02. Stack

ARNA FARIZA

YULIANA SETIOWATI

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

Capaian Pembelajaran

1. Mahasiswa mengerti konsep stack dan operasi pada stack.
2. Mahasiswa dapat menggunakan stack untuk memecahkan permasalahan pemrograman.

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

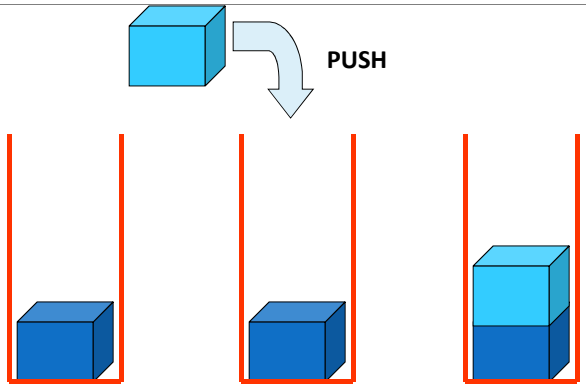
Materi

- Apakah Stack itu?
- Operasi pada Stack : Push dan Pop
- Konversi Infix ke Postfix dengan Stack

Apakah Stack itu?

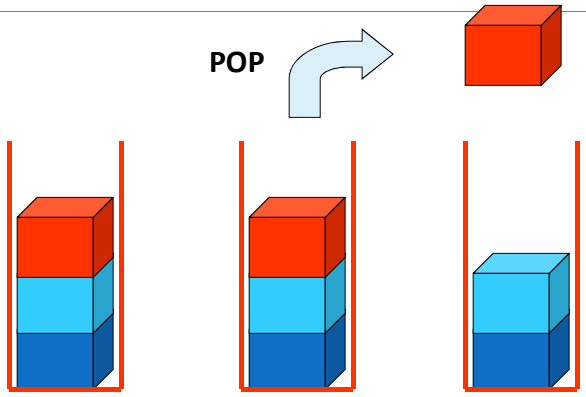
- Merupakan konsep Last In First Out (LIFO)
- Data yang disimpan terakhir akan diambil lebih dahulu
- Implementasi stack dapat menggunakan array atau linked list
- Implementasi dengan array, kemungkinan stack dalam kondisi penuh
- Implementasi dengan linked list, stack tidak pernah penuh

Ilustrasi Stack



Operasi Push meletakkan elemen/item ke stack

Ilustrasi Stack



Operasi Pop mengambil elemen/item dari stack

Elemen Stack

- Elemen/item yang diletakkan ke penyimpan
- Top of Stack (TOS)

Representasi Stack dengan Array

```
#define MAX 5
typedef int Itemtype
typedef struct {
   Itemtype item[MAX];
   int count;
} Stack;
```

Operasi pada Stack

- Push : menyimpan item pada stack
- Pop : mengambil item dari stack
- Inisialisasi : inisialisasi awal stack
- Penuh : stack dalam kondisi penuh
- Kosong : stack dalam kondisi kosong

Operasi Inisialisasi

- Menginisialisasi agar TOS menunjuk ke indeks array awal (indeks 0)

```
void Inisialisasi (Stack *s)
{
   s->count=0;
}
```

Operasi Penuh

- Melakukan pengecekan apakah stack Penuh atau Tidak (Jika penuh return value=1, sebaliknya value=0)
- Digunakan bila melakukan operasi PUSH

```
int Penuh (Stack *s)
{
    return (s->count==MAX);
}
```


Operasi Kosong

- Melakukan pengecekan apakah stack Kosong atau Tidak (Jika kosong return value=1, sebaliknya value=0)
- Digunakan bila melakukan operasi POP

```
int Kosong (Stack *s)
{
    return (s->count==0);
}
```

Operasi PUSH

- Untuk menyimpan data pada posisi teratas
- Bila array penuh, tidak dapat melakukan operasi Push
- Setelah dilakukan penyimpanan, posisi TOS di-increment

```
void Push (Stack *s, Itemtype x)
{
   if(Penuh(s))
      printf("Stack Penuh, Data tidak dapat disimpan\n");
   else {
      s->item[s->count]=x;
      s->count++;
   }
}
```


Operasi POP

- Mengambil data pada posisi teratas
- Bila array Kosong tidak dapat dilakukan operasi Pop
- Sebelum mengambil data TOS di-decrement

```
Itemtype Pop (Stack *s)
{
   Itemtype temp;
   if(Kosong(s)) {
      printf("Stack Kosong, tidak dapat mengambil data\n");
      return ' ';
   }
   else {
      s->count--;
      temp=s->item[s->count];
      return(temp);
   }
}
```

