

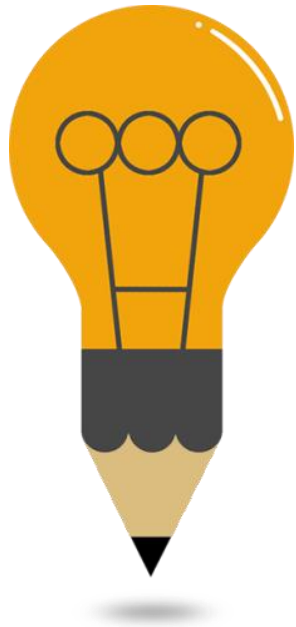
STRUKTUR DATA

Pertemuan 6



Ratih Ngestrini, Nori Wilantika

Agenda Pertemuan



1

Review Latihan Double Linked List

2

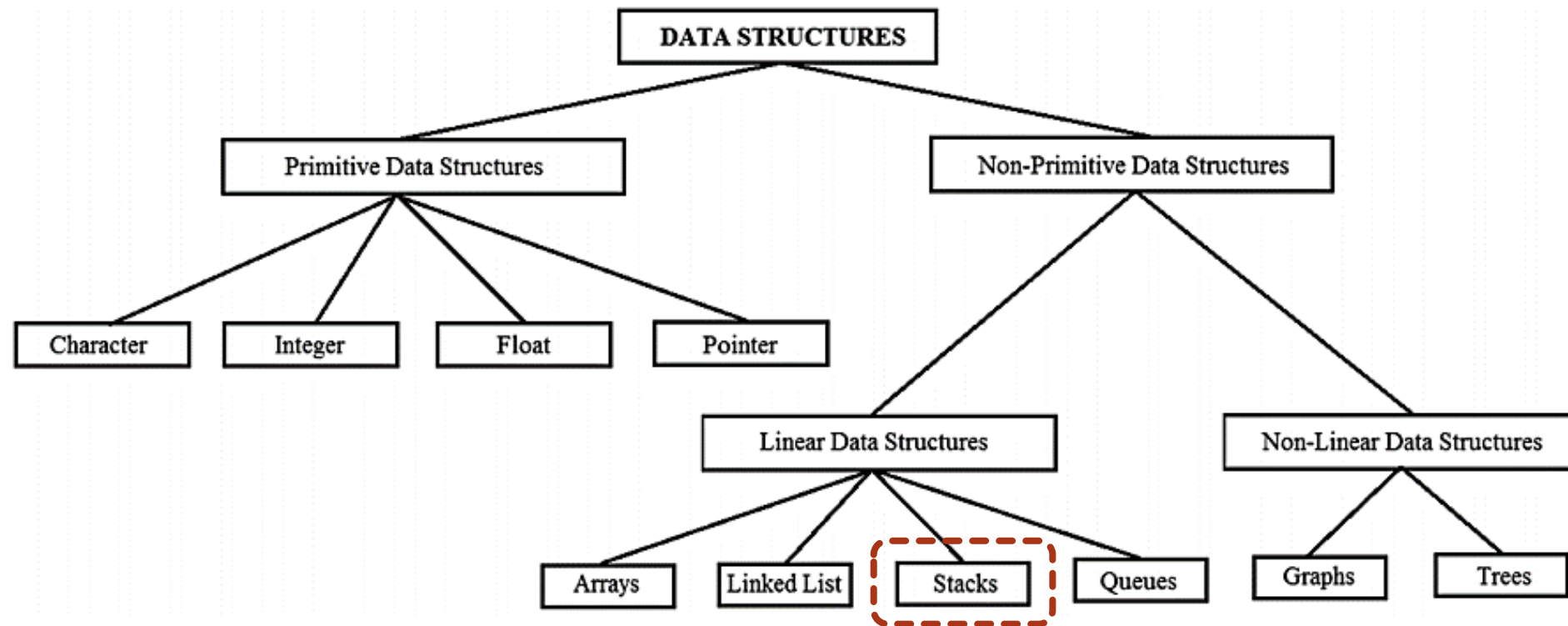
Tumpukan (*stack*)



TUMPUKAN (STACK)

