# JOBSHEET 06

## Stack

| Fakultas | Proram Studi | Kode MK | Waktu |
|---|---|---|---|
| Teknik | Teknik Informatika | INF1.62.2014 | 2 x 50 Menit |

## TUJUAN PRAKTIKUM

1. Mahasiswa memahami konsep tumpukan atau stack dalam pemrograman C.
2. Mahasiswa memahami operasi yang ada dalam struktur data tumpukan atau stack.
3. Mahasiswa mampu mengimplementasikan struktur data tumpukan atau stack menggunakan pemrograman C dengan IDE.

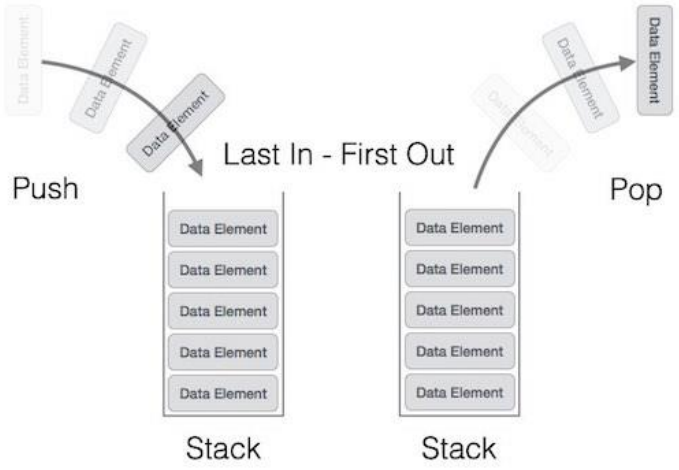
## HARDWARE & SOFTWARE

1. Personal Computer
2. DevC++ IDE

## TEORI SINGKAT

### A. Stack

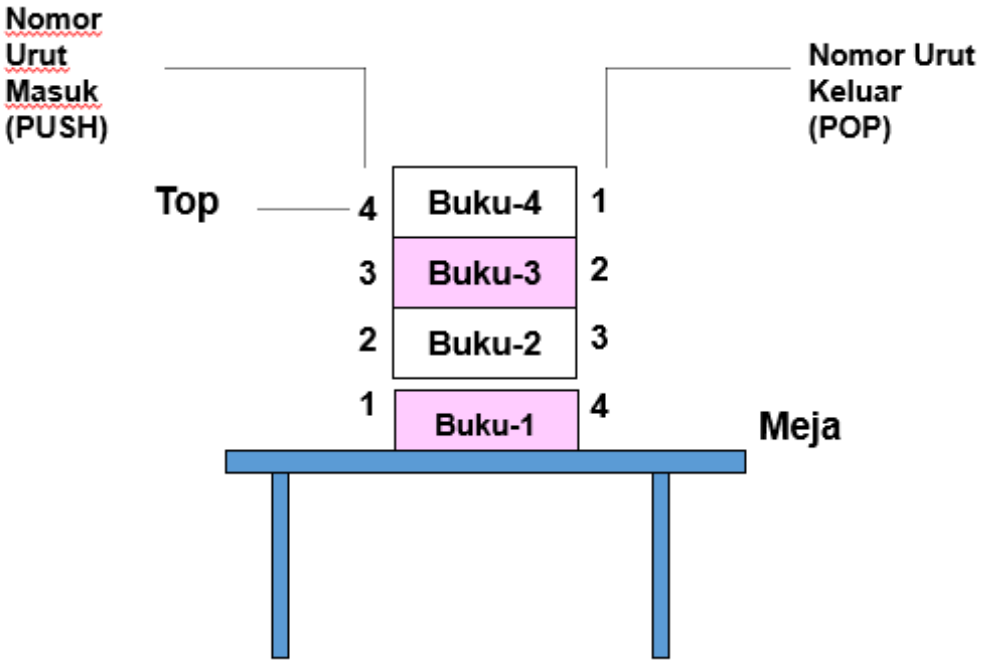
Stack adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO. Dengan demikian, elemen terakhir yang disimpan dalam stack menjadi elemen pertama yang diambil. Dalam proses komputasi, untuk meletakkan sebuah elemen pada bagian atas dari stack, disebut push. Dan untuk memindahkan dari tempat yang atas tersebut, disebut pop.



