

MODUL 3

STACK ARRAY

Everyone: it's a game for kids



Programmers:



A. Tujuan Praktikum

- Memahami konsep *stack array*
- Memahami operasi dasar *stack array*
- Mampu menerapkan penggunaan *stack array* dalam bahasa C

B. Pendahuluan

Stack jika diartikan ke Bahasa Indonesia adalah tumpukan. Dalam kehidupan sehari-hari, *Stack* sering kita jumpai di lingkungan sekitar kita seperti tumpukan buku, tumpukan kertas, tumpukan baju, dan lain sebagainya. Konteks *Stack* hanya memiliki satu jalan masuk saja.

Dengan begitu, segala sesuatu yang **masuk pertama akan keluar terakhir** dan segala sesuatu yang **masuk terakhir akan keluar terlebih dahulu**. Konsep tersebut secara sederhana bisa kita sebut sebagai **First In Last Out (FILO)** atau **Last In First Out**

(LIFO). Contoh dari penerapan konsep FILO / LIFO dalam kehidupan sehari-hari seperti mainan puzzle menara hanoi.



Gambar 1. Mainan Puzzle Menara Hanoi

Teman-teman semua tentunya pernah melihat atau memainkan benda tersebut waktu kecil. Mainan tersebut sebenarnya menerapkan konsep stack dengan FILO / LIFO. Dikarenakan mainan tersebut hanya memiliki satu jalan masuk saja, kita tidak bisa secara langsung mengambil bagian puzzle paling bawah tanpa mengambil terlebih dahulu bagian puzzle yang ada di atasnya. Oleh karena itulah bagian puzzle yang masuk pertama akan terakhir dikeluarkan dan bagian puzzle yang masuk terakhir akan dikeluarkan terlebih dahulu.

C. Penerapan Stack dalam Pemrograman

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

#define max_stack 5

typedef char string[100];

typedef struct{
    string nama;
    int umur;
}Data;

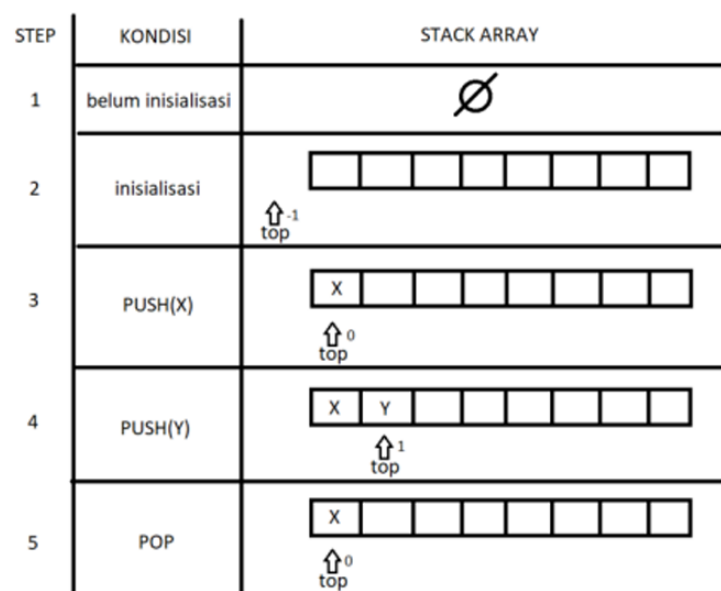
typedef struct{
    int top;
    Data d[max_stack];
} Stack;
```

Gambar 2. Pendefinisian Stack

Struktur dari suatu *Stack* terdiri dari top dan beberapa atribut yang dibutuhkan. Pada contoh code di atas, terdapat *Stack* yang terdiri dari atribut top berupa tipe data integer dan atribut lain yang merupakan record Data dengan array sejumlah max_stack. Struktur *Stack*

dapat berubah-ubah sesuai dengan kebutuhan. 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 dimasukkan berdasarkan posisi *top*. Maka dari itu *top* merupakan acuan utama dalam *Stack*. Untuk lebih jelas dapat dilihat ilustrasi alur *Stack* berikut.



Gambar 3. Ilustrasi Alur Stack

Indikasikan *top* adalah sebuah integer. Ketika inisialisasi telah dilakukan, struktur *Stack Array* akan terbentuk pada langkah 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 langkah 3 dimana **PUSH** telah terjadi. Data X akan dimasukkan ke dalam *Stack array* dengan acuan *top*. Sebelumnya *top* berada pada indeks -1, setelah terjadinya **PUSH**, *top* bertambah (+1) menjadi 0 dan data X dimasukkan ke dalam *Stack array* pada indeks 0 (indeks dengan acuan *top*). Begitu pula dengan langkah 4.

Pada langkah 5 ketika **POP** terjadi, *Stack* akan mengeluarkan data dari indeks yang ditunjuk oleh *top*. Sebelumnya pada langkah 4, *top* berada di indeks 1 menunjuk data Y.

Karena data inilah yang ditunjuk oleh *top*, maka data Y akan diambil/dihapus, sehingga *top* akan berkurang (-1) kembali menunjuk indeks 0 sebagai data paling atas/ujung yang baru.

D. Fungsi dan Prosedur dalam Stack

1) Inisialisasi



