

Struktur Data Stack

Universitas Muslim Indonesia

Oleh
Lutfi Budi Ilmawan

Stack / Tumpukan

- Merupakan *list* di mana penambahan dan pengambilan elemen hanya dilakukan pada satu sisi yang disebut top (puncak) dari stack.
- Menggunakan aturan LIFO (Last In First Out), yaitu elemen yang terakhir masuk akan pertama kali diambil atau dilayani.
- Top adalah posisi teratas dari elemen pada stack.

Operasi Dasar

Operasi utama:

- Push : proses penambahan elemen baru pada stack
- Pop : proses pengambilan elemen pada stack

Operasi penunjang:

- CreateStack : membuat stack baru
- EmptyStack : proses pengecekan pada sebuah stack bahwa stack tersebut telah terisi atau tidak
- FullStack : proses pengecekan pada sebuah stack bahwa stack tersebut telah mencapai kapasitasnya
- DestroyStack : mengosongkan isi dari stack

Contoh

Stack S dengan kapasitas $n = 5$



Top = -1

Contoh

Push(S, a)



Top = 0

Contoh

Push(S, b)



Top = 1

Contoh

Push(S, c)



Top = 2

Contoh

Pop(S, x)



Top = 1

x = c

Contoh

Push(S, d)



Top = 2

Contoh

Push(S, e)



Top = 3

Contoh

Pop(S, x)



Top = 2

x = e

Contoh

Pop(S, x)



Top = 1

x = d

CreateStack(Stack)

- **Function** Initialize Stack to an empty state
- **Input** None
- **Preconditions** None
- **Output** Stack
- **Postconditions** Stack is empty

DestroyStack(Stack)

- **Function** Removes all elements on stack, leaving the stack empty
- **Input** Stack
- **Preconditions** Stack has been created
- **Output** Stack
- **Postconditions** Stack is empty

EmptyStack(Stack)

- **Function** Test whether Stack is empty
- **Input** Stack
- **Preconditions** Stack has been created
- **Output** True or False
- **Postconditions** Stack is empty

FullStack(Stack)

- **Function** Test whether Stack is full
- **Input** Stack
- **Preconditions** Stack has been created
- **Output** True or False
- **Postconditions** Stack is full

Push(Stack, NewElement)

- **Function** Add new element to the top of Stack
- **Input** Stack, NewElement
- **Preconditions** Stack is not full
- **Output** Stack
- **Postconditions** Stack = original Stack with new element added on top

Pop(Stack, PoppedElement)

- **Function** Removes top element from Stack and returns it in PoppedElement
- **Input** Stack
- **Preconditions** Stack is not empty
- **Output** Stack, PoppedElement
- **Postconditions** Stack = original Stack with top element removed
PoppedElement = top element of original stack

Deklarasi

```
const int Max=10;  
struct StackType{  
    char Elements[Max];  
    int Top;  
};  
StackType stack;
```

CreateStack dan DestroyStack

```
void CreateStack(StackType &Stack){  
    Stack.Top = -1;  
}
```

EmptyStack

```
bool EmptyStack(StackType Stack){  
    return (Stack.Top == -1);  
}
```

FullStack

```
bool FullStack(StackType Stack){  
    return (Stack.Top == max-1);  
}
```

Push

```
void Push(StackType &Stack, char NewElement){  
    Stack.Top = Stack.Top + 1;  
    Stack.Elements[Stack.Top] = NewElement;  
}
```

Pop

```
void Pop(StackType &Stack, char &PoppedElement){  
    PoppedElement = Stack.Elements[Stack.Top];  
    Stack.Top      = Stack.Top - 1;  
}
```