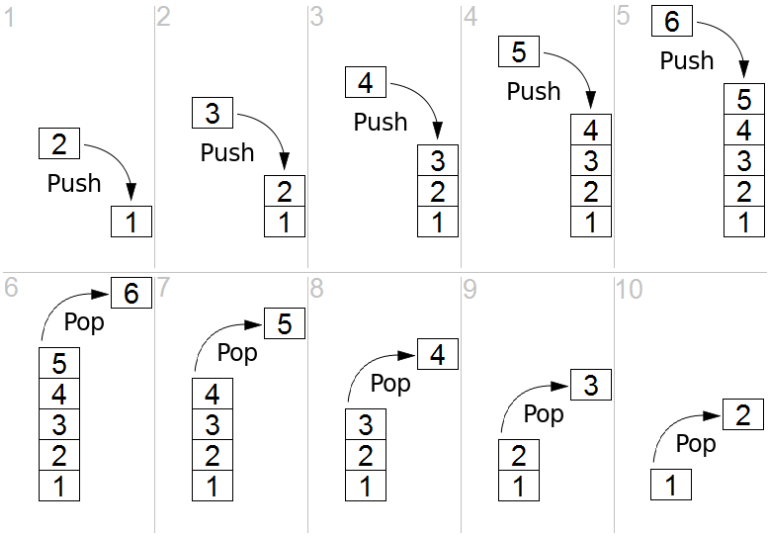

LIFO : "terakhir masuk sebagai yang pertama keluar" (Last In First Out)

Contoh:

kita mempunyai dua buah kotak yang kita tumpuk, sehingga kotak kita letakkan di atas kotak yang lain. Jika kemudian stack dua buah kotak tersebut kita tambah dengan kotak ketiga dan seterusnya, maka akan kita peroleh sebuah stack kotak, yang terdiri dari N kotak.




Gambar 1. Contoh Ilustrasi Stack




Gambar 2. Contoh Ilustrasi Push dan Pop Stack (LIFO)

## B. Stack dengan Array

Ada beberapa cara untuk menyajikan sebuah stack tergantung pada permasalahan yang akan diselesaikan. Bentuk penyajian stack bisa menggunakan tipe data array, tetapi sebenarnya penyajian stack menggunakan array adalah kurang tepat karena banyaknya elemen dalam array adalah statis, sedangkan dalam stack banyaknya elemen sangat bervariasi atau dinamis. Meskipun demikian, array bisa digunakan untuk penyajian stack, tetapi dengan anggapan bahwa banyaknya elemen maksimal dari suatu stack tidak melebihi batas maksimum banyaknya elemen array. Pada suatu saat, ukuran stack akan sama dengan ukuran array. Bila diteruskan menambah data, maka akan terjadi overflow. Oleh karena itu, perlu ditambahkan data untuk mencatat posisi ujung stack. Ada dua macam penyajian stack menggunakan array, yaitu Single stack dan Double stack.

Deklarasi Stack:

```
#define MAX 5
typedef int Itemtype
typedef struct {
        Itemtype item[MAX];
        int count;
} Stack;
```

Pada saat ukuran stack sudah terpenuhi sebanyak MAX, kalau diteruskan menambah data melebihi batas maksimum maka akan terjadi overflow. Dengan demikian perlu data tambahan untuk mencatat posisi ujung stack.

C. Operasi pada Stack

1) Push digunakan untuk menambahkan elemen atau data baru dalam tumpukan. Elemen baru tersebut pasti akan menjadi elemen yang paling atas dalam tumpukan setiap kali ditambahkan. Sebelum menambahkan elemen baru kita harus memastikan tumpukan belum penuh.

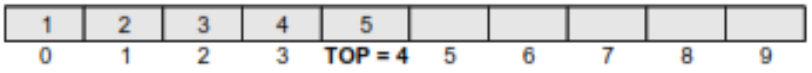



**Figure 7.5** Stack

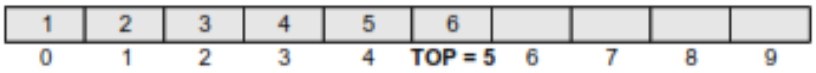



**Figure 7.6** Stack after insertion

2) Pop digunakan untuk menghapus elemen yang berada pada posisi paling atas dari stack.