```
void init(Stack *s){
    (*s).top=-1;
}
```

Gambar 4. Code Prosedur Inisialisasi Stack

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*.

2) isEmpty



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

Gambar 5. Code Fungsi isEmpty Stack

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.

3) isFull

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in C and defines a function named `isFull` that takes a `Stack s` as an argument and returns a boolean value. The function checks if the `top` index of the stack is equal to `max_stack - 1`.

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

Gambar 6. Code Fungsi isFull Stack

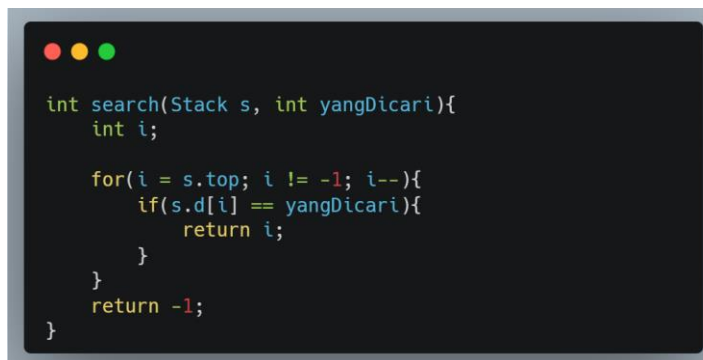
isFull memiliki kegunaan untuk mengecek apakah *Stack array* penuh sehingga tidak dapat dilakukan operasi **PUSH** lagi. Fungsi *isFull* juga memiliki nilai balikan *Boolean*. Secara logika, kita menganggap *Stack* penuh apabila *top* sama dengan jumlah `max_stack - 1`. Mengapa dikurangi dengan 1, ingat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

4) Traversal

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in C and defines a function named `show` that takes a `Stack s` as an argument. The function iterates from the `top` index down to -1, printing each element of the stack.

```
void show(Stack s){  
    int i;  
    for(i = s.top; i != -1; i--){  
        printf(".....");  
    }  
}
```

Gambar 7. Code Prosedur Show Data Stack

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in C and defines a function named `search` that takes a `Stack s` and an integer `yangDicari` as arguments. The function iterates from the `top` index down to -1, checking if the element at the current index matches `yangDicari`. If found, it returns the index; otherwise, it returns -1.

```
int search(Stack s, int yangDicari){  
    int i;  
  
    for(i = s.top; i != -1; i--){  
        if(s.d[i] == yangDicari){  
            return i;  
        }  
    }  
    return -1;  
}
```

Gambar 8. Code Fungsi Search Data Stack

Traversal dapat digunakan untuk tampil data maupun *search data*. Traversal berarti seluruh data akan dikunjungi. Sebagai contoh, untuk tampil 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.

5) TopData

