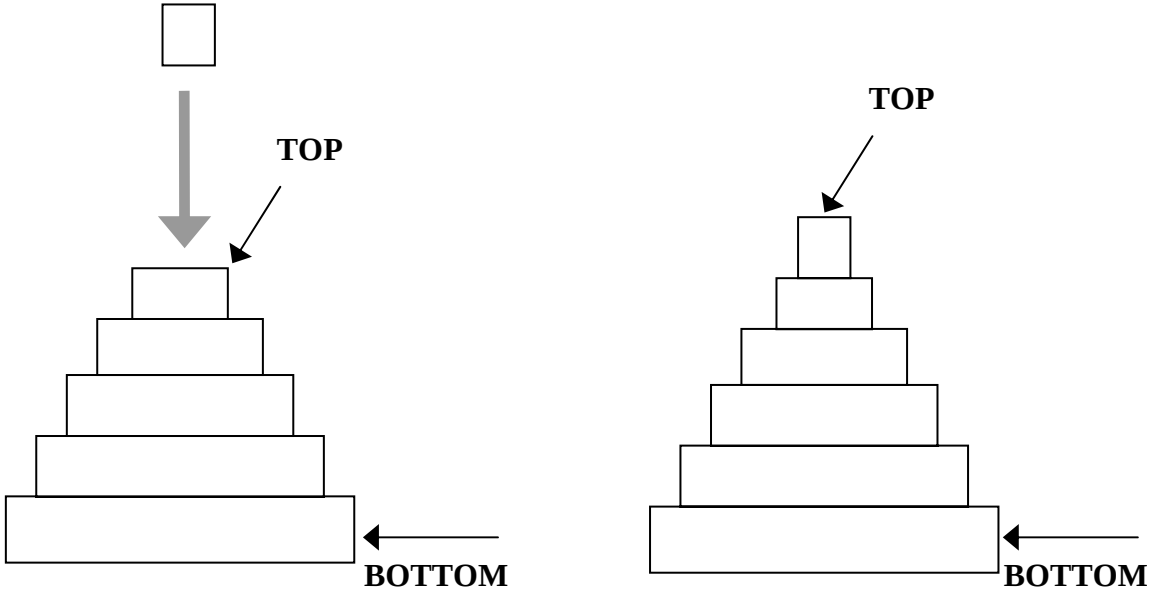
Stack (Tumpukan)

Tim Pengajar IF2030

Stack

- Stack: list linier yang:
  - dikenali elemen puncaknya (TOP)
  - aturan penyisipan dan penghapusan elemennya tertentu:
    - Penyisipan selalu dilakukan "di atas" TOP
    - Penghapusan selalu dilakukan pada TOP
- TOP adalah satu-satunya alamat tempat terjadi operasi
- Elemen Stack tersusun secara LIFO *(Last In First Out)*

Stack



- Dengan definisi semacam ini, representasi **tabel** sangat tepat untuk mewakili Stack, karena operasi penambahan dan pengurangan hanya dilakukan di salah satu “ujung” tabel

Pemakaian Stack

- Pemanggilan prosedur
- Perhitungan ekspresi aritmatika
- Rekursifitas
- *Backtracking*
- dan algoritma lanjut yang lain

Definisi Fungsional

- Jika diberikan **S** adalah Stack dengan elemen **ElmtS**:

