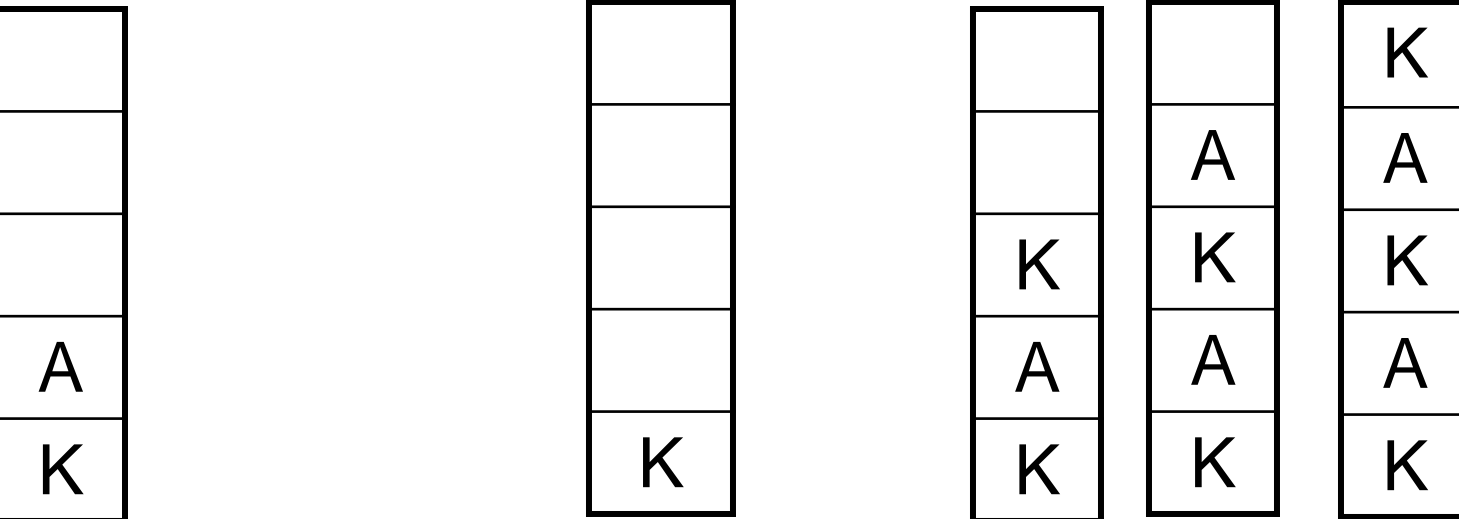


Contoh Penggunaan Stack

1. Untuk mengecek kalimat Polindrom
2. Untuk Mengubah Desimal ke Biner

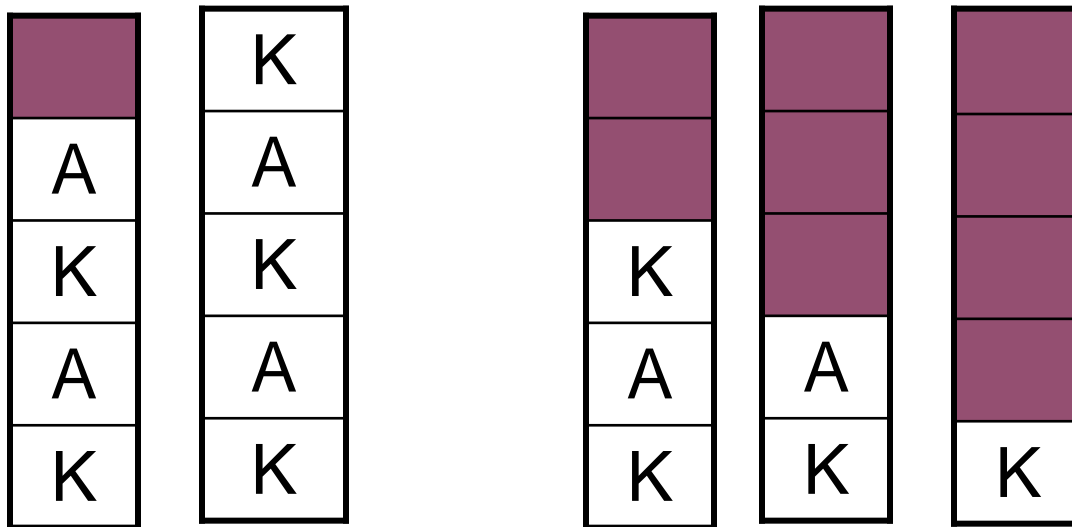
Mengecek Kalimat Palindrom

Kalimat : KAKAK



Operasi Push

Mengecek Kalimat Palindrom



Hasil = "

Hasil = K

Hasil = KA

Hasil = KAK

Hasil = KAKA

Hasil = KAKAK

Operasi POP

Kalimat = hasil

➔ Polindrom

Algoritma

Inisialisasi Struktur Data

Stack = record

isi : Array[1..255] of char

atas : integer

End

Kalimat, hasil : string

Procedure push(xbaru : Char, s : Stack)

If s.atas < 255

Then

s.atas = s.atas+1

s.isi[s.atas] = xbaru

Else

stack sudah penuh

End if

Algoritma

```
Procedure Pop(S:Stack)
```

```
  If S.atas>0 then
```

```
    cout s.isi[s.atas]
```

```
    hasil = hasil +s.isi[s.atas]
```

```
    s.atas= s.atas-1
```

```
  Else
```

```
    Stack Kosong
```

```
  End if
```

```
  //modul utama
```

```
  i=1
```

```
  While i<= length(kalimat)
```

```
  Do
```

```
    Push(Kalimat[i],s)
```

```
    i=i+1
```

```
While S.atas>0 do
```

```
  pop(s)
```

```
  If kalimat = hasil
```

```
  Then
```

```
    Polindrom
```

```
  Else
```

```
    Tidak polindrom
```

```
  End if
```


QUEUE (ANTRIAN)

Antrian (Queue) merupakan kumpulan data yang mana penambahan elemen hanya bias dilakukan pada suatu ujung yaitu rear/tail/belakang, dan penghapusan dilakukan melalui ujung yang lainnya yaitu front/head/depan. Antrian disebut FIFO (First In First Out) yaitu elemen yang lebih dulu disisipkan merupakan elemen yang akan lebih dulu diambil.

Operasi-operasi dasar dari sebuah queue adalah :

1. Enqueue : proses penambahan elemen di posisi belakang
2. Dequeue : proses pengambilan elemen di posisi depan

Selain operasi dasar di atas, ada pula operasi-operasi lain yang dapat dilakukan terhadap sebuah queue yaitu :

1. Operasi pemeriksaan queue kosong (fungsi kosong)
 2. Operasi pemeriksaan queue penuh (fungsi penuh).
 3. Operasi inisialisasi queue (fungsi inisialisasi)
- 

