# STACK dan QUEUE

## PENGANTAR

### Definisi

Stack disebut juga tumpukan dimana data hanya dapat dimasukkan dan diambil **dari satu sisi**.
Karena itu, stack bersifat **LIFO(Last In First Out).**

Operasi yang dapat dilakukan stack adalah:

1. Menambah (push)
2. Mengambil (pop)
3. megecek apakah stack penuh (isFull)
4. mengecek apakah stack kosong (isEmpty)
5. membersihkan stack (clear).
6. Mencetak isi stack (print)

### Operasi-operasi stack

Saat ini, kita akan mencoba membuat stack dan operasi-operasi yang dapat dilakukannya.

1. Mendefinisikan stack dengan menggunakan struct

```
typedef struct stack //       Mendefinisikan stack dengan menggunakan struct
{
int top;
char data [15][20];      // menampung 15 data dengan jumlah string max 20 huruf
};
```

2. Mendefinisikan max_stack untuk maksimum isi stack

```
#define max_stack 15
```

3. Membuat variable array sebagai implementasi stack

```
stack tumpuk;
```

4. Mendeklarasikan operasi-operasi/fungsi yang dapat dilakukan stack.
   a. Push (menginputkan data pada stack)

```
void push(char d[20])
{
        tumpuk.top++;
        strcpy(tumpuk.data[tumpuk.top],d);
        printf("data berhasil dimasukkan");
}
```

   b. Pop (mengambil data pada stack)

```
void pop()
{
        printf ("data %s terambil",tumpuk.data[tumpuk.top]);
        tumpuk.top--;
}
```

c. IsFull(megecek apakah stack penuh)

```
int isFull()
{
        if (tumpuk.top==max_stack-1)
                return 1;
        else
                return 0;
}
```

d. isEmpty(mengecek apakah stack kosong)

```
int isEmpty()
{
        if (tumpuk.top==-1)
                return 1;
        else
                return 0;
}
```

e. clear (membersihkan seluruh isi stack)

```
void clear()
{
        tumpuk.top=-1;
        printf("semua data terhapus.");

}
```

f. print (mencetak seluruh isi stack)

```
void print()
{
        for (int i=tumpuk.top;i>=0;i--)
                printf ("%s\n",tumpuk.data[i]);
}
```

**QUEUE**

Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain. Karena itu, queue bersifat FIFO(First In First Out).

Saat ini, kita akan mencoba membuat queue dan operasi-operasi yang dapat dilakukannya.

Hal-hal yang perlu dilakukan untuk membuat queue yaitu

1. Mendefinisikan queue dengan menggunakan struct dimana kita perlu menggunakan variable head dan tail sebagai penanda pada stack.

```
typedef struct queue //       Mendefinisikan queue dengan menggunakan struct
{
int head;
int tail;
char data [15][20];      // menampung 15 data dengan jumlah string max 20 huruf
};
```

2. Mendefinisikan max untuk maksimum isi queue
   #define max 15

3. Membuat variable array sebagai implementasi queue
   queue antri;

4. Mendeklarasikan operasi-operasi/fungsi yang dapat dilakukan queue.
   a. Enqueue (menginputkan data pada queue)

```
void enqueue(char d[20])
{
        antri.head=0;
        antri.tail++;
        strcpy(antri.data[antri.tail],d);
        printf("data berhasil dimasukkan");
}
```

   b. Dequeue (mengambil data dari queue)

```
void dequeue()
{
        printf ("data %s terambil",antri.data[antri.head]);
        for (int i=antri.head;i<=antri.tail;i++)
                strcpy (antri.data[i],antri.data[i+1]);
        antri.tail--;
}
```

   c. isEmpty (mengecek apakah antrian kosong)

```
int isEmpty()
{
        if (antri.tail==-1)
        {
                antri.head=-1;
                return 1;
        }
        else
                return 0;
}
```

   d. isFull (mengecek apakah antrian penuh)

```
int isFull()
{
        if (antri.tail==max-1)
                return 1;
        else
                return 0;
}
```

   e. clear (membersihkan seluruh isi antrian)

```
void clear()
{
        antri.head=antri.tail=-1;
        printf("semua data terhapus.");

}
```