```
Data topData(Stack s){  
    return s.d[s.top];  
}
```

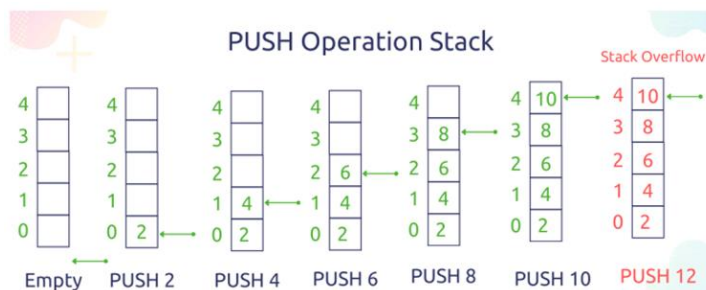
Gambar 9. Code Fungsi topData Stack

TopData dapat digunakan untuk mengembalikan/mendapatkan data atau nilai yang terletak pada *top* tanpa menghapus data tersebut dari *Stack*. Untuk penggunaan fungsi *topData* pastikan *stack* tidak dalam keadaan kosong.

6) Push (Penambahan Data Pada Stack)

```
void push(Stack *s, Data d){  
    (*s).top++;  
    (*s).d[(*s).top]=d;  
}
```

Gambar 10. Code Prosedur Push Data dalam Stack



Gambar 11. Visualisasi Operasi Push

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* pada indeks ke *top*. Ini dikarenakan pada awal inisialisasi *top* bernilai -1. Jika tidak di

increment terlebih dahulu, data tidak akan masuk ke dalam *Stack array* karena indeks *array* dimulai dari 0. Untuk lebih jelasnya, bisa dilihat pada gambar 11.

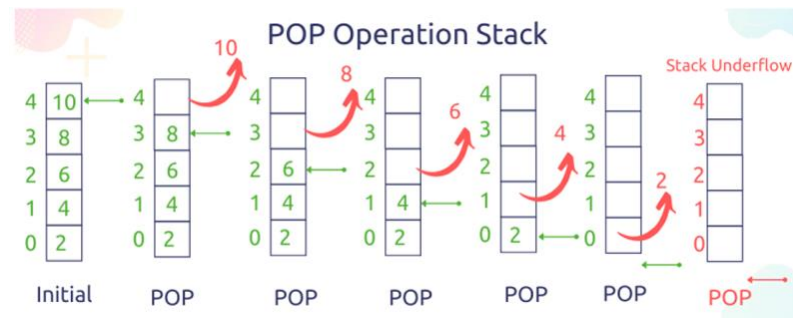
7) Pop (Pengurangan / Penghapusan Data Pada Stack)

```

Data pop(Stack *s){
    Data temp;

    temp = (*s).d[(*)s).top];
    (*s).top--;
    return temp;
}
```

Gambar 12. Code Fungsi Pop Data dalam Stack



Gambar 13. Visualisasi Operasi Pop

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 di atas 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. Untuk lebih jelasnya bisa dilihat pada gambar 13.

E. Contoh Penerapan Stack dalam Struktur Data :

- *Reverse String*
- *Conversion for Prefix to Infix, Prefix to Postfix, Postfix to Prefix, Postfix to Infix, Infix to Postfix*
- *Undo and Redo*

- *Sorting*
- *The Stock Span Problem*
- Mengecek Keseimbangan tanda kurung
- *Tower of Hanoi*, dll.

Pelajari lebih lanjut mengenai penerapan *Stack* pada struktur data untuk memecahkan masalah. Dapat dipelajari lebih lanjut pada: [geeksforgeeks.org/stack-data-structure/](https://www.geeksforgeeks.org/stack-data-structure/)

Catatan !!!

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

GUIDED

Pak Agoes adalah seorang programmer yang ahli. Pada kesempatan ini, ia diminta oleh klien untuk membuat sebuah program yang bisa menyimpan data nama dan umur. Klien meminta untuk jumlah maksimal data yang dapat disimpan adalah sebanyak 5 data. Pak Agoes lalu ingin mencoba membuat program tersebut dengan penyimpanan data berupa *Stack*.

Header

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

#define max_stack 5

typedef char string[100];

typedef struct{
    string nama;
    int umur;
}Data;

typedef struct{
    int top;
    Data d[max_stack];
}Stack;

int isEmpty(Stack s);
int isFull(Stack s);

void init(Stack *s);
void push(Stack *s, Data d);

Data pop(Stack *s);

void show(Stack s);
int search(Stack s, string nama);
Data topData(Stack s);
```

