



Modul 3

STACK ARRAY

Pratikum Informasi & Struktur Data

Asisten Informasi & Struktur Data 2022/2023

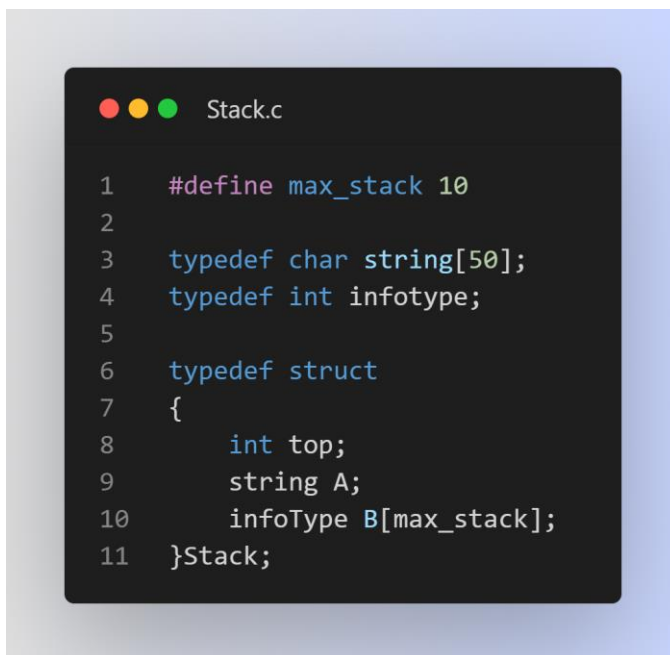
MODUL 3 – STACK ARRAY

1. Tujuan Pratikum :

- Memahami konsep stack dan penggunaan array sebagai stack
- Memahami operasi-operasi dasar stack
- Mampu mengembangkan penggunaan stack dalam Bahasa C

2. Dasar Teori

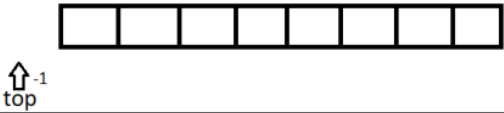
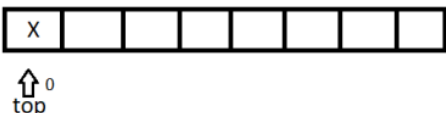
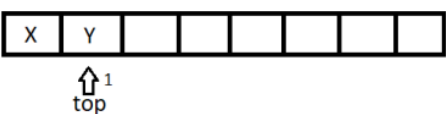
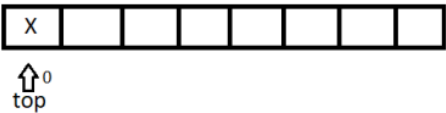
Stack (tumpukan) dapat dianalogikan sebagai kumpulan kertas yang ditumpuk berlapis – lapis. Stack memiliki 2 operasi utama, penambahan data ke stack disebut **PUSH** dan pengurangan data disebut dengan **POP**. Kedua operasi ini menggunakan konsep **LIFO**(*Last In First Out*), data terakhir yang masuk akan keluar terlebih dahulu. Berikut adalah contoh code struktur Stack array:

A screenshot of a code editor window titled "Stack.c" with a dark background and light-colored text. The code defines a stack structure with a maximum size of 10. It includes typedefs for a string (char array of 50), an infoType (int), and a struct Stack containing an integer top, a string A, and an array of infoType B of size max_stack.

```
1  #define max_stack 10
2
3  typedef char string[50];
4  typedef int infoType;
5
6  typedef struct
7  {
8      int top;
9      string A;
10     infoType B[max_stack];
11 }Stack;
```

Struktur dari suatu Stack terdiri dari top dan beberapa atribut yang dibutuhkan. Pada contoh code diatas, terdapat Stack yang terdiri dari atribut top berupa tipe data integer, atribut A bertipe string, dan atribut B yang merupakan tipe data infoType bentukan dari integer dengan array sejumlah max_stack. Struktur Stack dapat berubah-ubah sesuai dengan kebutuhan, antribut A tidak diwajibkan ada pada sebuah Stack. Pastikan sebuah Stack **WAJIB** memiliki top, dan jika diimplementasikan pada array maka **WAJIB** memiliki atribut array-nya.

Stack tidak akan terbentuk sendiri tanpa adanya inisialisasi awal. Top digunakan untuk mencatat index elemen terakhir(ujung) suatu Stack. Oleh karena itu, jika operasi **POP** dieksekusi, Stack akan mengeluarkan data dengan acuan posisi top. Begitu pula dengan operasi **PUSH**, data akan dimasukan berdasarkan posisi top. Maka dari itu top merupakan **acuan utama** dalam Stack. Untuk lebih jelas dapat dilihat ilustrasi alur Stack berikut.