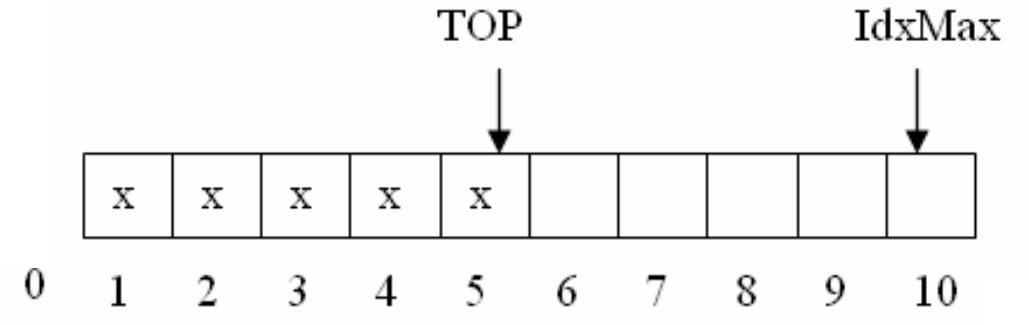
CreateEmpty	: $\rightarrow$ S	{ Membuat sebuah stack kosong}
IsEmpty	: S $\rightarrow$ boolean	{ Test stack kosong, true jika stack kosong, false jika S tidak kosong }
IsFull	: S $\rightarrow$ boolean	{ Test stack penuh, true jika stack penuh, false jika S tidak }
Push	: ElmtS x S $\rightarrow$ S	{ Menambahkan sebuah elemen ElmtS sebagai TOP, TOP berubah nilainya }
Pop	: S $\rightarrow$ S x ElmtS	{ Mengambil nilai elemen TOP, sehingga TOP yang baru adalah elemen yang datang sebelum elemen TOP, mungkin S menjadi kosong }

Definisi Fungsional

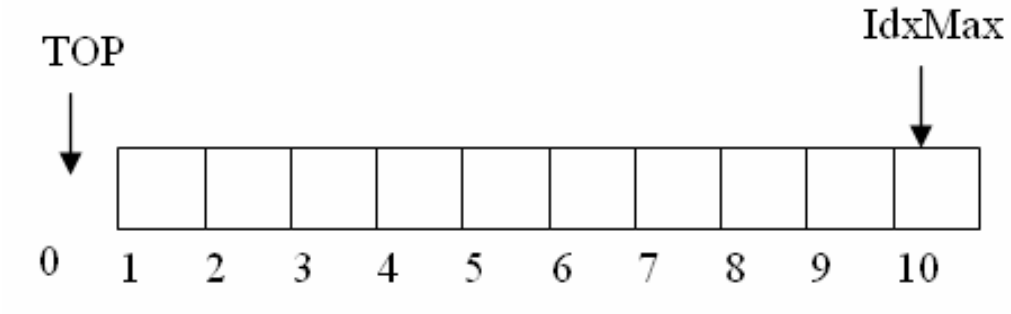
- Definisi Selektor:
  - Jika **S** adalah sebuah Stack, maka
    - **Top(S)** adalah alamat elemen TOP, di mana operasi penyisipan/penghapusan dilakukan
    - **InfoTop(S)** adalah informasi yang disimpan pada Top(S)
- Definisi **Stack kosong** adalah Stack dengan **Top(S)=Nil** (tidak terdefinisi)

Implementasi Stack dengan Tabel

- Ilustrasi Stack tidak kosong, dengan 5 elemen:



- Ilustrasi Stack kosong



ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
/* File : stackt.h */
/* Implementasi Stack dalam bahasa C dengan alokasi statik */
#ifndef stackt_H
#define stackt_H
#include "boolean.h"
#define Nil 0  /* Nil adalah stack dengan elemen kosong */
#define MaxEl 10

typedef int infotype;
typedef int address;   /* indeks tabel */
/* Versi I : dengan menyimpan tabel dan alamat top secara eksplisit */
typedef struct { infotype T[MaxEl+1]; /* tabel penyimpan elemen */
                 address TOP;    /* alamat TOP: elemen puncak */
               } Stack;

