BAB 3

STACK (TUMPUKAN)

LINIER LIST

Suatu struktur data umum yang berisi suatu kumpulan terurut dari elemen; jumlah elemen di dalam list dapat berubah-ubah.

Linier list A yang terdiri dari T elemen pada waktu t, dinotasikan sebagai : A = [A1, A2, ..., AT]

Jika T = 0, maka A disebut “Empty List” atau “Null List”

Suatu elemen dapat dihilangkan/dihapus dari sembarang posisi dalam linier list, dan dapat pula dimasukkan elemen baru sebagai anggota list.

Contoh :
1. File, dengan elemennya berupa record
2. Buku telepon
3. Stack
4. Queue
5. Linear link list

STACK

Stack adalah suatu bentuk khusus dari linier list, dengan operasi penyisipan dan penghapusan dibatasi hanya pada satu sisinya, yaitu puncak stack (TOP).

Elemen teratas dari stack dinotasikan sebagai TOP(S).
Untuk stack S, dengan S = [S_1, S_2, S_3, ..., S_T]
maka TOP(S) = S_T

Jumlah elemen di dalam stack kita notasikan dengan NOEL(S).
NOEL(S) menghasilkan nilai integer.
Untuk stack S = [S_1, S_2, S_3, ..., S_T] maka NOEL (S) = T.

Operator penyisipan (insertion) : PUSH
Operator penghapusan (deletion) : POP
Operasi stack : **LIFO** (**L**ast **I**n **F**irst **O**ut), yaitu : yang terakhir masuk yang pertama keluar.

Jika ada NOEL elemen didalam stack, maka elemen ke NOEL merupakan elemen puncak (TOP).

Stack secara umum :

$S = [S_1, S_2, ..., S_{NOEL}]$
bahwa : S_I berada di atas elemen S_J, untuk $I > J$
S_I akan dikeluarkan lebih dulu dari elemen di bawahnya.

Contoh stack : Tumpukan baki dalam cafetaria

Empat operasi dasar yang berlaku pada stack :
1. CREATE(stack)
2. ISEMPTY(stack)
3. PUSH(elemen, stack)
4. POP(stack)

- **CREATE**
  adalah operator yang menunjukkan suatu stack kosong dengan nama S.

  Jadi : NOEL(CREATE(S)) = 0
  TOP(CREATE(S)) adalah TIDAK TERDEFINISI.

- **ISEMPTY**
  adalah operator yang menentukan apakah stack S kosong.
  Operandnya terdiri dari type data stack. Hasilnya merupakan type data Boolean.

  ISEMPTY(S) = True. Jika S hampa, yakni bila NOEL(S) = 0.

- **PUSH**
  adalah operator yang menambahkan elemen E pada puncak stack S. Hasilnya merupakan stack yang lebih besar.
  PUSH(E,S). E ditempatkan sebagai TOP(S).

- **POP(stack)**
  adalah operator yang menghapus sebuah elemen dari puncak stack S. Hasilnya merupakan stack yang lebih kecil.

  - POP(S) mengurangi NOEL(S)
  - POP(CREATE(S)) $\rightarrow$ kondisi error
  - POP(PUSH(E,S)) = S

DEKLARASI STACK DALAM COBOL DAN PASCAL

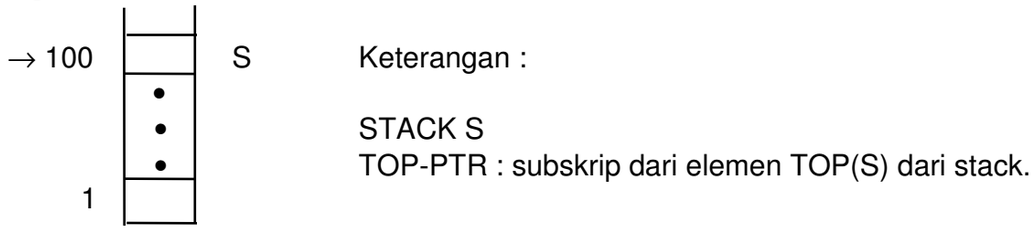
Cara yang paling sederhana adalah membentuk Stack dalam bentuk semacam Array.

Penempatan Stack dalam suatu Array mengakibatkan suatu keterbatasan, yakni bahwa elemen Stack harus homogen.