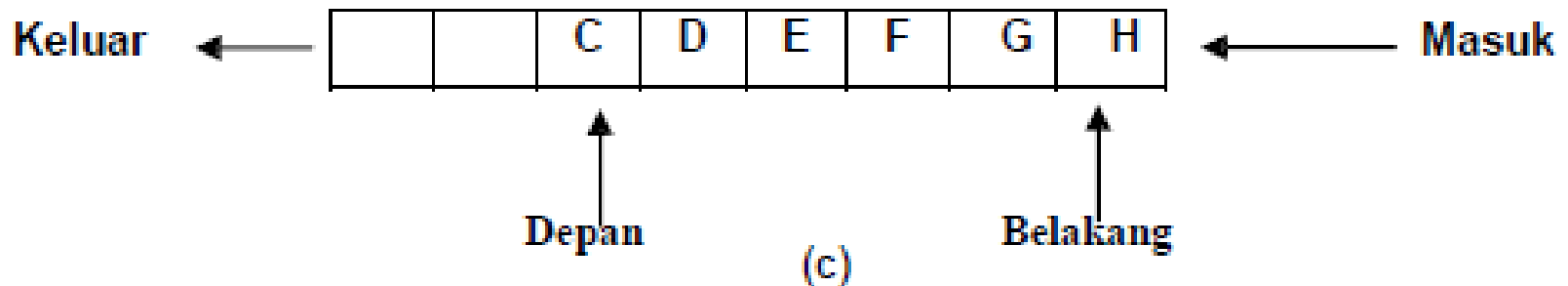
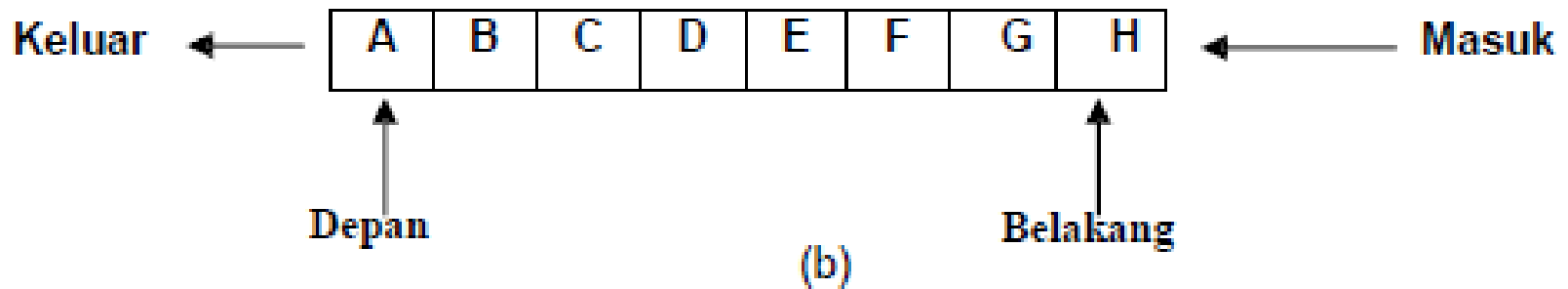
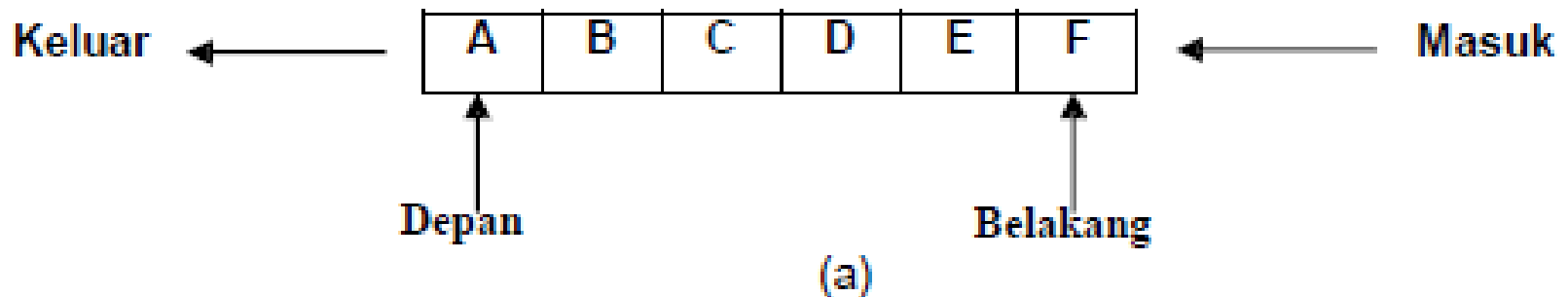
Karakteristik Antrian

Karakteristik antrian sebagai berikut :

1. Elemen antrian yaitu item-item data yang terdapat di elemen antrian
2. Front (elemen terdepan dari antrian)
3. Tail (elemen terakhir dari antrian)
4. Count (jumlah elemen pada antrian)
5. Status antrian apakah penuh atau kosong.
 - ☐ Penuh, jika elemen pada antrian mencapai kapasitas maximum antrian. Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian.
 - ☐ Kosong, jika tidak ada elemen pada antrian. Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen dari antrian.

