



Struktur Data
Pendidikan Teknik Informatika
Universitas Negeri Padang
TIM DOSEN ©2022

JOBSHEET (JS-04)

Stack

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep stack dalam menyelesaikan kasus.

HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

A. Stack (Tumpukan)

Stack (tumpukan) adalah struktur data linier yang mengikuti prinsip **LIFO (Last-In-First-Out)**. Stack memiliki satu ujung, ini hanya berisi satu penunjuk yang menunjuk ke elemen paling atas dari tumpukan. Setiap kali sebuah elemen ditambahkan di tumpukan, itu ditambahkan di atas tumpukan, dan elemen hanya dapat dihapus dari tumpukan. Dengan kata lain, *tumpukan dapat didefinisikan sebagai wadah di mana penyisipan dan penghapusan dapat dilakukan dari satu ujung yang dikenal sebagai bagian atas tumpukan.*

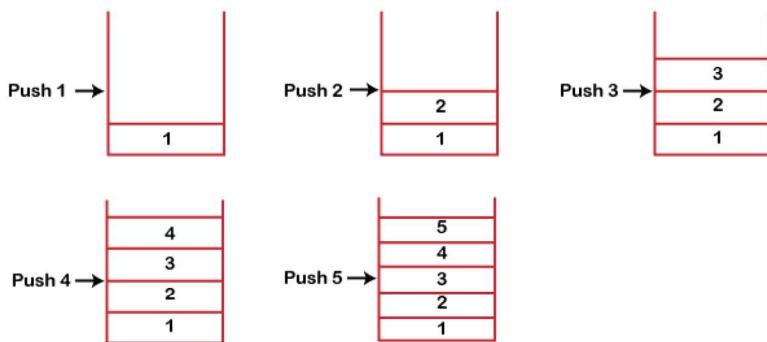
1. Beberapa poin penting terkait dengan tumpukan

1. Disebut sebagai tumpukan karena berperilaku seperti tumpukan dunia nyata, tumpukan buku, dll.

- Stack adalah tipe data abstrak dengan kapasitas yang telah ditentukan sebelumnya, yang berarti dapat menyimpan elemen dengan ukuran terbatas.
- Merupakan struktur data yang mengikuti beberapa urutan untuk menyisipkan dan menghapus elemen, dan urutan itu dapat berupa LIFO atau FILO.

2. Bekerja dari Stack

Stack bekerja pada pola LIFO. Seperti yang dapat kita amati pada gambar di bawah ini, ada lima blok memori dalam tumpukan; oleh karena itu, ukuran tumpukan adalah 5. Misalkan kita ingin menyimpan elemen dalam tumpukan dan anggap tumpukan itu kosong. Tumpukan ukuran 5 seperti yang ditunjukkan di bawah ini di mana satu per satu elemen didorong hingga tumpukan menjadi penuh.



Tumpukan penuh karena ukuran tumpukan adalah 5. Dalam kasus di atas, kita dapat mengamati bahwa tumpukan itu bergerak dari atas ke bawah ketika kita memasukkan elemen baru dalam tumpukan. Tumpukan akan diisi dari bawah ke atas.

Saat kita melakukan operasi penghapusan pada tumpukan, hanya ada satu cara untuk masuk dan keluar karena ujung yang lain ditutup. Ini mengikuti pola LIFO, yang berarti bahwa nilai yang dimasukkan pertama akan dihapus terakhir. Dalam kasus di atas, nilai 1 dimasukkan terlebih dahulu, sehingga hanya akan dihapus setelah penghapusan semua elemen lainnya.

3. Operasi Tumpukan Standar

Berikut ini adalah beberapa operasi umum yang diterapkan pada stack:

