

# Stack dengan Struktur Berkait

IF2110/IF2111 – Algoritma dan Struktur Data  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung

# Stack

**Stack**, sederetan elemen yang:

- dikenali elemen puncaknya (Top)
- aturan penambahan dan penghapusan elemennya tertentu:  
**Penambahan** selalu dilakukan **"di atas" Top**  
**Penghapusan** selalu dilakukan **pada Top**

Top adalah satu-satunya lokasi terjadinya operasi

Elemen Stack tersusun secara LIFO (Last In First Out)

# Definisi operasi

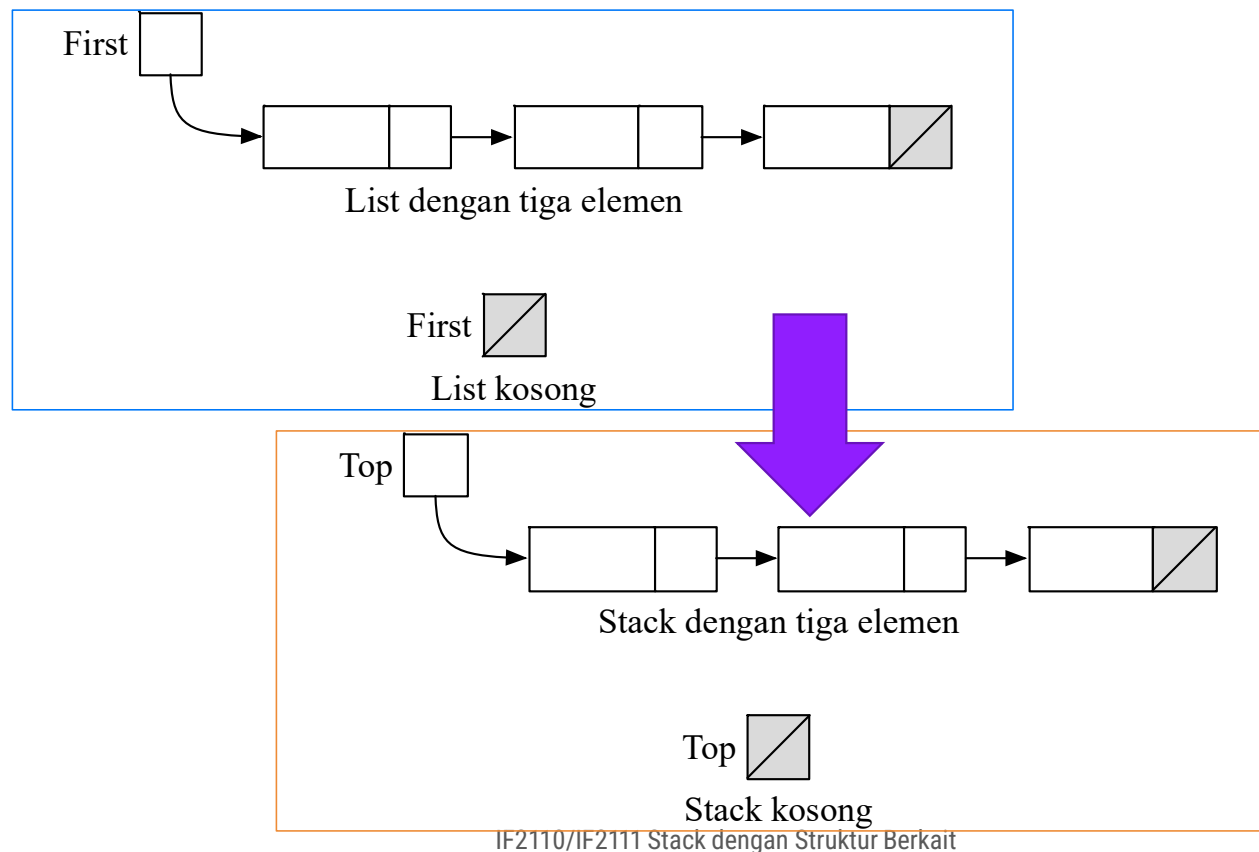
Jika diberikan  $S$  adalah Stack dengan elemen  $ElmtS$

$CreateStack: \rightarrow S$	{ Membuat sebuah tumpukan kosong }
$top: S \rightarrow ElmtS$	{ Mengirimkan elemen teratas $S$ saat ini }
$length: S \rightarrow \underline{integer}$	{ Mengirimkan banyaknya elemen $S$ saat ini }
$push: ElmtS \times S \rightarrow S$	{ Menambahkan sebuah elemen $ElmtS$ sebagai TOP, TOP berubah nilainya }
$pop: S \rightarrow S \times ElmtS$	{ Mengambil nilai elemen TOP, sehingga TOP yang baru adalah elemen yang datang sebelum elemen TOP, mungkin $S$ menjadi kosong }
$isEmpty: S \rightarrow \underline{boolean}$	{ Test stack kosong, true jika $S$ kosong, false jika $S$ tidak kosong }