Source

```
#include "header.h"

void init(Stack *s){
    (*s).top=-1;
}

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

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

void push(Stack *s, Data d){
    (*s).top++;
    (*s).d[(*s).top]=d;
}

Data pop(Stack *s){
    Data temp;

    temp = (*s).d[(*s).top];
    (*s).top--;
    return temp;
}

void show(Stack s){
    int i;

    printf("\n\t");
    for(i = s.top; i != -1; i--){
        printf(" | %s - %d ", s.d[i].nama, s.d[i].umur);
    };
    printf("| ");
}

int search(Stack s, string nama){
    int i;

    for(i = s.top; i != -1; i--){
        if(strcmp(s.d[i].nama, nama)==0){
            return i;
        }
    }
    return -1;
}

Data topData(Stack s){
    return s.d[s.top];
}
```

Main

```
#include "header.h"

int main(int argc, char *argv[]) {
    int menu, index;
    string nama;
    Data d, temp;
    Stack s;
    init(&s);

    do{
        system("cls");
        printf("\n\t==== MODUL 3 STACK ARRAY ====");
        printf("\n\t[1] PUSH");
        printf("\n\t[2] POP");
        printf("\n\t[3] SHOW");
        printf("\n\t[4] TOP DATA");
        printf("\n\t[5] SEARCH DATA");
        printf("\n\t[0] EXIT");
        printf("\n\t>>> "); scanf("%d", &menu);

        switch(menu){
            case 1 :
                if(!isFull(s)){
                    printf("\n\tMasukkan Data Nama : "); fflush(stdin); gets(d.nama);
                    printf("\n\tMasukkan Data Umur : "); scanf("%d", &d.umur);
                    push(&s, d);
                    printf("\n\t\tData %s berhasil dimasukkan ~", d.nama);
                }else{
                    printf("\n\t\tStack Overflow [!]" );
                }
                break;

            case 2 :
                if(!isEmpty(s)){
                    temp = pop(&s);
                    printf("\n\t\tData %s berhasil dikeluarkan ~", temp.nama);
                }else{
                    printf("\n\t\tStack Underflow [!]" );
                }
                break;

            case 3 :
                if(!isEmpty(s)){
                    show(s);
                }else{
                    printf("\n\t\tStack Empty [!]" );
                }
                break;

            case 4 :
                if(!isEmpty(s)){
                    temp = topData(s);
                    printf("\n\t\tData paling atas dari stack adalah %s dengan umur %d ~", temp.nama, temp.umur);
                }else{
                    printf("\n\t\tStack Empty [!]" );
                }
                break;

            case 5 :
                if(!isEmpty(s)){
                    printf("\n\tMasukkan nama orang yang ingin dicari : "); fflush(stdin); gets(nama);
                    index = search(s, nama);
                    if(index != -1){
                        printf("\n\tData tersebut berada di urutan ke-%d dalam stack", index+1);
                    }else{
                        printf("\n\t\tError 404 Data Not Found...");
                    }
                }else{
                    printf("\n\t\tStack Empty [!]" );
                }
                break;

            case 0 :
                printf("\n\t\tGood Luck Have Fun :D");
                printf("\n\t\tNama Lengkap - NPM - Kelas");
                break;

            default :
                printf("\n\t\tTidak Tersedia [!]" );
                break;
        }
        getch();
    }while(menu!=0);

    return 0;
}
```

Ketentuan Guided :

- 1) Pastikan membaca modul terlebih dahulu sebelum mengerjakan guided.
- 2) Kerjakan guided secara **MANDIRI**.
- 3) **Pahami alur masing-masing code** dalam prosedur dan fungsi stack, **JANGAN HANYA COPAS** karena akan sangat membantu kalian saat pengerjaan UGD nantinya.
- 4) Terlambat mengumpulkan **tidak akan diberikan toleransi**.
- 5) Jangan lupa untuk **menerapkan konsep ADT** selama mengerjakan Guided ini.
- 6) Pastikan **ekstensi file .c bukan .cpp**.
- 7) Format penamaan file **GD3_Y_XXXXX.zip** dengan ketentuan :
 - a. Y = Kelas
 - b. X = 5 Digit NPM