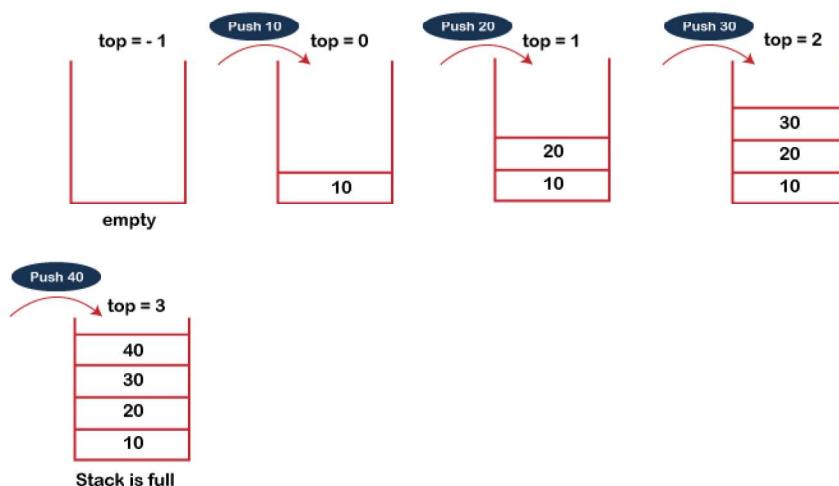
- push():** Ketika kita menyisipkan elemen dalam tumpukan maka operasi ini dikenal sebagai push. Jika stack penuh maka terjadi kondisi overflow.

2. **pop()**: Saat menghapus elemen dari tumpukan, operasi ini dikenal sebagai pop. Jika tumpukan kosong berarti tidak ada elemen di tumpukan, keadaan ini dikenal sebagai keadaan underflow.
3. **isEmpty()**: Ini menentukan apakah tumpukan kosong atau tidak.
4. **isFull()**: Ini menentukan apakah tumpukan penuh atau tidak.'
5. **peek()**: Ini mengembalikan elemen pada posisi yang diberikan.
6. **count()**: Ini mengembalikan jumlah total elemen yang tersedia dalam tumpukan.
7. **change()**: Ini mengubah elemen pada posisi yang diberikan.
8. **display()**: Ini mencetak semua elemen yang tersedia di tumpukan.

4. Operasi Push (Dorong)

Langkah-langkah yang terlibat dalam operasi PUSH diberikan di bawah ini:

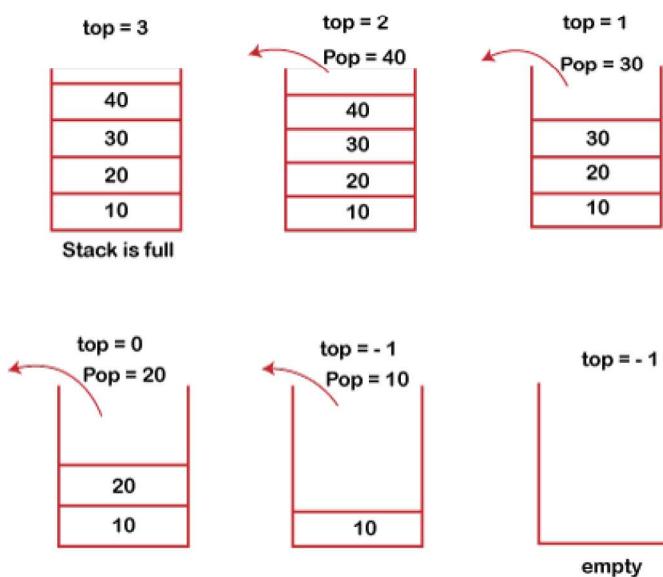
1. Sebelum memasukkan elemen ke dalam tumpukan, diperiksa apakah tumpukan sudah penuh.
2. Jika mencoba memasukkan elemen ke dalam stack, dan stack sudah penuh, maka terjadi kondisi **overflow**.
3. Saat menginisialisasi tumpukan, menetapkan nilai atas sebagai -1 untuk memeriksa apakah tumpukan kosong.
4. Ketika elemen baru didorong ke dalam tumpukan, pertama, nilai top bertambah, yaitu $\text{top}=\text{top}+1$, dan elemen akan ditempatkan pada posisi baru **top** .
5. Elemen akan dimasukkan sampai kita mencapai ukuran **maksimal** tumpukan.



5. Operasi POP

Langkah-langkah yang terlibat dalam operasi POP diberikan di bawah ini:

1. Sebelum menghapus elemen dari tumpukan, diperiksa apakah tumpukan kosong.
2. Jika kita mencoba untuk menghapus elemen dari tumpukan kosong, maka terjadi kondisi ***underflow***.
3. Jika tumpukan tidak kosong, pertama-tama kita mengakses elemen yang ditunjuk oleh bagian **atas**
4. Setelah operasi pop dilakukan, bagian atas dikurangi dengan 1, yaitu, **top=top-1** .