3) Peek digunakan untuk mengecek elemen atau data paling atas tanpa menghapusnya dari stack.

4) isFull digunakan untuk memeriksa apakah kondisi stack sudah penuh.

   Dengan cara:

   a. Menambah satu (increment) nilai TOP of STACK setiap ada penambahan elemen stack selama stack masih belum penuh.

   b. Isikan nilai baru ke stack berdasarkan indeks TOP of STACK setelah ditambah satu (increment).

5) isEmpty digunakan untuk memeriksa apakah stack masih dalam kondisi kosong.

   Dengan cara:

   memeriksa TOP of STACK. Jika TOP masih = -1 maka berarti stack masih kosong.

6) Clear digunakan untuk mengosongkan stack.

## PERCOBAAN

1. **Implementasi Stack menggunakan Array Program 1**

```
// C program untuk implementasi stack menggunakan array
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// struktur untuk merepresentasikan stack
struct Stack {
    int atas;
    unsigned kapasitas;
    int* array;
};
```

```
// fungsi untuk membuat stack berdasarkan kapasitas yang
diberikan. Hal ini menginisialisasi ukuran
// stack as 0
struct Stack* buatStack(unsigned kapasitas)
{
   struct Stack* stack = (struct Stack*)malloc(sizeof(struct
Stack));
   stack->kapasitas = kapasitas;
   stack->atas = -1;
   stack->array = (int*)malloc(stack->kapasitas * sizeof(int));
   return stack;
}

// Stack penuh ketika top (atas) sama dengan indeks terakhir
int isFull(struct Stack* stack)
{
   return stack->atas == stack->kapasitas - 1;
}

// Stack kosong ketika top (atas) sama dengan -1
int isEmpty(struct Stack* stack)
{
   return stack->atas == -1;
}

// Fungsi push (dorong) untuk menambahkan item ke stack. Hal
ini menaikkan top (atas) sebesar 1
void push(struct Stack* stack, int item)
{
   if (isFull(stack))
         return;
   stack->array[++stack->atas] = item;
   printf("%d di push atau ditambahkan ke stack\n", item);
}

// Fungsi pop (melepaskan/hapus) sebuah item dari stack. Hal
ini menurunkan top (atas) sebesar 1
int pop(struct Stack* stack)
{
   if (isEmpty(stack))
         return INT_MIN;
   return stack->array[stack->atas--];
}

// Fungsi untuk mengembalikan bagian atas (top) dari stack
tanpa melepasnya
int peek(struct Stack* stack)
{
   if (isEmpty(stack))
         return INT_MIN;
   return stack->array[stack->atas];
}

// Driver program untuk mengetes fungsi-fungsi diatas
int main()
{
   struct Stack* stack = buatStack(100);

   push(stack, 10);
```

```
    push(stack, 20);
    push(stack, 30);

    printf("%d dilepaskan/dihapus dari stack\n", pop(stack));

    return 0;
}
```

2. **Implementasi Stack menggunakan Array Program 2**

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

#define UKURAN 5

void push(int);
void pop();
void display();

int stack[UKURAN], atas = -1;

int main() {
   int value, pilihan;
   while(1){
      printf("\n\n***** MENU *****\n");
      printf("1. Push\n2. Pop\n3. Tampilkan\n4. Keluar");
      printf("\nMasukan pilihanmu: ");
      scanf("%d",&pilihan);
      switch(pilihan){
          case 1:
               printf("Masukan value yang ingin dimasukan: ");
               scanf("%d",&value);
               push(value);
               break;
          case 2:
               pop();
               break;
          case 3:
               display();
               break;
          case 4:
               exit(0);
          default:
               printf("\nPilihan tidak valid! silakan coba
lagi!");
         }
               }
}
void push(int value){
   if(atas == UKURAN-1)
      printf("\nStack telah penuh, Tidak memungkinkan menambah
item");
   else{
      atas++;
      stack[atas] = value;
      printf("\nItem berhasil ditambahkan");
   }
}
```