Keharusan pemrograman untuk menentukan batas atas dari subskrip Array, walaupun Stack sebenarnya tidak memiliki batas maksimum dalam jumlah elemen.

Yang membedakan Stack dengan Array adalah banyaknya elemen Stack dapat bertambah atau berkurang setiap waktu, sementara banyaknya elemen sebuah array selalu tetap.

Deklarasi dari Variabel S yang bertipe data Stack. Diasumsikan bahwa elemen dari S masing-masing bertipe data integer dan panjang Stack maksimum 100 elemen.
Kita mendeklarasikan sebuah Array yang dilengkapi dengan Variabel TOP-PTR. Variabel TOP-PTR ini menyatakan subkrip dari elemen TOP(S) dari Stack.



COBOL

```
01   STACK-STRUCT  → kombinasi dari array dan indikator untuk TOP
     02     S OCCURS 100 TIMES PIC 9(5)
     02     TOP-PTR                PIC 9(3)
```

PASCAL

```
TYPE STACKSTRUCT  =      RECORD
                         STACK : ARRAY [1..100] of integer;
                         TOPPTR : integer;
                         END;

VAR  S :  STACKSTRUCT;
```

NOEL(S) = TOP-PTR, ISEMPTY(S) = true, bila TOP-PTR = 0.

OPERASI PUSH & POP

PUSH

```
IF TOP-PTR < NOEL-MAX
     THEN COMPUTE TOP-PTR = TOP-PTR + 1
          MOVE EON  TO S(TOP-PTR)
     ELSE Overflow condition
```

POP

```
IF TOP-PTR > 0
    THEN MOVE S(TOP-PTR) TO EOFF
            COMPUTE TOP-PTR = TOP-PTR  -  1
    ELSE Underflow condition
```

EON : elemen yang di PUSH ke dalam S.
EOFF : elemen yang di POP ke luar S.
NOEL-MAX : panjang max stack.

PUSH

```
Procedure PUSH (eon: integer);
Begin
    if (s.topptr < noelmax)
    then
        Begin
            s.topptr := s.topptr + 1;
            s.stack [s.topptr] := eon;
        End;
    else Overflow-condition
End;
```

POP

```
Procedure POP (var eoff : integer);
Begin
    if (s.topptr > 0)
    then
        Begin
            eoff := s.stack [s.topptr];
            s.topptr := s.topptr - 1;
        End;
    else Underflow Condition
End;
```

APLIKASI STACK

1. Penjodohan Tanda Kurung/Matching Parantheses

ALGORITMA

a. Amati barisan elemen dari kiri ke kanan
b.
- bila bertemu '(', maka '(' di push ke dalam stack.
- bila bertemu ')', maka periksa stack hampa atau tidak.
  bila ***hampa*** → ada ')' dan tidak ada '(' (error)
  bila ***tidak hampa*** → ada sepasang '(' & ')' & POP elemen keluar

2. NOTASI POSTFIX

ALGORITMA

Amati barisan dari kiri ke kanan

1. Jika ‘(‘, maka PUSH ke dalam stack.
2. Jika ‘)’, POP elemen dalam stack sampai simbol ‘(‘. Semua di POP merupakan output kecuali ‘(‘ tadi.
3. Jika simbol operand, langsung merupakan output.
4. Jika simbol operator, maka :
   Jika elemen TOP stack dengan level >= maka POP sebagai output teruskan sampai ‘(‘.
   elemen TOP <, operator yang diamati di PUSH ke dalam stack.
5. Bila ‘;’ kita POP semua elemen dalam stack hingga hampa.

APLIKASI STACK

Notasi Postfix

Contoh :

Notasi Infix : ((A+B) * C/D+E^F)/G;

Simbol yang diamati	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	(	(	A	+	B	)	*	C	/	D	+	E	^	F	)	/	G	;
TOP dari STACK	(	(	(	+	+	(	*	*	/	/	+	+	^	^	(	/	/	(
				(	(		(	(	(	(	(	(	+	+				
				(	(			(	(	(	(	(	(	(				
OUTPUT			A		B	+		C	*	D	/	E		F	^+		G	/

Soal :

1. A * B - (C + D) -(E - F) + F/H ^ I;
2. ((B * C) + C/D ^ F) + G;
3. A ^ B * C - D + E/F / (G + H);