B. Implementasi array dari Stack

Dalam implementasi array, stack dibentuk dengan menggunakan array. Semua operasi mengenai tumpukan dilakukan menggunakan array. Mari kita lihat bagaimana setiap operasi dapat diimplementasikan pada stack menggunakan struktur data array.

1. Menambahkan elemen ke tumpukan (operasi push)

Menambahkan elemen ke bagian atas tumpukan disebut sebagai operasi push. Operasi push melibatkan dua langkah berikut.

- a. Tingkatkan variabel Top sehingga sekarang dapat merujuk ke lokasi memori berikutnya.
- b.Tambahkan elemen pada posisi atas yang bertambah. Ini disebut sebagai menambahkan elemen baru di bagian atas tumpukan.

Stack overflow ketika kita mencoba memasukkan elemen ke dalam stack yang terisi penuh oleh karena itu, fungsi utama kita harus selalu menghindari kondisi stack overflow.

Algoritma:

```
begin
    if top = n then stack full
    top = top + 1
    stack (top) := item;
end
```

2. Implementasi algoritma push dalam bahasa C

```
void push (int val,int n) //n is size of the stack
{
    if (top == n )
        printf("\n Overflow");
    else
    {
        top = top +1;
        stack[top] = val;
    }
}
```

3. Penghapusan elemen dari tumpukan (Operasi Pop)

Penghapusan elemen dari atas tumpukan disebut operasi pop. Nilai atas variabel akan berkurang 1 setiap kali item dihapus dari tumpukan. Elemen paling atas dari tumpukan disimpan dalam variabel lain dan kemudian bagian atas dikurangi dengan 1. operasi mengembalikan nilai yang dihapus yang disimpan dalam variabel lain sebagai hasilnya. Kondisi underflow terjadi ketika mencoba menghapus elemen dari tumpukan yang sudah kosong.

Algoritma:

```
begin
    if top = 0 then stack empty;
    item := stack(top);
    top = top - 1;
end;
```

Implementasi algoritma POP menggunakan bahasa C :

```
int pop ()
{
    if (top == -1)
    {
        printf("Underflow");
        return 0;
    }
    else
    {
        return stack[top - 1];
    }
}
```

4. Mengunjungi setiap elemen tumpukan (operasi Peek)

Operasi mengintip melibatkan pengembalian elemen yang ada di bagian atas tumpukan tanpa menghapusnya. Kondisi underflow dapat terjadi jika kita mencoba mengembalikan elemen teratas dalam stack yang sudah kosong.

Algoritma:**PEEK (STACK, TOP)**

```
Begin
    if top = -1 then stack empty
    item = stack[top]
    return item
End
```

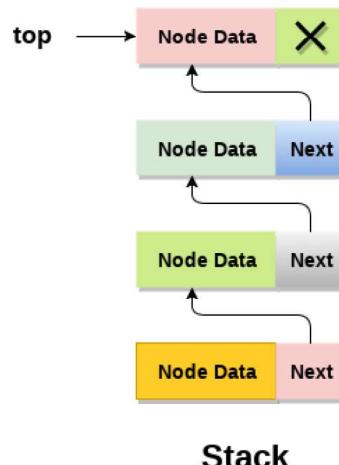
Implementasi algoritma Peek dalam bahasa C :

```
int peek()
{
    if (top == -1)
    {
        printf("Underflow");
        return 0;
    }
    else
    {
        return stack [top];
    }
}
```

C. Implementasi linked list dari stack

Alih-alih menggunakan array, kita juga dapat menggunakan linked list untuk mengimplementasikan tumpukan. Linked list mengalokasikan memori secara dinamis. Namun, kompleksitas waktu di kedua skenario sama untuk semua operasi yaitu push, pop dan peek.

Dalam implementasi linked list dari tumpukan, node dipertahankan secara tidak berurutan dalam memori. Setiap node berisi pointer ke node penerus langsung di stack. Stack dikatakan overflow jika space yang tersisa di memory heap tidak cukup untuk membuat node.