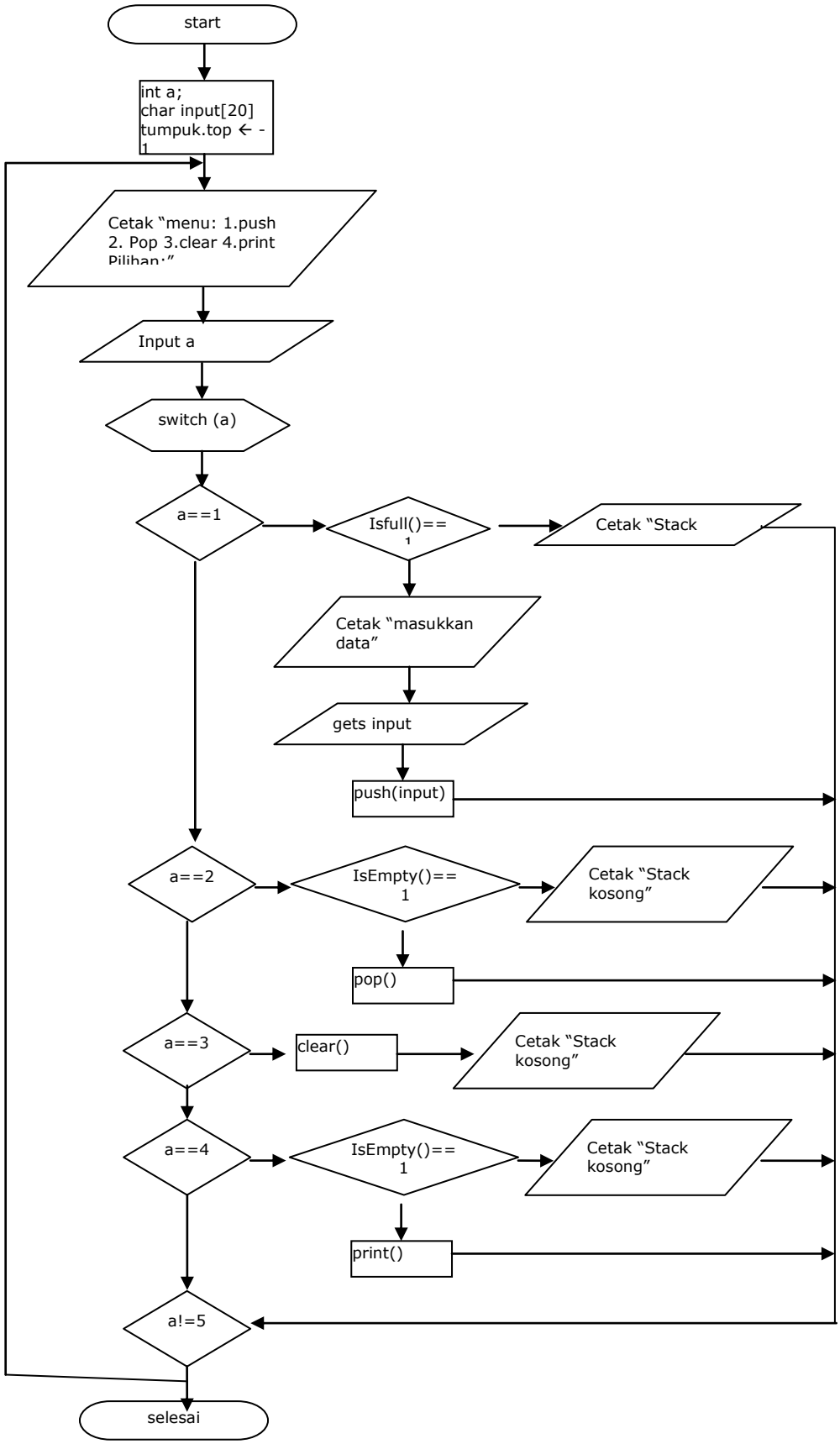
f. Print(mencetak seluruh isi antrian)

```
void print()
{
        for (int i=0;i<=antri.tail;i++)
                printf ("%s\n",antri.data[i]);
}
```

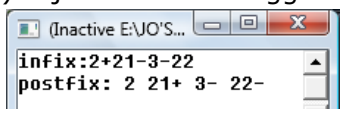
# SOAL LATIHAN:

## Guided STACK:

1. Dari flowchart dibawah ini, buatlah fungsi utama untuk menjalankan stack diatas dengan menggunakan menu.

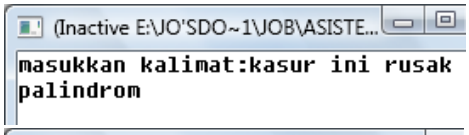


2. Buatlah program pengubah infix ke postfix dengan asumsi operator hanya tambah(+) dan kurang(-) saja. buatlah menggunakan stack!