/* Definisi akses dengan Selektor : */
#define Top(S) (S).TOP
#define InfoTop(S) (S).T[(S).TOP]
```

ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
/*** Konstruktor/Kreator ***/
void CreateEmpty(Stack *S);
/* I.S. Sembarang */
/* F.S. Membuat sebuah stack S yang kosong berkapasitas MaxEl */
/* jadi indeksnya antara 1..MaxEl karena 0 tidak dipakai */
/* Ciri stack kosong : TOP bernilai Nil */
/********** Predikat Untuk test keadaan KOLEKSI **********/
boolean IsEmpty (Stack S);
/* Mengirim true jika Stack kosong: lihat definisi di atas */
boolean IsFull(Stack S);
/* Mengirim true jika tabel penampung nilai elemen stack penuh */
/*********** Menambahkan sebuah elemen ke Stack **********/
void Push (Stack *S, infotype X);
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */
/*********** Menghapus sebuah elemen Stack **********/
void Pop (Stack *S, infotype *X);
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */
#endif
```

ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
void CreateEmpty(Stack *S)
/* I.S. Sembarang */
/* F.S. Membuat sebuah stack S yang kosong berkapasitas MaxEl */
/* jadi indeksnya antara 1.. MaxEl karena 0 tidak dipakai */
/* Ciri stack kosong : TOP bernilai Nil */
{  /* Kamus Lokal */

   /* Algoritma */
   Top(*S) = Nil;
}
```

ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
boolean IsEmpty (Stack S)
/* Mengirim true jika Stack kosong: lihat definisi di atas */
{  /* Kamus Lokal */

   /* Algoritma */
   return (Top(S)==Nil);
}
boolean IsFull(Stack S)
/* Mengirim true jika tabel penampung nilai elemen stack penuh */
{  /* Kamus Lokal */

   /* Algoritma */
   return(Top(S)==MaxEl);
}
```

ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
void Push (Stack *S, infotype X)
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru, TOP bertambah 1 */
{  /* Kamus Lokal */

   /* Algoritma */
   Top(*S)++;
   InfoTop(*S) = X;
}
```

ADT Stack dalam Bahasa C (dengan array eksplisit-statik)


```
void Pop (Stack *S, infotype *X);
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */
{  /* Kamus Lokal */