```
   void pop(){
      if(atas == -1)
         printf("\nStack kosong, tidak memungkinkan untuk
   menghapus item");
      else{
         printf("\nItem berhasil dihapus: %d", stack[atas]);
         atas--;
      }
   }
   void display(){
      if(atas == -1)
         printf("\nStack kosong. Tidak ada item yang bisa
   ditampilkan");
      else{
         int i;
         printf("\nItem yang ada pada stack adalah:\n");
         for(i=atas; i>=0; i--)
       printf("%d\n",stack[i]);
      }
   }
```

3. **Implementasi Stack menggunakan Linked List Program1**

```
// C program untuk implementasi stack menggunakan array
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// struktur untuk merepresentasikan stack
struct StackNode {
      int data;
      struct StackNode* next;
};

struct StackNode* newNode(int data)
{
      struct StackNode* stackNode = (struct StackNode*)
malloc(sizeof(struct StackNode));
      stackNode->data = data;
      stackNode->next = NULL;
      return stackNode;
}

int isEmpty(struct StackNode* root)
{
      return !root;
}

void push(struct StackNode** root, int data)
{
      struct StackNode* stackNode = newNode(data);
      stackNode->next = *root;
      *root = stackNode;
      printf("%d di push atau ditambahkan ke stack\n", data);
}

int pop(struct StackNode** root)
{
```

```
	if (isEmpty(*root))
		return INT_MIN;
	struct StackNode* temp = *root;
	*root = (*root)->next;
	int popped = temp->data;
	free(temp);
	return popped;
}

int peek(struct StackNode* root)
{
	if (isEmpty(root))
		return INT_MIN;
	return root->data;
}

int main()
{
	struct StackNode* root = NULL;
	push(&root, 10);
	push(&root, 20);
	push(&root, 30);
	printf("%d dilepaskan/dihapus dari stack\n", pop(&root));
	printf("Elemen atas (Top) adalah %d\n", peek(root));
	return 0;
}
```

4. **Implementasi Stack menggunakan Linked List Program 2**

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

struct Node
{
   int data;
   struct Node *next;
}*atas = NULL;

void push(int);
void pop();
void tampilkan();

int main()
{
   int pilihan, value;
   printf("\n== Stack Menggunakan Linked List ==\n");
   while(1){
      printf("\n****** MENU ******\n");
      printf("1. Push\n2. Pop\n3. Tampilkan\n4. Keluar\n");
      printf("Masukan pilihanmu: ");
      scanf("%d",&pilihan);
      switch(pilihan){
            case 1: printf("Masukan nilai yang ingin ditambahkan
ke stack: ");
                  scanf("%d", &value);
                  push(value);
                  break;
            case 2:
```

```
                  pop();
                  break;
            case 3:
                  tampilkan();
                  break;
            case 4: exit(0);
            default: printf("\nPilihan tidak valid, silakan coba
lagi!\n");
      }
   }
}
void push(int value)
{
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   if(atas == NULL)
      newNode->next = NULL;
   else
      newNode->next = atas;
   atas = newNode;
   printf("\nItem berhasil ditambahkan\n");
}
void pop()
{
   if(atas == NULL)
      printf("\nStack kosong!!!\n");
   else{
      struct Node *temp = atas;
      printf("\nItem berhasil dihapus: %d", temp->data);
      atas = temp->next;
      free(temp);
   }
}
void tampilkan()
{
   if(atas == NULL)
      printf("\nStack kosong!!!\n");
   else{
      struct Node *temp = atas;
      while(temp->next != NULL){
       printf("%d--->",temp->data);
       temp = temp -> next;
      }
      printf("%d--->NULL",temp->data);
   }
}
```