**Representasi berkait seperti apa yang paling cocok untuk stack?**

# Stack dengan Struktur Berkait

List linier “biasa”  $\approx$  stack



# Operasi-Operasi Dasar pada Stack

CreateStack  $\approx$  CreateList

push  $\approx$  insertFirst

pop  $\approx$  deleteFirst

length  $\approx$  length

isEmpty  $\approx$  isEmpty

# ADT Stack Representasi List

```
/* File: stack_linked.h */
#ifndef STACK_LINKED_H
#define STACK_LINKED_H
#include "boolean.h"
#include <stdlib.h>

#define NIL NULL
/* Deklarasi infotype */
typedef int ElType;
/* Stack dengan representasi berkait dengan pointer */
typedef struct node* Address;
typedef struct node {
    ElType info;
    Address next;
} Node;

/* Type stack dengan ciri Top: */
typedef struct {
    Address addrTop; /* alamat Top: elemen puncak */
} Stack;
```

# ADT Stack Representasi List

```
/* Selektor */
#define NEXT(p) (p)->next
#define INFO(p) (p)->info
#define ADDR_TOP(s) (s).addrTop
#define TOP(s) (s).addrTop->Info

/* Prototype manajemen memori */
Address newNode(EIType x);
/* Mengembalikan alamat sebuah Node hasil alokasi dengan info = x,
   atau NIL jika alokasi gagal */
```

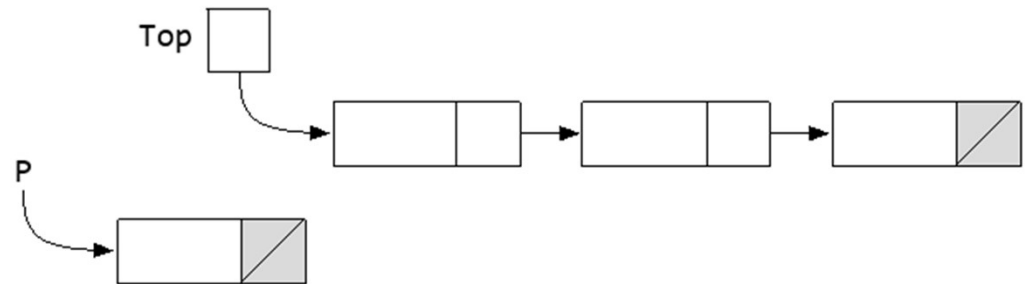


# ADT Stack Representasi List

```
/* ***** PROTOTYPE REPRESENTASI LOGIK STACK ***** */
boolean isEmpty(Stack s);
/* Mengirim true jika Stack kosong: TOP(s) = NIL */
void CreateStack(Stack *s);
/* I.S. sembarang */
/* F.S. Membuat sebuah stack s yang kosong */
void push(Stack *s, ElType x);
/* Menambahkan x sebagai elemen Stack s */
/* I.S. s mungkin kosong, x terdefinisi */
/* F.S. x menjadi Top yang baru jika alokasi x berhasil, */
/*      jika tidak, s tetap */
/* Pada dasarnya adalah operasi insertFirst pada list linier */
void pop(Stack *s, ElType *x);
/* Menghapus Top dari Stack s */
/* I.S. s tidak kosong */
/* F.S. x adalah nilai elemen Top yang lama, */
/*      elemen Top yang lama didealokasi */
/* Pada dasarnya adalah operasi deleteFirst pada list linier */
#endif
```

# ADT Stack Representasi List

```
void push(Stack *s, ElType x);  
/* Menambahkan x sebagai elemen Stack s */  
/* I.S. s mungkin kosong, x terdefinisi */  
/* F.S. x menjadi Top yang baru jika alokasi x berhasil, */  
/*     jika tidak, s tetap */  
/* Pada dasarnya adalah operasi insertFirst pada list linier */  
  
/* Kamus Lokal */  
Address p;  
  
/* Algoritma */  
p = newNode(x);  
if (p != NIL) {  
    NEXT(p) = ADDR_TOP(*s);  
    ADDR_TOP(*s) = p;  
} /* else: alokasi gagal, s tetap */  
}
```



# ADT Stack Representasi List

```
void pop(Stack *s, ElType *x);  
/* Menghapus Top dari Stack s */  
/* I.S. s tidak mungkin kosong */  
/* F.S. x adalah nilai elemen Top yang lama, */  
/*     elemen Top yang lama didealokasi */  
/* Pada dasarnya adalah operasi deleteFirst pada list linier */  
  
/* Kamus Lokal */  
Address p;  
  
/* Algoritma */  
*x = TOP(*s);  
p = ADDR_TOP(*s);  
ADDR_TOP(*s) = NEXT(ADDR_TOP(*s));  
NEXT(p) = NIL;  
free(p);  
}
```