STEP	KONDISI	STACK ARRAY
1	belum inisialisasi	\emptyset
2	inisialisasi	
3	PUSH(X)	
4	PUSH(Y)	
5	POP	

Indikasikan top adalah sebuah int. Ketika inisialisasi telah dilakukan, struktur Stack Array akan terbentuk pada step 2 dimana top bernilai -1. Karena nilai -1 dalam array tidak akan menunjuk elemen apapun (**INGAT**, setiap array dimulai dari index 0). Kondisi inilah yang disebut sebagai **Empty Stack**.

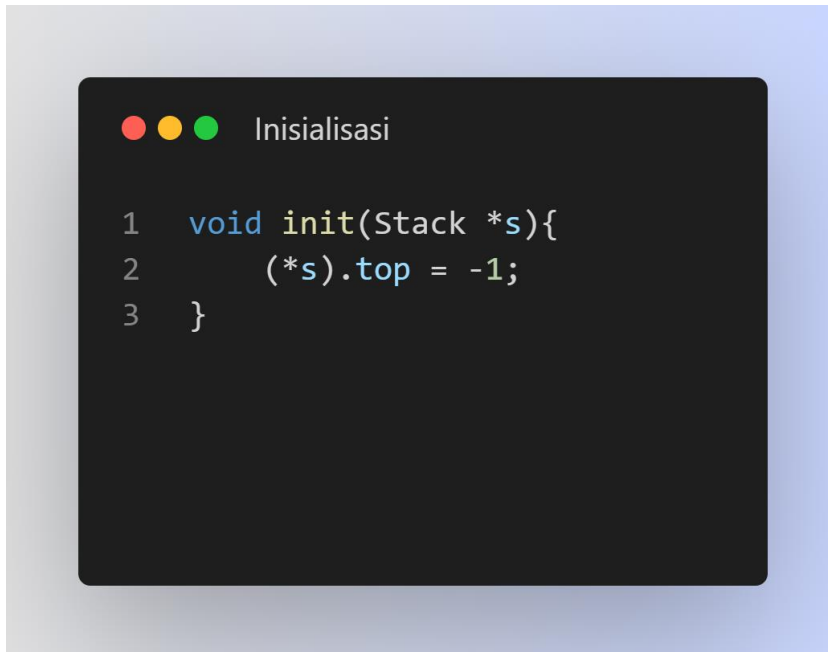
Perhatikan step 3 dimana **PUSH** telah terjadi. Data X akan dimasukan ke dalam Stack array dengan acuan top. Sebelumnya top berada pada index -1, setelah terjadinya **PUSH**, top bertambah (+1) menjadi 0 dan data X dimasukkan ke dalam Stack array pada index 0 (index dengan acuan top). Begitu pula dengan step 4.

Pada step 5 ketika **POP** terjadi, Stack akan mengeluarkan data dari index yang ditunjuk oleh top. Sebelumnya pada step 4, top berada di index 1 menunjuk data Y. Karena data inilah yang ditunjuk oleh top, maka data Y akan diambil/dihapus, sehingga top akan berkurang(-1) Kembali menunjuk index 0 sebagai data paling atas/ujung yang baru.

3. Fungsi/Prosedur dalam Stack

a) Inisialisasi

Inisialisasi merupakan hal wajib yang harus dilakukan ketika akan membuat suatu struktur Stack. Pastikan inisialisasi dipanggil untuk setiap project, supaya struktur Stack dapat terbentuk dan operasinya dapat berjalan dengan baik. Top diset menjadi -1 sehingga tidak menunjuk elemen apapun di dalam array.



b) isEmpty

Empty Stack/Stack kosong adalah sebuah kondisi dimana Stack tidak berisi elemen/data apapun. Salah satu kegunaan **isEmpty** adalah untuk mengecek apakah Stack array kosong dan dapat dilakukan **PUSH**. Selain itu juga berguna menjadi pembatas suatu Stack saat akan melakukan **POP**. Stack kosong tidak dapat dilakukan **POP** karena tidak ada data sama sekali di dalam array. Fungsi **isEmpty** memiliki nilai balikan Boolean. Secara logika, kita menganggap stack kosong dengan top bernilai -1.

Cek Kosong.c

```
1  int isEmpty(Stack s){  
2      return s.top == -1;  
3  }
```

c) isFull

isFull memiliki kegunaan untuk mengecek apakah Stack array penuh sehingga tidak dapat dilakukan operasi **PUSH** lagi. Fungsi **isFull** juga memiliki nilai balikan Boolean. Secara logika, kita menganggap Stack penuh apabila top sama dengan jumlah $\text{max_stack} - 1$. Mengapa dikurangi dengan 1, ingat array selalu dimulai dari index 0.