Implementasi Antrian Array Statis

Jika ada elemen baru yang akan masuk pada gambar (a), maka ia akan diletakkan di sebelah kanan F (gambar (b)). Jika ada elemen yang akan dihapus, maka A akan dihapus lebih dulu (gambar (c)).



Implementasi Antrian Array Statis

```
Const max = 5  
Var antrian : array [1..max] of char;  
    belakang, depan : integer; xbaru : char;
```

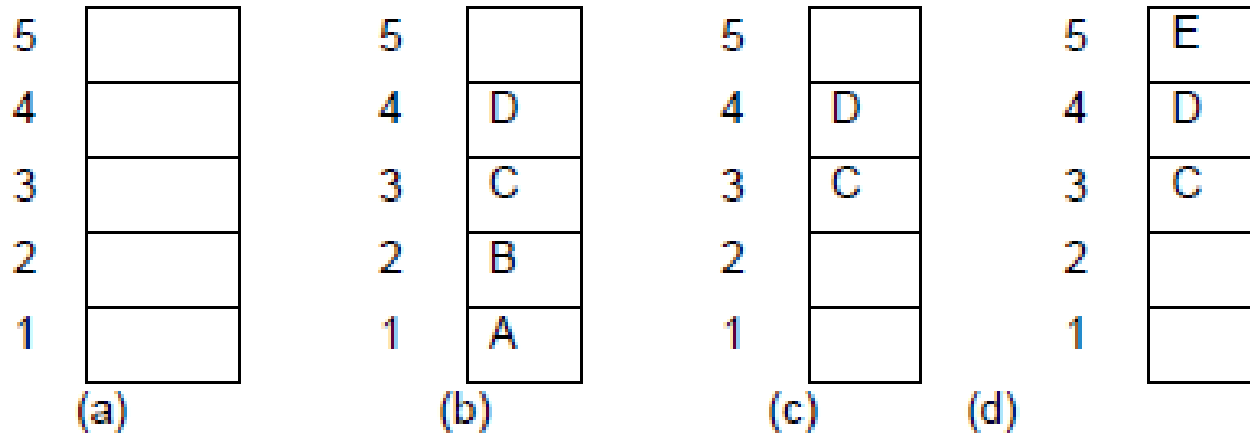
Dengan menggunakan array, maka overflow dapat terjadi jika antrian telah penuh , sementara masih ingin menambah elemen ke dalam antrian. Dengan mengabaikan adanya overflow, maka penambahan elemen baru yang dinyatakan oleh var xbaru dapat diimplementasikan dengan statemen :

```
belakang = belakang + 1;  
antrian[belakang] = xbaru;
```

Operasi penghapusan bisa diimplementasikan dengan ;

```
xbaru = antrian[depan];  
depan = depan + 1;
```

