Stack

BAB 3

Materi

3.1 DESKRIPSI STACK

3.2 PENYAJIAN STACK

3.3 OPERASI PADA STACK

- 3.3.1 Operasi Push
- 3.3.2 Operasi Pop

3.4 Notasi POLISH

3.5 Kesimpulan

3.1 DESKRIPSI STACK

Salah satu konsep yang efektif untuk menyimpan dan mengambil data adalah "terakhir masuk sebagai yang pertama keluar" (Last In First Out / LIFO).

Dengan konsep ini, pengambilan data akan berkebalikan urutannya dengan penyimpanan data.

Stack adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO.

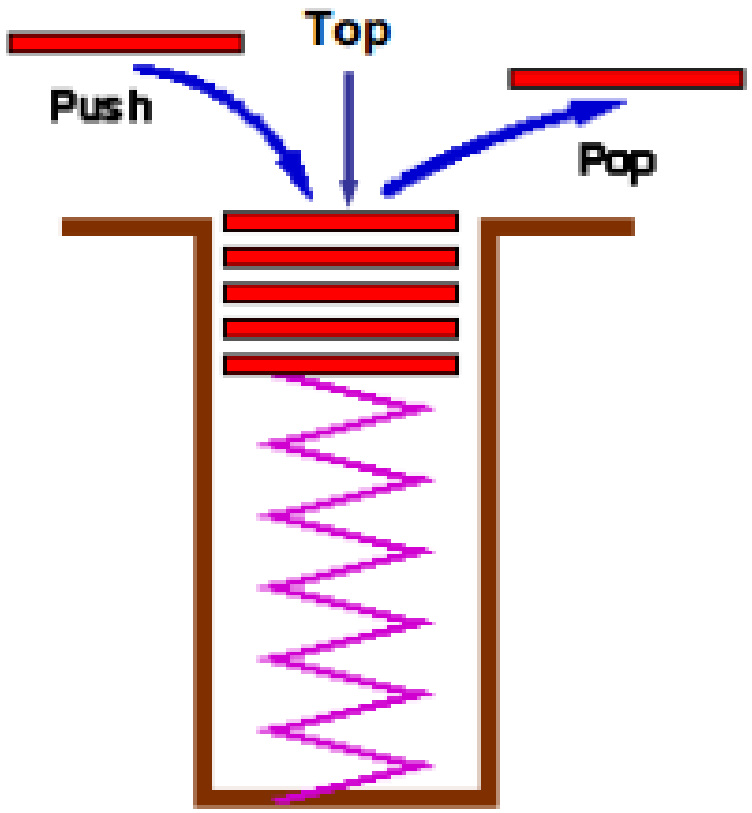
Ilustrasi Stack

Dalam proses komputasi, untuk meletakkan sebuah elemen pada bagian atas dari stack, maka kita melakukan push.

Dan untuk memindahkan dari tempat yang atas tersebut, kita melakukan pop.

Misalnya kita mempunyai dua buah kotak yang kita tumpuk, sehingga kotak kita letakkan di atas kotak yang lain.

Jika kemudian stack dua buah kotak tersebut kita tambah dengan kotak ketiga dan seterusnya, maka akan kita peroleh sebuah stack kotak, yang terdiri dari N kotak.



3.2 PENYAJIAN STACK

Dalam bab ini kita akan menggunakan cara yang paling sederhana, tipe data yang sudah kita kenal, yaitu array.

Kita dapat menggunakan array untuk menyajikan sebuah stack, dengan anggapan bahwa banyaknya elemen maksimum dari stack tersebut tidak akan melebihi batas maksimum banyaknya elemen dalam array.

Pada saat ukuran stack, kalau kita teruskan menambah data lagi, akan terjadi overflow. Dengan demikian perlu data tambahan untuk mencatat posisi ujung stack.

Dengan kebutuhan seperti ini, kita dapat menyajikan stack dengan menggunakan tipe data struktur (struct) yang terdiri dari dua field.

Field pertama bertipe array untuk menyimpan elemen stack, medan kedua bertipe integer untuk mencatat posisi ujung stack.

```
#define MAXSTACK 100
        typedef int ItemType;
        /* Definisi struktur stack */
        typedef struct
{
        int Item[MAXSTACK]; /* Array yang berisi data tumpukan */
        int Count; /* menunjukkan indeks data paling atas
        dari stack */
}Stack;
```

3.3 OPERASI PADA STACK

1. Operasi menciptakan T sebagai stack kosong

*void InitializeStack(Stack *S)*

{ S->Count = 0; }

2. Fungsi yang melakukan pengecekan apakah stack dalam kondisi kosong

*int Empty(Stack *S)*

{ return (S->Count == 0); }

3. Fungsi yang melakukan pengecekan apakah stack dalam kondisi penuh

*int Full(Stack *S)*

{ return (S->Count == MAXSTACK); }

3.3 OPERASI PADA STACK

4. Operasi menyisipkan elemen x ke stack T dan mengembalikan stack baru

```
void Push(ItemType x, Stack *S)
{ if (S->Count==MAXSTACK)
printf("Stack penuh! Data tidak dapat masuk!"); else
{ S->Item[S->Count]=x;
++(S->Count); } }
```

5. Operasi mengambil elemen puncak stack T, (pop(T,x))

```
int Pop(Stack *S, ItemType *x)
{ if (S->Count==0)//stack kosong
printf("Stack masih kosong!");
else
{ --(S->Count);
*x = S->Item[S->Count]; } }
```

3.3.1 Operasi Push

Fungsi tersebut menyiapkan tempat untuk **x** yang akan dipush ke dalam tumpukan,

yaitu dengan menambah nilai **S->Count** dengan **1** dan kemudian menyisipkan **x** ke dalam **S->Item**.