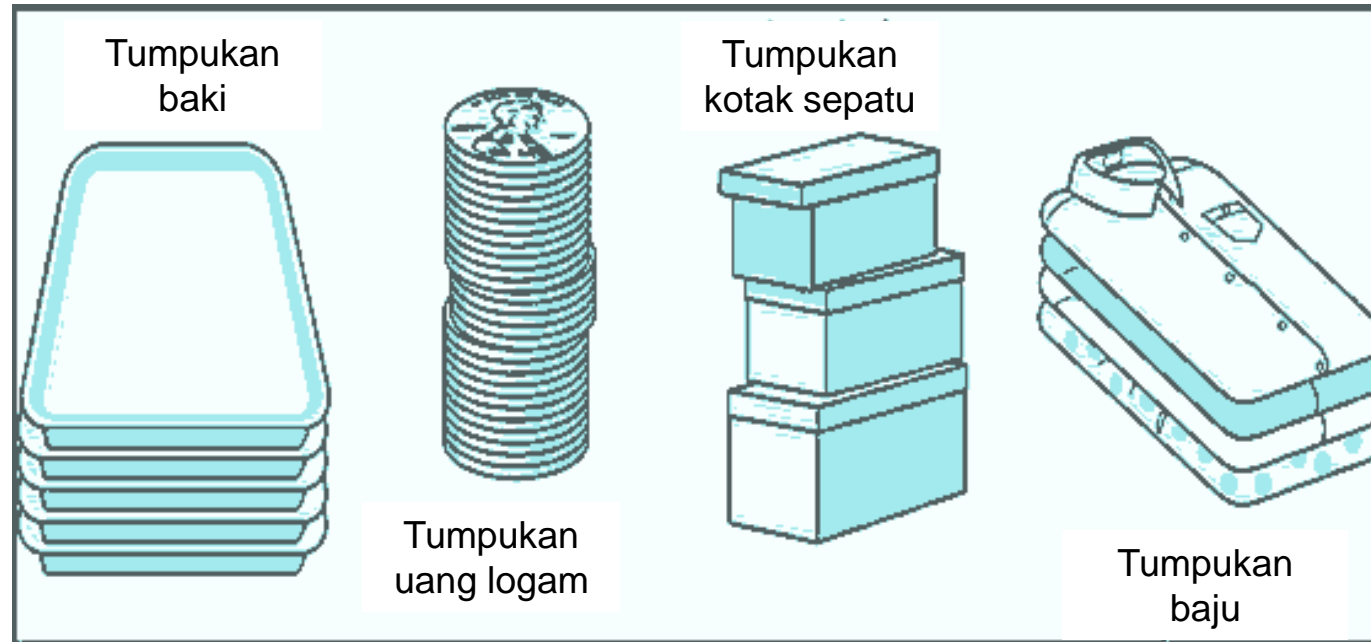


Jenis-Jenis Struktur Data



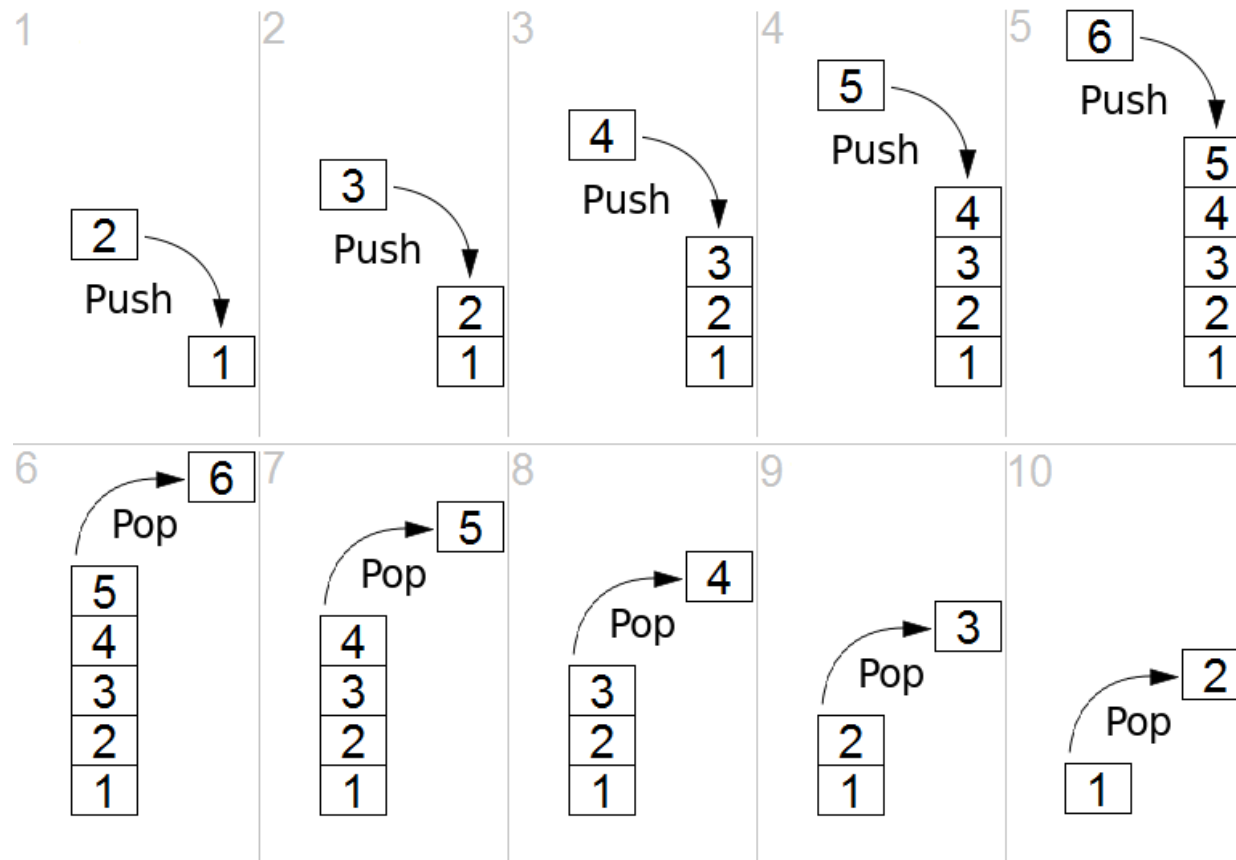
Stack (Tumpukan)

- Salah satu konsep penggunaan array atau linked list
- Struktur data untuk menyimpan data dengan *order* **LIFO (Last In First Out)**. Maksudnya, setiap data yang terakhir masuk, itu yang akan di panggil lebih dulu atau keluar lebih dulu
 - Atau bisa juga disebut **FILO (First In Last Out)**



Operasi pada Stack

- **Stack / Push:** insert elemen ke dalam stack