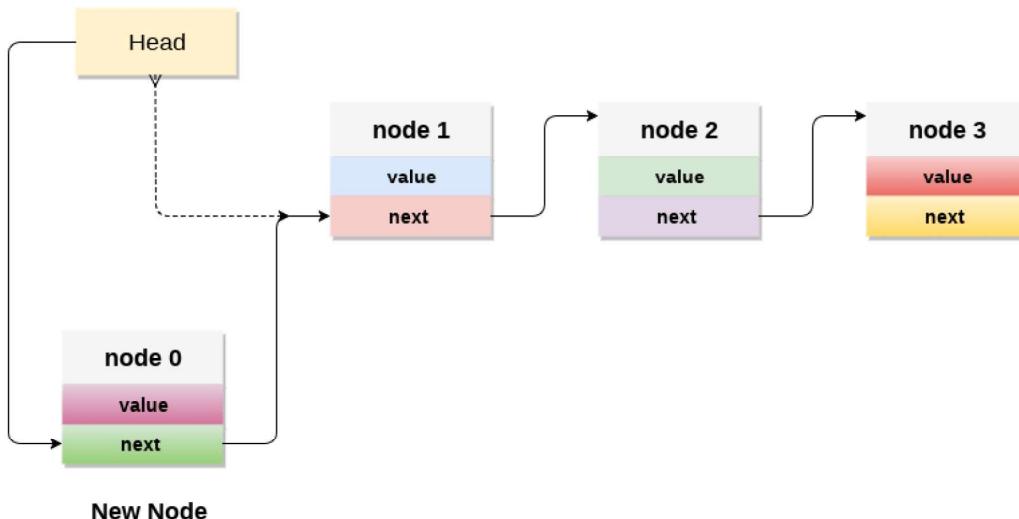
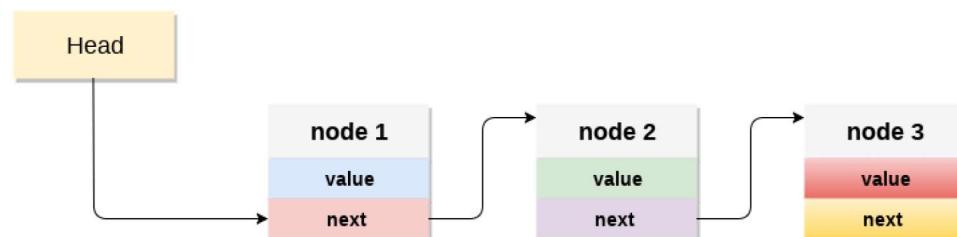


Node paling atas di tumpukan selalu berisi nol di bidang alamatnya. Mari kita bahas caranya, setiap operasi dilakukan dalam implementasi linked list dari tumpukan.

1. Menambahkan node ke tumpukan (Operasi push)

Menambahkan node ke stack disebut sebagai operasi **push**. Mendorong elemen ke tumpukan dalam implementasi linked list berbeda dari implementasi array. Untuk mendorong elemen ke tumpukan, langkah-langkahnya sebagai berikut :

- a. Buat node terlebih dahulu dan alokasikan memori untuk itu.
- b. Jika daftar kosong maka item akan didorong sebagai node awal daftar. Ini termasuk menetapkan nilai ke bagian data dari node dan menetapkan null ke bagian alamat dari node.
- c. Jika sudah ada beberapa node dalam daftar, maka kita harus menambahkan elemen baru di awal daftar (agar tidak melanggar properti tumpukan). Untuk tujuan ini, tetapkan alamat elemen awal ke bidang alamat node baru dan buat node baru, node awal dari daftar.



Implementasi dalam C:

```
void push ()  
{  
    int val;  
    struct node *ptr =(struct node*)malloc(sizeof(struct node));  
    if(ptr == NULL)  
    {  
        printf("not able to push the element");  
    }  
    else  
    {  
        printf("Enter the value");  
        scanf("%d",&val);  
        if(head==NULL)  
        {  
            ptr->val = val;  
            ptr -> next = NULL;  
            head=ptr;  
        }  
        else  
        {  
            ptr->val = val;  
            ptr->next = head;  
            head=ptr;  
        }  
        printf("Item pushed");  
    }  
}
```

2. Menghapus node dari stack (operasi POP)

Menghapus sebuah node dari atas tumpukan disebut sebagai operasi **pop**. Menghapus sebuah node dari implementasi linked list dari tumpukan berbeda dengan yang ada dalam implementasi array. Untuk mengeluarkan elemen dari tumpukan, kita harus mengikuti langkah-langkah berikut:

- a. **Periksa kondisi underflow:** Kondisi underflow terjadi ketika mencoba untuk mengeluarkan dari tumpukan yang sudah kosong. Tumpukan akan kosong jika penunjuk kepala daftar menunjuk ke nol.
- b. **Sesuaikan penunjuk kepala yang sesuai:** Dalam tumpukan, elemen hanya muncul dari satu ujung, oleh karena itu, nilai yang disimpan dalam penunjuk kepala harus dihapus dan node harus dibebaskan. Node berikutnya dari node kepala sekarang menjadi node kepala.