Cek Penuh.c

```
1  int isFull(Stack s){  
2      return s.top == max_stack - 1;  
3  }
```

d) Traversal

Traversal dapat digunakan untuk show data maupun search data. Traversal berarti seluruh data akan dikunjungi. Sebagai contoh, untuk show data, seluruh data yang ada di dalam Stack array akan dikunjungi dan ditampilkan. Begitu juga dengan search data. Semua data akan dikunjungi hingga data yang dicari ditemukan/tidak ditemukan.

Tampil Data.c

```
1 //Traversal Show Data
2 void show(Stack s){
3     int i;
4
5     for (i = s.top; i != -1; i--)
6     {
7         printf(...);
8     }
9 }
```

Cari Data.c

```
1 //Traversal Search Data
2 int search(Stack s, int yangDicari){
3     int i;
4
5     for (i = s.top; i != -1; i--)
6     {
7         if(s.B[i] == yangDicari){
8             return i; //returnkan index yang dicari
9         }
10    }
11
12    return -1; //tidak ditemukan
13 }
```

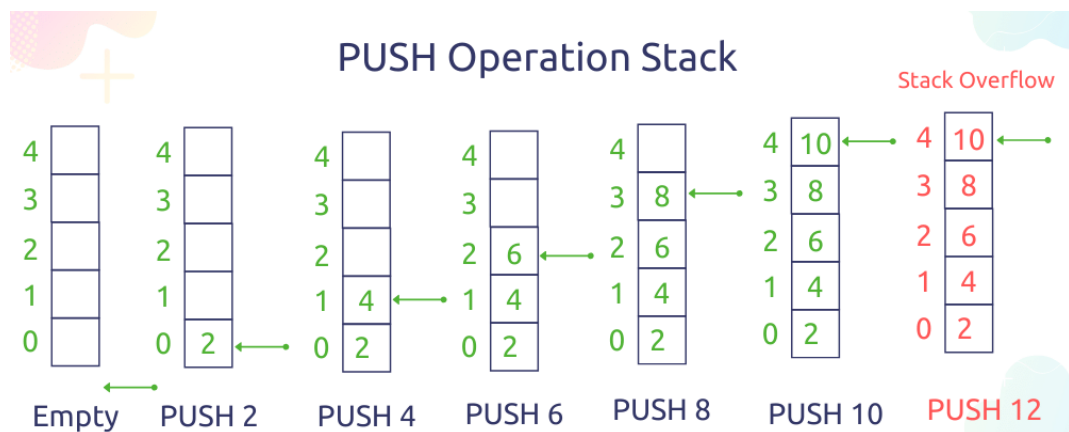
4. Operasi Stack

a) **PUSH** (Operasi Penambahan Data dalam Stack)

Operasi ini berfungsi untuk menambahkan data ke dalam stack dan akan ditempatkan pada bagian paling atas(top pada Stack) dari Stack yang bersangkutan. Untuk melakukan **PUSH**, top terlebih dahulu harus dilakukan *increment*. Setelah itu, data baru dapat dimasukkan ke dalam Stack. Ini dikarenakan pada awal inisialisasi top bernilai -1. Jika tidak di *increment* terlebih dahulu, data tidak akan masuk ke dalam Stack array karena index array dimulai dari 0.

```
PUSH.c

1 void push(Stack *s, infoType data){
2     (*s).top++;
3     (*s).data[(*s).top] = data; //top sebagai acuan index
4 }
```




Sumber gambar :

<https://holycoders.com/data-structures-stack/>

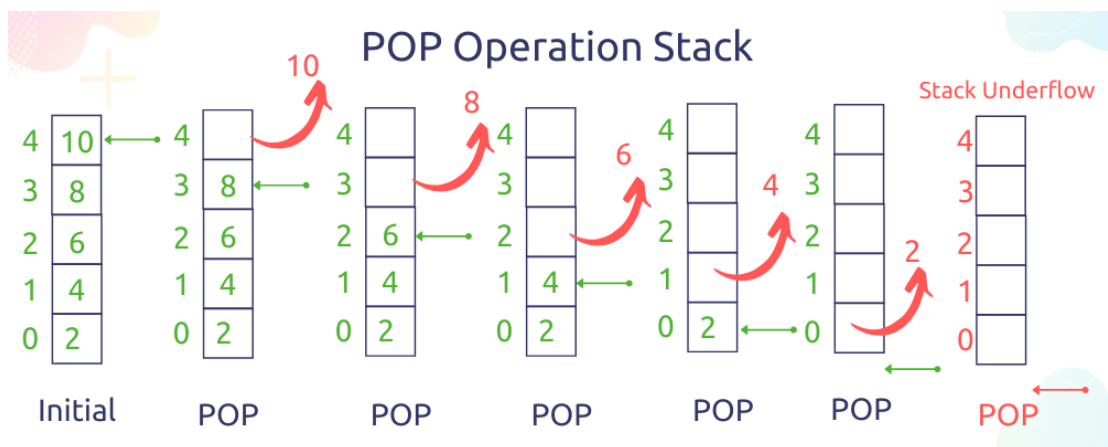
b) **POP** (Operasi Pengurangan Data dalam Stack)