- **Unstack / Pop:** hapus elemen yang terakhir ditambahkan ke dalam stack



- Stack hanya mempunyai satu *end/pointer* yaitu **TOP** (elemen teratas dalam stack tersebut)
- Item dapat di-**push** atau di-**pop** menggunakan **TOP**
- **TOP = -1** stack kosong

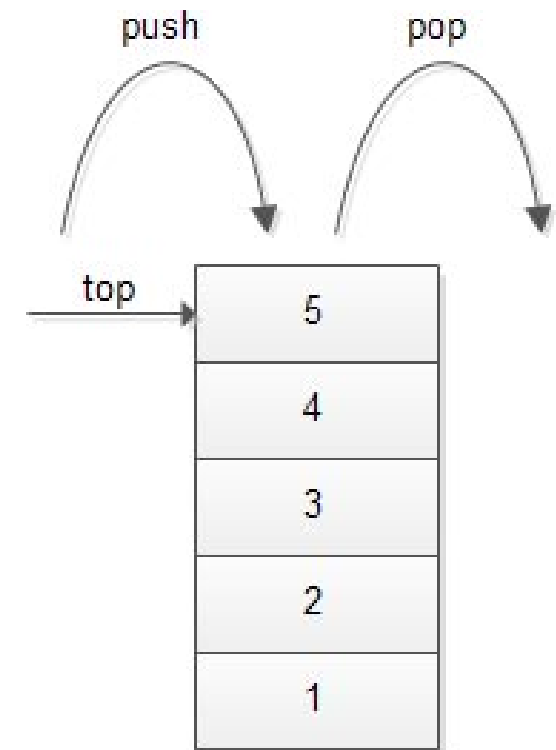
Implementasi Stack menggunakan Array

1. Deklarasikan stack **STACK** sebagai Array dengan ukuran **N** (kapasitas dari stack) dan **TOP** sebagai indeks array dari elemen paling atas stack tersebut

```
#define N 10 //konstanta  
int STACK[N], TOP;
```