Konversi Notasi Infix ke Posfix

Notasi Infix	Notasi Postfix
A + B	AB+
A*B+C	AB*C+
A*(B+C)	ABC+*
A*B+C/D	AB*CD/+
(A+B)*C-D/E	AB+C*DE/-
(A+B)*(C-D)^E/F	AB+CD-E^*F/
A+B*C-D^E/F	ABC*+DE^F/-

Implementasi Konversi Notasi Infix ke Posfix dengan Stack

- Stack digunakan untuk menyimpan operator
- Operator mempunyai tingkatan level dengan urutan (dari level tertinggi ke terendah) : '^', '*' & '/', '+' dan '-'
- Notasi infix dibaca satu per satu
- Jika berupa operan langsung dicetak
- Jika operator mengikuti aturan:
  - Jika notasi '(' PUSH ke stack
  - Jika notasi ')' POP dan cetak s/d tanda ')' tetapi tidak dicetak
  - Jika operator, cek bila stack Kosong atau level operator > level operator TOS maka PUSH
  - Lainnya POP dan cetak lalu PUSH, ulangi perbandingan
- Jika notasi infix sudah berakhir, POP stack sampai Kosong

Ilustrasi Konversi Infix ke Postfix

A + B

Notasi infix	Stack	Cetak
A		A
+	+	A
B	+	AB
		AB+

Ilustrasi Konversi Infix ke Postfix

(A + B) * C

Notasi infix	Stack	Cetak
(	(	
A	(	A
+	(+	A
B	(+	AB
)		AB+
*	*	AB+
C	*	AB+C
		AB+C*

Ilustrasi Konversi Infix ke Postfix

A + B * C

Notasi infix	Stack	Cetak
A		A
+	+	A
B	+	AB
*	+*	AB
C	+*	ABC
		ABC*+

Ilustrasi Konversi Infix ke Postfix

A + B ^ C * D

Notasi infix	Stack	Cetak
A		A
+	+	A
B	+	AB
^	+^	AB
C	+^	ABC
*	+*	ABC^
D	+*	ABC^D
		ABC^D*+

Algoritma Konversi Notasi Infix ke Postfix

1. Sediakan stack untuk menyimpan operator (tipe : char)
2. Baca setiap karakter notasi infix dari awal
   1. Jika operand maka langsung dicetak
   2. Jika tanda '(' PUSH ke stack
   3. Jika tanda ')' POP dan cetak semua isi stack sampai TOS = '('. POP juga tanda '(' ini, tetapi tidak dicetak
   4. Jika operator : jika stack kosong atau derajad operator lebih tinggi dibanding derajad TOS, PUSH operator ke dalam stack.
   5. Jika tidak, POP dan cetak; kemudian ulangi pembandingan dengan TOS. Kemudian di-push
3. Jika akhir notasi infix telah tercapai, dan stack masih belum kosong, pop semua isi stack dan cetak hasilnya

Menghitung Hasil Operasi Postfix

- Stack menyimpan operan dan hasil operasi
- Siapkan variabel opLeft dan opRight untuk menyimpan operan kiri dan kanan
- Misalnya : 34+
  - PUSH '3'
  - PUSH '4'
  - Operator '+' → POP '4' ke opRight, POP '3' ke opLeft, lakukan operasi
  - Hasilnya '7' PUSH ke stack
  - Bila notasi berakhir, POP stack sebagai hasil operasi

Ilustrasi Hasil Operasi Postfix

23+5*

Notasi Postfix	Stack	Hasil Operasi
2	2	
3	2 3	
+	5	2 + 3
5	5 5	
*	25	5 * 5
		25

Algoritma Hasil Operasi Postfix

1. Baca notasi postfix satu per satu
   1. Jika notasi adalah operan maka PUSH ke stack
   2. Jika notasi adalah operator maka
      1. POP ke OpRight
      2. POP ke OpLeft
      3. Hasil = OpLeft operator OpRight
      4. PUSH Hasil
2. Jika notasi postfix berakhir, POP stack sebagai hasil operasi

Rangkuman

- Stack menyimpan elemen/item dengan konsep LIFO, dimana item yang terakhir masuk akan keluar terlebih dahulu
- Elemen pada stack terdiri dari : item yang disimpan di penyimpan, penunjuk top of stack sekaligus menghitung jumlah elemen (count)
- Terdapat dua operasi stack yaitu PUSH dan POP
- Selain itu terdapat operasi tambahan yaitu inisialisasi, Penuh, Kosong

Latihan

1. Buatlah konversi bilangan desimal ke biner, oktal dan heksa menggunakan stack
2. Buatlah pembalik kalimat menggunakan stack

   Contoh : `Struktur Data`
   `ataD rutkurtS`

   Buatlah pengecekan palindrom atau bukan, Contoh : sugus bila diBuatlah pembalik kalimat menggunakan stack
3. Implementasikan notasi infix ke postfix menggunakan stack
4. Implementasikan operasi notasi postfix menggunakan stack