Operasi ini berfungsi untuk mengambil/mengeluarkan data dari Stack. Data yang diambil adalah data paling atas (top pada Stack). Bila terdapat n data dan data ke- n di **POP**, maka data $n-1$ menjadi yang paling atas. Pada code dibawah terdapat variabel lokal **temp**. Variabel **temp** digunakan untuk menyimpan data sementara yang telah dikeluarkan dari Stack array agar dapat dibalikan nilainya. Bila memakai prosedur, maka tidak perlu menggunakan variabel lokal.



```
POP.c

1 //Fungsi POP
2 infoType pop(Stack *s){
3     infoType temp;
4     temp = (*s).data[(*s).top];
5     (*s).top--;
6     return temp;
7 }
```



Sumber gambar :

<https://holycoders.com/data-structures-stack/>

Hal yang perlu diingat !!!

- 1) Pada modul ini, akan digunakan array untuk mengimplementasikan Stack. Selalu ingat array bersifat statis dimana jumlah elemennya sudah ditentukan dari awal.
- 2) Pastikan Inisialisasi sudah dilakukan dan terpanggil pada setiap project.
- 3) Index elemen array selalu dimulai dari 0.
- 4) Bila Stack array penuh tidak dapat melakukan **PUSH**.
- 5) Bila Stack array kosong tidak dapat melakukan **POP**.
- 6) Top selalu menjadi acuan

GUIDED 3 – STACK ARRAY

Silahkan kerjakan guided di bawah ini sebelum mengikuti praktikum. Format pengumpulan guided adalah **GD3_Y_XXXXX.zip** dengan **XXXXX** adalah **5-digit terakhir NPM** dan **Y** adalah kelas. Good Luck ~~ 🍀

header.h

```
header.h

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define max_stack 10
5
6  typedef int infoType;
7
8  typedef struct{
9      int top;
10     infoType data[max_stack];
11 } Stack;
12
13 void init(Stack *s);
14 void push(Stack *s, infoType data);
15 infoType pop(Stack *s);
16 void show(Stack s);
17
18 int isEmpty(Stack s);
19 int isFull(Stack s);
20
```

source.c

```
source.c

1  #include "header.h"
2
3  void init(Stack *s){
4      (*s).top = -1;
5  }
6
7  int isFull(Stack s){
8      return s.top == max_stack - 1;
9  }
10
11 int isEmpty(Stack s){
12     return s.top == -1;
13 }
14
15 void push(Stack *s, infoType data){
16     (*s).top++;
17     (*s).data[(*s).top] = data;
18 }
19
20 infoType pop(Stack *s){
21     infoType temp;
22
23     temp = (*s).data[(*s).top];
24     (*s).top--;
25
26     return temp;
27 }
28
29 void show(Stack s){
30     int i;
31
32     for(i=0;i!=s.top+1;i++){
33         printf("| %d ", s.data[i]);
34     }
35
36     printf("|");
37 }
38
39
```

main.c

```
main.c

1  #include "header.h"
2
3  int main(int argc, char *argv[]) {
4      int pil;
5      infoType data;
6      Stack s;
7
8      init(&s);
9
10     do{
11         system("cls");
12         printf("\t---MODUL 3 STACK ARRAY---");
13         printf("\n[1]. PUSH");
14         printf("\n[2]. POP");
15         printf("\n[3]. SHOW");
16         printf("\n[0]. Exit");
17         printf("\n>>> "); scanf("%d", &pil);
18
19         switch(pil){
20             default:
21                 printf("Pilihan tidak tersedia [!]);
22                 break;
23
24             case 1:
25                 if(!isFull(s)){
26                     printf("\n Masukkan Data Angka : "); scanf("%d",&data);
27                     push(&s, data);
28                 }else{
29                     printf("\n Stack Penuh [!]);
30                 }
31                 break;
32
33             case 2:
34                 if(!isEmpty(s)){
35                     printf("\nData %d berhasil dikeluarkan ~", pop(&s));
36                 }else{
37                     printf("\n Stack Kososng [!]);
38                 }
39                 break;
40
41             case 3:
42                 if(!isEmpty(s)){
43                     show(s);
44                 }else{
45                     printf("\n Stack Kososng [!]);
46                 }
47                 break;
48
49             case 0:
50                 printf("Good Luck~~");
51                 break;
52         }
53         getch();
54     }while(pil!=0);
55
56     return 0;
57 }
```