5. **Implementasi Double Stack**

```
#include<stdio.h>
#define MAX 5

//Deklarasi Double Stack
typedef struct
{
int atas1;
int atas2;
```

```
int ele[MAX];
}DoubleStack;

//Menginisialisasi Double Stack
void init( DoubleStack *s )
{
s->atas1 = -1;
s->atas2 = MAX;
}

//Fungsi Push pada Stack1
void pushA( DoubleStack *s, int item )
{
if( s->atas2 == s->atas1 + 1 )
{
printf("\nStack Overflow Stack1");
return;
}

s->atas1++;
s->ele[s->atas1] = item;

printf("\nItem ditambah pada Stack1 : %d",item);
}

//Fungsi Push pada Stack2
void pushB( DoubleStack *s, int item )
{
if( s->atas2 == s->atas1 + 1 )
{
printf("\nStack Overflow Stack2");
return;
}

s->atas2--;
s->ele[s->atas2] = item;

printf("\nItem ditambah pada Stack2 : %d",item);
}

//Fungsi Pop pada Stack1
int popA( DoubleStack *s, int *item )
{
if( s->atas1 == -1 )
{
printf("\nStack Underflow Stack1");
return -1;
}

*item = s->ele[s->atas1--];
return 0;
}

//Fungsi Pop pada Stack2
int popB( DoubleStack *s, int *item )
{
if( s->atas2 == MAX )
{
printf("\nStack Underflow Stack2");
```

```
return -1;
}
*item = s->ele[s->atas2++];
return 0;
}

int main()
{
int item = 0;
DoubleStack s;
init(&s);
pushB( &s, 10);
pushA( &s, 20);
pushA( &s, 30);
pushB( &s, 40);
pushB( &s, 50);
pushB( &s, 60);


if( popA(&s, &item) == 0 )
printf("\nItem dihapus dari Stack1 : %d",item);
if( popA(&s, &item) == 0 )
printf("\nItem dihapus dari Stack1  : %d",item);
if( popA(&s, &item) == 0 )
printf("\nItem dihapus dari Stack1  : %d",item);
if( popB(&s, &item) == 0 )
printf("\nItem dihapus dari Stack2 : %d",item);
if( popB(&s, &item) == 0 )
printf("\nItem dihapus dari Stack2 : %d",item);
if( popB(&s, &item) == 0 )
printf("\nItem dihapus dari Stack2 : %d",item);

printf("\n");
return 0;
}
```

6. **Reversing Stack Menggunakan Recursion**

```
// C Program  untuk membalikkan stack menggunakan rekursi
#include <stdio.h>
#include <stdlib.h>
#define bool int

// Struktur node dari Stack
struct sNode {
      char data;
      struct sNode* next;
};

// Fungsi Prototype
void push(struct sNode** top_ref, int new_data);
int pop(struct sNode** top_ref);
bool isEmpty(struct sNode* top);
void print(struct sNode* top);

// Di bawah ini adalah fungsi rekursif
// yang menyisipkan elemen
// di bagian bawah stack.
```

```
void insertAtBottom(struct sNode** top_ref, int item)
{
	if (isEmpty(*top_ref))
		push(top_ref, item);
	else {

		// Menahan semua item di Fungsi Call Stack hingga
mencapai ujung Stack.
		// Ketika stack menjadi kosong, isEmpty(*top_ref)
menjadi True
		// bagian if di atas dijalankan dan item disisipkan di
bagian bawah
		int temp = pop(top_ref);
		insertAtBottom(top_ref, item);

		// Setelah item ditambahkan di bagian bawah
		// dorong semua item yang disimpan di Fungsi Call
Stack
		push(top_ref, temp);
	}
}

// Di bawah ini adalah fungsi yang membalikkan stack yang
diberikan
// menggunakan insertAtBottom()
void reverse(struct sNode** top_ref)
{
	if (!isEmpty(*top_ref)) {
		// Menahan semua item di Fungsi Call Stack
		// hingga akhir stack dicapai
		int temp = pop(top_ref);
		reverse(top_ref);

		// Memasukan semua item (ditahan di Function Call
Stack) satu per satu
		// dari bawah ke atas. Setiap item dimasukkan di
bagian bawah
		insertAtBottom(top_ref, temp);
	}
}

// Driver Code
int main()
{
	struct sNode* s = NULL;
	push(&s, 4);
	push(&s, 3);
	push(&s, 2);
	push(&s, 1);

	printf("\n Stack Awal ");
	print(s);
	reverse(&s);
	printf("\n Stack yang dibalikkan ");
	print(s);
	return 0;
}

// Fungsi untuk mengecek apakah Stack kosong
```

```
bool isEmpty(struct sNode* top)
{
	return (top == NULL) ? 1 : 0;
}

// Fungsi untuk menambahkan item ke stack
void push(struct sNode** top_ref, int new_data)
{

	// mengalokasikan node
	struct sNode* new_node
		= (struct sNode*)malloc(sizeof(struct sNode));

	if (new_node == NULL) {
		printf("Stack overflow \n");
		exit(0);
	}

	// Memasukan kedalam data
	new_node->data = new_data;

	// Menautkan list lama dari node baru
	new_node->next = (*top_ref);

	// Menggerakan head  untuk menunjuk (point) ke node baru
	(*top_ref) = new_node;
}

// Fungsi untuk mengeluarkan item dari stack
int pop(struct sNode** top_ref)
{
	char res;
	struct sNode* top;

	// jika Stack kosong, maka error
	if (*top_ref == NULL) {
		printf("Stack overflow \n");
		exit(0);
	}
	else {
		top = *top_ref;
		res = top->data;
		*top_ref = top->next;
		free(top);
		return res;
	}
}

// Fungsi untuk menampilkan linked list
void print(struct sNode* top)
{
	printf("\n");
	while (top != NULL) {
		printf(" %d ", top->data);
		top = top->next;
	}
}
```