   /* Algoritma */
   *X = InfoTop(*S);
   Top(*S)--;
}
```

ADT Stack dalam Bahasa C (dengan array eksplisit-dinamik)

- Diktat Struktur Data hlm. 46

Contoh Aplikasi Stack

- Evaluasi ekspresi matematika yang ditulis dengan notasi POLISH (Postfix)
  - Diktat Struktur Data hlm. 48
  - Latihan: Implementasikan dalam Bahasa C

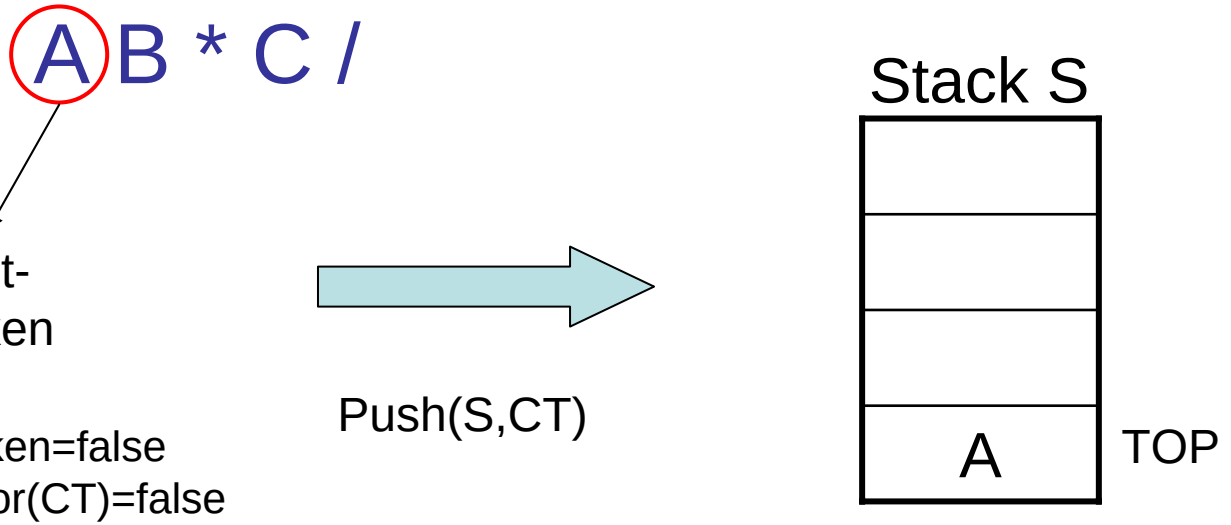
Kasus: Evaluasi Ekspresi Matematika