2. Buat fungsi untuk men-**display**, mem-**push**, dan mem-**pop** elemen stack

```
void display(int stack[])  
void push(int stack[],int item)  
void pop(int stack[])
```



Implementasi Stack menggunakan Array - `display()`

Fungsi untuk menampilkan isi dari stack

```
void display(int stack[])
{
    if(TOP >= 0){
        printf("Isi STACK : \n");
        for(int i = TOP; i >= 0; i--)
        {
            printf("\n%d", stack[i]);
        }
    }
    else{
        printf("STACK kosong .\n");
    }

    printf("\n\n");
}
```

Jika nilai **TOP** ≥ 0 artinya elemen teratas dari array stack ada di index 0, 1, 2, dst. Berarti stack tersebut ada isinya.

TOP = -1 artinya stack kosong karena index array dimulai dari 0.

Implementasi Stack menggunakan Array - **push()**

Fungsi untuk menambahkan elemen ke dalam stack

```
void push(int stack[], int item)
{
    if (TOP == N-1) {
        printf("\nSTACK penuh, tidak dapat ditambahkan item baru\n");
    }
    else {
        TOP++;
        stack[TOP] = item;
    }
}
```

Jika **TOP** (elemen teratas) berada di array index **N-1** berarti stack tersebut sudah penuh


Jika belum penuh, maka:

- **TOP = TOP + 1**
- isi array stack pada **TOP** dengan item

Implementasi Stack menggunakan Array - `pop()`

Fungsi untuk menghapus elemen dari stack : **LIFO (Last In First Out)** – yang dihapus adalah elemen di indeks **TOP**

```
void pop(int stack[])
{
    if(TOP == -1){
        printf("STACK sudah kosong.\n");
    }
    else{
        int deletedItem = stack[TOP];
        TOP--;
        printf("%d telah terhapus\n", deletedItem);
    }
}
```



Jika stack tidak kosong,
maka **TOP = TOP-1**

Apakah elemen yang dihapus yaitu `deletedItem` masih ada di array?

Jawab: masih, yang kita ubah-ubah hanya **TOP**, ketika kita **display()** tetap akan terbaca sampai **TOP**, ketika **push()** pun elemen baru akan menempa elemen yang tadinya sudah di **pop()**

Implementasi Stack menggunakan Array – Misal kita buat program yang menampilkan menu sehingga user bisa memilih operasi yang akan dilakukan

```
#include <stdio.h>
#include <stdlib.h>

#define N 10
int STACK[N], TOP;

void display(int stack[])
void push(int stack[],int item)
void pop(int stack[])
```

```
int main()
{
    TOP = -1;
    int choice = 0;
    do
    {
        printf("Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                display(STACK);
                break;
            case 2:
                printf("Masukan Item untuk Ditambahkan :");
                int ITEM = 0;
                scanf("%d",&ITEM);
                push(STACK,ITEM);
                break;
            case 3:
                pop(STACK);
                break;
            case 4:
                printf("\nKELUAR ");
                break;
            default:
                printf("\nPilihan Tidak Valid.");
        }
    }
    while(choice != 4);
    return 0;
}
```

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :1
STACK kosong .

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :2

Masukan Item untuk Ditambahkan :12

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :2

Masukan Item untuk Ditambahkan :13

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :2

Masukan Item untuk Ditambahkan :56

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :1

Isi STACK :