Implementasi Antrian Array Statis

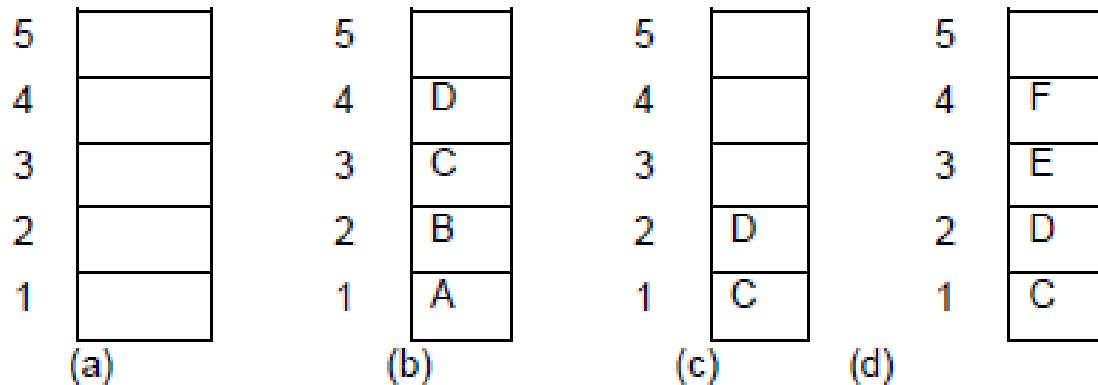


Pada gambar (a) posisi depan = 1 dan belakang = 0. Pada gambar (b) keadaan setelah penambahan empat buah elemen dimana posisi depan = 1 dan belakang = 4. Pada gambar (c) keadaan setelah penghapusan dua buah elemen dimana posisi depan = 3 dan belakang = 4. Pada gambar (d) keadaan setelah penambahan satu buah elemen dimana posisi depan = 3 dan belakang = 5.

Jika akan ditambah elemen baru, maka nilai belakang harus ditambah satu menjadi 6. Sedangkan larik antrian tersebut hanya terdiri dari 6 elemen sehingga tidak bisa ditambah lagi meskipun sebenarnya larik tersebut masih kosong di dua tempat. Oleh karena itu dilakukan dengan metoda penggeseran dimana jika ada elemen yang dihapus, maka semua elemen lain digeser sehingga antrian selalu dimulai dari depan = 1.

Implementasi Antrian Array Statis

```
x = antrian [1];  
for i = 1 to belakang-1 do  
begin  
    antrian[i] = antrian [i +1];  
end;
```



Pada gambar (a) posisi depan = 1 dan belakang = 0. Pada gambar (b) keadaan setelah penambahan empat buah elemen dimana posisi depan = 1 dan belakang = 4. Pada gambar (c) keadaan setelah penghapusan dua buah elemen dimana posisi depan = 1 dan belakang = 2. Pada gambar (d) keadaan setelah penambahan dua buah elemen dimana posisi depan = 1 dan belakang = 4.

Queue Dengan Circular Array

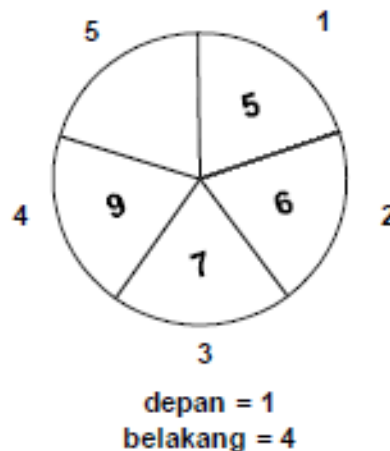
Jika menggunakan array untuk queue seperti di atas, maka ketika ada proses pengambilan (dequeue) ada proses pergeseran data. Proses pergeseran data ini pasti memerlukan waktu apalagi jika elemen queue-nya banyak. Oleh karena itu solusi agar proses pergeseran dihilangkan adalah dengan metode circular array. Queue dengan circular array dapat dibayangkan sebagai berikut :

Depan=1

Belakang=4



Atau agar lebih jelas, array di atas dapat dibayangkan ke dalam bentuk seperti lingkaran di bawah ini.