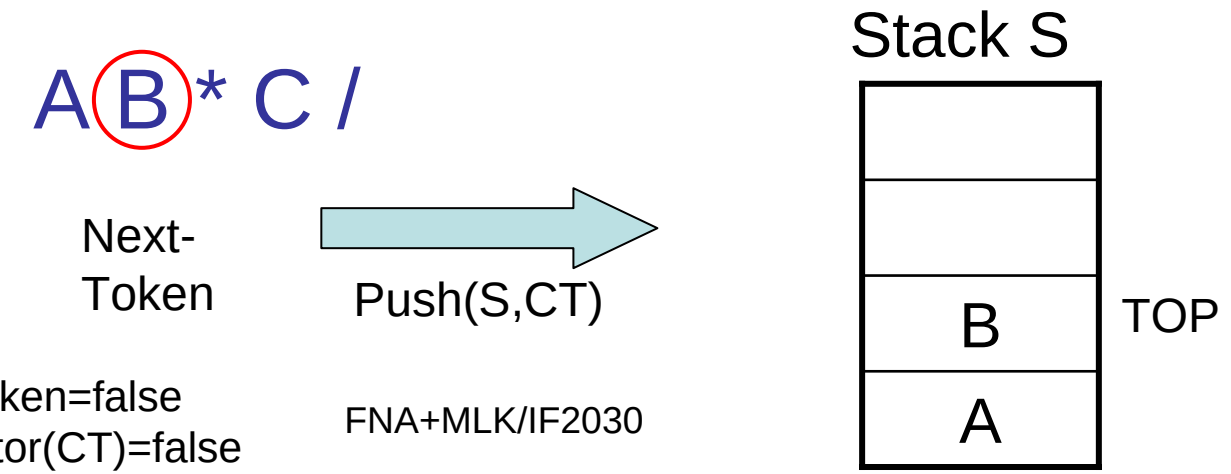
- Ekspresi ditulis dalam notasi postfix
- Operator: '*','/','+','-','^'
- Contoh:

  A B * C / → **(A*B)/C**

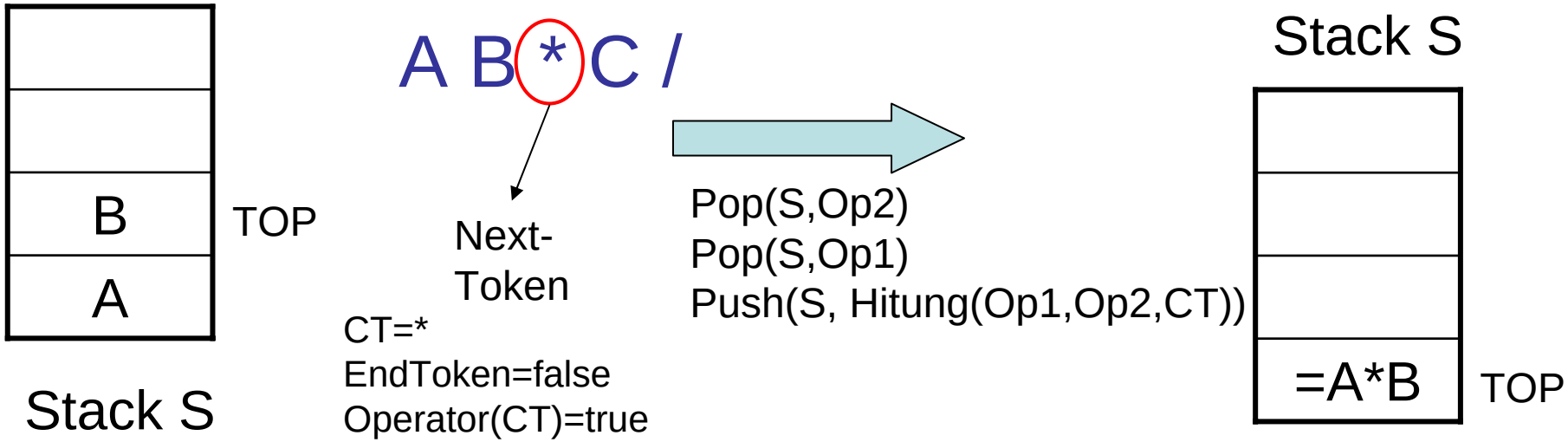
  A B C ^ / D E * + A C * - → **(A/(B^C))+(D*E)-(A*C)**
- Token: kata yang mewakili operan atau operator

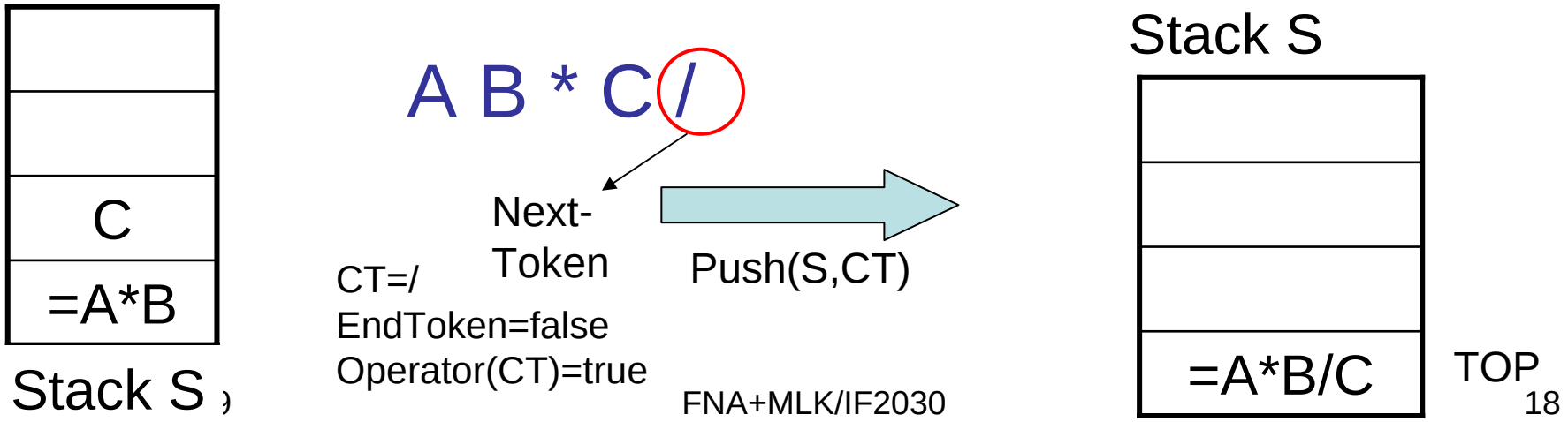
Program Ekspresi (hal 48)



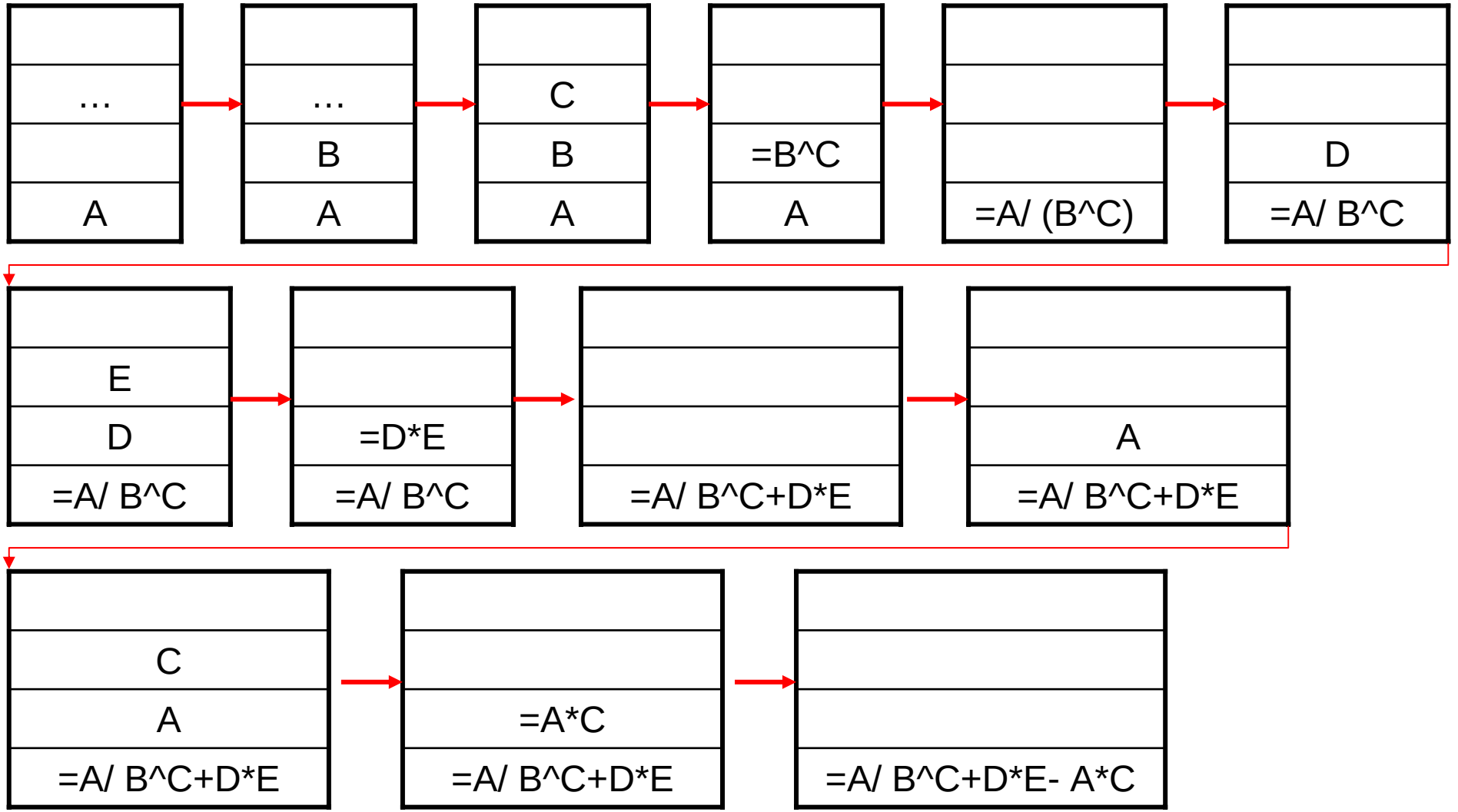


Program Ekspresi (lanjutan)





ABC^/DE*+AC*- ?


…
A
…
B
A
C
B
A
=B^C
A
=A/ (B^C)
D
=A/ B^C
E
D
=A/ B^C
=D*E
=A/ B^C
=A/ B^C+D*E
A
=A/ B^C+D*E
C
A
=A/ B^C+D*E
=A*C
=A/ B^C+D*E
=A/ B^C+D*E- A*C

PR

- Modul pra-praktikum:
  - P-08.Stack
    - Bagian 1. Representasi Tabel Kontigu dengan Alokasi Memori Statik
    - Bagian 2. Representasi Tabel Kontigu dengan Alokasi Memori Dinamik