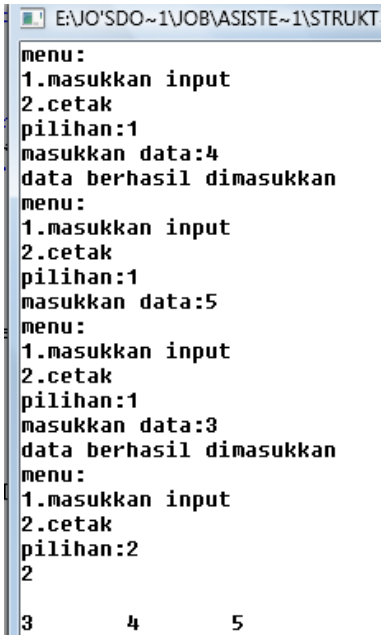

## Unguided STACK:

1. Buatlah program pengecekan bilangan palindrom menggunakan stack!





2. Buatlah program pengurutan menggunakan stack! Sehingga input yang kita masukkan selalu urut.
   Hint: buatlah menggunakan 2buah stack!




## TAKEHOME STACK!

1. Kembangkan soal unguided nomor 2 untuk mengurutkan kata!
2. Kembangkan soal guided nomor 2 dengan operator bertingkat! Gunakan 2 buah stack. Contoh tersedia di bawah ini:

   Infix: a+b*c-d

   Stack (kosong) dan Postfik (kosong)
   Scan a
   Postfik: a
   Scan +
   Stack: +
   Scan b
   Postfik: ab
   Scan *, karena ToS (+) < *, maka add ke Stack
   Stack: +*
   Scan c
   Postfik: abc
   Scan –, karena * > -, maka pop Stack, dan add ke Postfik
   Stack: +
   Postfik: abc*
   Karena + <= -, maka pop Stack, dan add ke Postfik, karena Stack kosong, maka push – ke stack
   Stack: -
   Postfik: abc*+
   Scan d
   Postfik: abc*+d
   Karena sudah habis, push ToS stack ke Posfix

# Kunci jawaban soal GUIDED STACK:

No.1:

Fungsi utamanya:

```
void main()
{
	int a;
	char input[20];
	tumpuk.top=-1;
do {
	printf ("menu:\n1.push\n2.Pop\n3.clear\n4.print\n5.terminate\npilihan:");
	scanf ("%i",&a);
	fflush(stdin);
	switch (a)
	{
	case 1:
		if (isFull()==1)
			printf("stack penuh.\n");
		else
			{
			printf("masukkan data:");
			gets(input);
			fflush(stdin);
			push(input);
			}
		break;
	case 2:
		if (isEmpty()==1)
			{
				printf("stack kosong");
			}
		else
			{
				pop();
			}
		break;
	case 3:
			clear();
			printf("stack kosong\n");
			break;
	case 4:
		if (isEmpty()==1)
			printf("stack kosong");
		else
			print();
	}
}
  while (a!=5);
}
```

No.2:

**Jangan lupa sesuaikan fungsi pop, dan push yang sudah ada dengan kontex penggunaannya dan juga tipe datanya!**

```
void main()
{
	int a;
	char input[20];
	tumpuk.top=-1;
	printf("infix:");
	gets(input);
	a=strlen(input);
	printf("%c",input[0]);
	for (int i=1;i<a;i++)
	{
		if (input[i]=='+' || input[i]=='-' )
		{
		push(input[i]);
		}
		else
		{
			printf("%c",input[i]);
			if(input[i+1]=='+' || input[i+1]=='-'|| i+1==a)
			{
				pop();
			}
		}

	}
}
```

**Latihan Terbimbing Queue**

1. Buatlah fungsi utama untuk menjalankan queue pada modul dengan menggunakan menu. (untuk alurnya, hampir sama dengan flowchart untuk stack diatas. Hanya saja fungsi diganti dengan fungsi-fungsi queue.)
2. Carilah total, rata-rata, nilai terkecil, nilai terbesar dari nilai yang telah di inputkan pada queue

**Latihan Mandiri Kelas Queue**

1. Tambahkanlah function untuk mencari suatu elemen dalam queue & stack
2. Tambahkan function untuk mengedit suatu elemen dalam queue & stack
3. Buatlah suatu fungsi untuk mencetak elemen stack ganjil
4. Buatlah suatu fungsi untuk mencetak elemen queue genap

**Latihan Mandiri Rumah /Tes Queue**

1. Buatlah program untuk membuat kalkulator scientific!
2. Buatlah program untuk membalikkan kata yang berguna untuk membalik susunan elemen dalam queue Q dengan menggunakan bantuan sebuah stack .
3. Buatlah program untuk menentukan apakah kata yang dimasukan merupakan palindrome atau bukan.