Implementasi dalam C:

```
void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");
    }
}
```

3. Menampilkan node (Traversing)

Menampilkan semua node dari tumpukan perlu melintasi semua node dari linked list yang diatur dalam bentuk tumpukan. Untuk tujuan ini, kita perlu mengikuti langkah-langkah berikut.

- a. Salin penunjuk kepala ke penunjuk sementara.
- b. Pindahkan penunjuk sementara melalui semua node daftar dan cetak bidang nilai yang dilampirkan ke setiap node.

Implementasi dalam C:

```
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

LATIHAN

1. Implementasi array dari Stack

```
/* Nama File : implementasi array dari Stack
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/
```

```

#include <stdio.h>
int stack[100],i,j,choice=0,n,top=-1;
void push();
void pop();
void show(); int main ()
{
    printf("*****Stack operations using
array*****\n");

printf("*****\n");
;
    printf("Enter the number of elements in the stack :
");
    scanf("%d",&n);

printf("\n-----
\n");
while(choice != 4)
{
    printf("Chose one from the below options...\n");
    printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
    printf("\n Enter your choice \n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
        {
            push();
            break;
        }
        case 2:
        {
            pop();
            break;
        }
        case 3:
        {
            show();
            break;
        }
        case 4:
        {
            printf("Exiting....");
    }
}
}

```

```
        break;
    }
default:
{
    printf("Please Enter valid choice ");
}
};

}

void push ()
{
    int val;
    if (top == n )
    printf("\n Overflow");
    else
    {
        printf("Enter the value?");
        scanf("%d",&val);
        top = top +1;
        stack[top] = val;
    }
}

void pop ()
{
    if(top == -1)
    printf("Underflow");
    else
    top = top -1;
}
void show()
{
    for (i=top;i>=0;i--)
    {
        printf("%d\n",stack[i]);
    }
    if(top == -1)
    {
        printf("Stack is empty");
    }
}
```

2. Implementasi linked list dari Stack

```
/* Nama File : implementasi linked list dari Stack
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/\n\n#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
    int val;
    struct node *next;
};
struct node *head;\n\nint main ()
{
    int choice=0;
    printf("\n*****Stack operations using linked
list*****\n");
    printf("\n-----\n");
    while(choice != 4)
    {
        printf("\n\nChose one from the below
options...\n");
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\n Enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
```

```
{  
    display();  
    break;  
}  
case 4:  
{  
    printf("Exiting....");  
    break;  
}  
default:  
{  
    printf("Please Enter valid choice ");  
}  
};  
}  
}  
}  
void push ()  
{  
    int val;  
    struct node *ptr = (struct node*)malloc(sizeof(struct  
node));  
    if(ptr == NULL)  
    {  
        printf("not able to push the element");  
    }  
    else  
    {  
        printf("Enter the value : ");  
        scanf("%d",&val);  
        if(head==NULL)  
        {  
            ptr->val = val;  
            ptr -> next = NULL;  
            head=ptr;  
        }  
        else  
        {  
            ptr->val = val;  
            ptr->next = head;  
            head=ptr;  
        }  
        printf("Item pushed");  
    }  
}
```

```

        }

    }

void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");

    }
}

void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}

```

TUGAS

Buatlah program untuk menentukan apakah sebuah kalimat yang diinputkan adalah sebuah palindrom atau bukan (**program dibuat dengan menggunakan stack**). Palindrom adalah kalimat yang jika dibaca dari depan dan dari belakang, maka bunyinya sama.

Contoh output program :

```
Masukkan kalimat : sugus
Kalimat tersebut adalah palindrom.
```

DAFTAR PUSTAKA

- Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.
- Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.
- Liem, Inggriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.
- Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.
- Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com