Berikut Prosedur operasi push

```
x = 5;

Push(x, &tum);
```

3.3.2 Operasi Pop

Operasi pop adalah operasi untuk menghapus elemen yang terletak pada posisi paling atas dari sebuah tumpukan.

Cara pemanggilan satu data prosedur pop di atas adalah sebagai berikut:

```
/* mengambil satu data dari tumpukan */
hasil = Pop(&tum);
printf("Data %d hasil pengambilan dari tumpukan : ",hasil);
```

3.4 Notasi POLISH

Salah satu pemanfaatan stack adalah untuk menulis ungkapan menggunakan notasi tertentu.

Seperti kita ketahui, dalam penulisan ungkapan, khususnya ungkapan numeris, kita selalu menggunakan tanda kurung untuk mengelompokkan bagian mana yang akan dikerjakan lebih dahulu.

Misal : (A + B) * (C – D)

Perbedaannya dengan A + B * C – D

Notasi Polish (Infix dan Prefix)

Notasi Polish atau notasi prefix, yang artinya adalah operator ditulis sebelum kedua operand yang akan disajikan. Berikut disajikan beberapa contoh notasi prefix dari notasi infix:

Infix	Prefix
A + B	+ A B
A + B – C	- + A B C
(A + B) * (C – D)	* + A B – C D
A – B / (C * D ^ E)	- A / B * C ^ D E

Notasi Polish (Postfix / Suffix)

lebih dikenal dengan notasi Polish Terbalik (Reverse Polish Notation atau RPN).

Dalam hal ini operator ditulis sesudah operand. Sama halnya dengan notasi prefix, maka dalam notasi postfix inipun tidak diperlukan adanya tanda kurung pengelompokan.

Dalam hal ini urutan penulisan operator menentukan operasi mana yang harus dikerjakan lebih dahulu. Berikut ini disajikan beberapa contoh lain hasil konversi notasi infix menjadi postfix.

Infix	Postfix
A + B – C	A B + C -
(A + B) * (C – D)	A B + C D - *
A – B / (C * D ^ E)	A B C D E ^ * / -

Algoritma Mengubah notasi Infix Menjadi Postfix

1. Baca ungkapan dalam notasi infix, misalnya S, tentukan panjang ungkapan tersebut, misalnya N karakter, siapkan sebuah stack kosong dan siapkan derajad masing-masing operator, misalnya: ^ berderajad 3, * dan / berderajad 2, + dan – berderajad 1 dan (berderajad 0.

2. Dimulai dari i = 1 sampai N kerjakan langkah-langkah sebagai berikut:
   - a. R = S[I]
   - b. Test nilai R. Jika R adalah:
     - operand : langsung ditulis
     - kurung buka : push ke dalam tumpukan
     - kurung tutup : pop dan tulis semua isi tumpukan sampai ujung tumpukan = ‘(‘. Pop juga tanda ‘(‘ ini, tetapi tidak usah ditulis
     - operator : jika tumpukan kosong atau derajad R lebih tinggi dibanding derajad ujung tumpukan, push operator ke dalam tumpukan. Jika tidak, pop ujung tumpukan dan tulis; kemudian ulangi pembandingan R dengan ujung tumpukan. Kenudian R di-push
   - c. Jika akhir notasi infix telah tercapai, dan tumpukan masih belum kosong, pop semua isi tumpukan dan tulis hasilnya

Ilustrasi (A + B) / ((C – D) * E ^ F)

Karakter dibaca	Isi Tumpukan	Karakter tercetak	Hasil Notasi Postfix Yang Terbentuk
(	(		
A	(+	A	A
+	(+		
B		B	A B
)		+	A B +
/	/		
(	/ (		
(	/ ((		
C	/ ((	C	A B + C
-	/ ((-		
D	/ ((-	D	A B + C D
)	/ (	-	A B + C D -
*	/ (*		
E	/ (*	E	A B + C D - E
^	/ (* ^		
F	/ (* ^	F	A B + C D - F
)	/ (*	^	A B + C D – F ^
	/ (	*	A B + C D – F ^ *
	/		
		/	A B + C D – F ^ *

3.5 Kesimpulan

1. Stack atau tumpukan merupakan data yang sifatnya terurut yang dapat dilakukan operasi penyisipan dan penghapusan pada ujung data

2. Stack sering disebut LIFO (Last In First Out) yaitu elemen yang paling akhir disisipkan menjadi elemen yang paling dulu diambil

Terima Kasih

Pertanyaan?