56

13

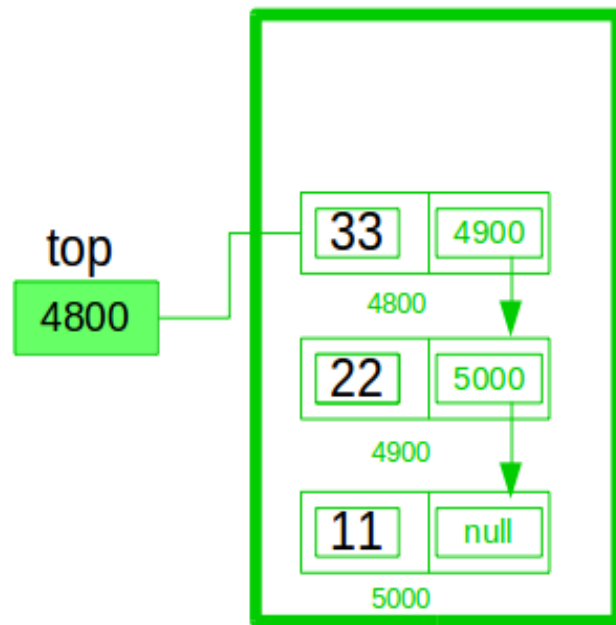
12

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :3

56 telah terhapus

Masukan Pilihan 1: Display, 2: Tambah (PUSH), 3: Hapus (POP), 4: Exit :

Implementasi Stack menggunakan Linked List



Stack menggunakan single linked list yang memiliki 3 elemen dan **TOP** (elemen teratas) mempunyai alamat **4800**

Implementasi Stack menggunakan Linked List

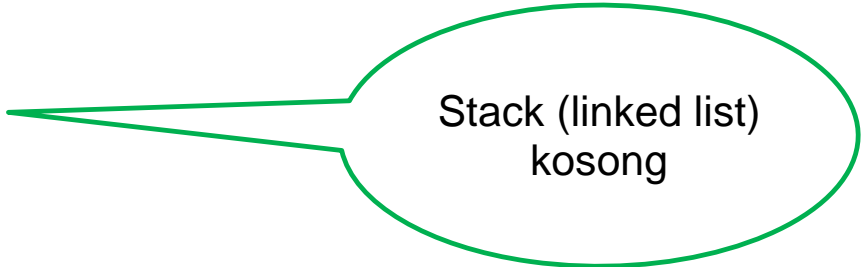
1. Deklarasikan elemen dari stack dengan mendefinisikan **node structure** linked list dan **TOP** dari stack

```
struct node
{
    int data;
    struct node *next;
};

typedef struct node* item;
item top;
```

2. Inisialisasi stack dengan membuat pointer head dari stack tersebut menunjuk ke NULL

```
void initialize()
{
    top = NULL;
}
```



Stack (linked list)
kosong

Implementasi Stack menggunakan Linked List - **push()**

Fungsi untuk menambahkan elemen baru ke stack

```
void push(int value)
{
    item new_node;
    new_node = (item)malloc(sizeof(struct node));
    new_node->data = value;
    new_node->next = top;
    top = new_node;
}
```

1. Buat node baru **new_node**
2. Masukkan data dari node **new_node**
3. Tunjuk pointer **next** dari node **new_node** ke **TOP** sebelumnya
4. Jadikan node **new_node** sebagai **TOP** yang baru

Implementasi Stack menggunakan Linked List - `pop()`

Fungsi untuk menghapus elemen dari stack : **LIFO (Last In First Out)**

```
void pop()
{
    item tmp;
    tmp = top;
    top = top->next;
    free(tmp);
}
```

1. Buat temporary node `tmp` yang menunjuk ke `TOP`
2. Jadikan node setelah `TOP` sebagai `TOP` yang baru
3. Hapus/bebaskan memory dari temporary node `tmp`

Implementasi Stack menggunakan Linked List - `display()`

Fungsi untuk menampilkan isi dari stack

```
void display(item top)
{
    if(top == NULL)
    {
        printf("Stack kosong\n");
    }
    else
    {
        printf("%d\n", top->data);
        display(top->next);
    }
}
```

Fungsi untuk menampilkan isi dari node TOP

```
int DisplayTop()
{
    return top->data;
}
```

Implementasi Stack menggunakan Linked List

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *next;
};

typedef struct node* item;
item top;

void initialize();
void push(int value);
void pop();
void display(item head);
int DisplayTop();

int main()
{
    initialize();
    push(10);
    push(20);
    push(30);
    push(40);
    printf("Top dari stack adalah %d\n", DisplayTop());
    pop();
    printf("Top dari stack setelah pop adalah %d\n", DisplayTop());
    display(top);
    return 0;
}
```

Aplikasi Stack dalam Dunia Nyata

- Fitur undo-redo pada editor seperti text editor, photoshop, dll
- Fitur backward dan forward pada web browser
- Backtracking pada game



TERIMA KASIH