Aturan queue menggunakan circular array

1. Proses penghapusan dilakukan dengan cara nilai depan (front) ditambah 1 : $\text{depan} = \text{depan} + 1$.
2. Proses penambahan elemen sama dengan linear array yaitu nilai belakang
 - ▶ ditambah 1 : $\text{belakang} = \text{belakang} + 1$.
3. Jika $\text{depan} = \text{maks}$ dan ada elemen yang akan dihapus, maka $\text{nilai depan} = 1$.
4. Jika $\text{belakang} = \text{maks}$ dan $\text{depan} \neq 1$ maka jika ada elemen yang akan ditambahkan, $\text{nilai belakang} = 1$.
5. Jika hanya tinggal 1 elemen di queue ($\text{depan} = \text{belakang}$) dan akan dihapus, maka $\text{depan} = 0$.

Aturan queue menggunakan circular array

Front dan Tail akan bergerak maju, jika ;

1. Untuk penambahan.
2. Tail sudah mencapai elemen terakhir array akan memakai elemen pertama array yang telah dihapus.
3. Untuk penghapusan.
4. Front telah mencapai elemen terakhir array, maka akan menuju elemen pertama jika antrian masih berisi elemen.

Keunggulan representasi circular adalah seluruh kapasitas antrian bisa terpakai seluruhnya. Berdasarkan ilustrasi sebelumnya dapat disusun prosedur untuk menambah dan menghapus elemen antrian sebagai berikut ini :

```
Const max_elemen = 5;  
Type antri = array [1..max_elemen] of char;  
Var antrian : antri;  
    depan, belakang : integer;
```

```
Nilai awal untuk depan dan belakang masing-masing 0.  
    Depan := 0; Belakang := 0;
```


Aturan queue menggunakan circular array

Proses enqueue data hanya bisa dilakukan jika queue tidak kosong. Ada beberapa hal yang harus diperhatikan ketika akan melakukan enqueue pada circular array, diantaranya adalah :

- ❑ Kondisi ketika queue masih kosong. Jika ini terjadi, maka posisi depan dan belakang bernilai 0.
- ❑ Ketika ketika nilai belakang sama dengan maks_queue, maka posisi belakang adalah 0. Ini terjadi ketika posisi depan lebih besar dari 0 (pernah ada dequeue).
- ❑ Ketika nilai belakang masih lebih kecil dari maks_queue maka posisi belakang ditambah 1 : **belakang=belakang+1;**

Aturan queue menggunakan circular array

Dari ketentuan-ketentuan di atas dapat diimplementasikan operasi Enqueue-nya :

```
procedure Enqueue (var q : antri; x : char);
begin
  if belakang = max_elemen then belakang := 1
  else belakang := belakang + 1;
  if depan = belakang then
    begin
      cout (' antrian sudah penuh');
      belakang := belakang - 1;
      if belakang = 0 then
        belakang := max_elemen;
      end
    end
  else q[belakang] := x;
end;
```

Aturan queue menggunakan circular array

Sedangkan proses dequeue hanya bisa dilakukan jika queue dalam keadaan tidak kosong. Ada beberapa kondisi yang harus diperhatikan ketika dequeue elemen queue yaitu :

- ❑ Kondisi ketika posisi depan sama dengan posisi belakang (queue hanya memiliki 1 elemen) maka nilai depan dan belakang diisi dengan 0 (queue kosong).
- ❑ Jika posisi depan sama dengan posisi maks_queue maka posisi depan berpindah ke 1.
- ❑ Selain itu, posisi depan ditambah dengan 1 : **depan=depan+1**

Implementasinya adalah :

```
function dequeue (var q : antri) : char;  
begin  
  if (depan=0) and 9belakang=0) then then  
    cout (' antrian kosong');  
  else  
    begin  
      dequeue := q [depan];  
      if depan = max_elemen then  
        depan := 1  
      else  
        depan := depan + 1;  
      end;  
    end;  
end;
```



Terima
kasih

Latihan Soal

1. Jelaskan perbedaan Stack dan Queue?
 2. Cari studi kasus yang menggunakan konsep Stack atau Queue kemudian susun algoritma sesuai dengan alurnya dan gambarkan bagaimana proses operasinya.
- 