## LATIHAN

1. Latihan 1

```
------
int MAXSIZE = 8;
int stack[8];
int top = -1;
int isempty() {

   if(top == -1)
      return 1;
   else
      return 0;
}

int isfull() {
   if(top == MAXSIZE - 1)
      return 1;
   else
      return 0;
}


int peek() {
   return stack[----];
}

int pop() {
   int data;

   if(!isempty()) {
      data = stack[----];
      top = top - 1;
      return ----;
   } else {
      printf("Tidak dapat mengambil data, Stack kosong.\n");
   }
}
int push(int data) {
   if(!isfull()) {
      top = top + 1;
      stack[----] = ----;
   } else {
      printf("Tidak dapat menambahkan data. Stack penuh\n");
   }
}

int -----() {
   // Menambahkan item ke stack
   push(3);
   push(5);
   push(9);
   push(1);
   push(12);
   push(15);

   printf("Elemen teratas pada Stack adalah: %d\n" ,----());
```

```
    printf("Semua elemennya adalah: \n");

    // print stack data
  while(!isempty()) {
    int data = pop();
    printf("%d\n",data);
  }

  printf("Stack full: %s\n" , isfull()?"true":"false");
  printf("Stack empty: %s\n" , isempty()?------------

    return 0;
}
```

2. Latihan 2

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

#define MAX 5   //Jumlah item maksimum yang dapat disimpan
int top=-1, stack[----];
void push();
void ----;
void ----;

int main()
{
    int ch;

    while(1)    //infinite loop, akan berakhir ketika pilihan
adalah 4
    {

        printf("\n*** Menu Stack  ***");
        printf("\n\n1.Push\n2.Pop\n3.Tampilkan\n4.Keluar");
        printf("\n\nMasukan pilihanmu (1-4):");
        scanf("%d",-----);

        switch(ch)
        {
            case 1: push();
                    break;
            case 2: pop();
                    break;
            case 3: display();
                    break;
            case 4: exit(0);

            -----: printf("\nPilihan tidak valid!");
        }
    }
}

void ----()
{
    int val;

    if(------==MAX-1)
```

```
        {
            printf("\nStack penuh!!");
        }
        else
        {
            printf("\nMasukan item yang ingin ditambahkan:");
            scanf("%d",-----);
            top=top+1;
            stack[top]=val;
        }
    }

    void pop()
    {
        if(top==-1)
        {
            printf("\nStack kosong!!");
        }
        else
        {
            printf("\nItem yang dihapus adalah %d",stack[top]);
            top=top-1;
        }
    }
```

```
void display()
{
    int i;

    if(top==-1)
    {
        printf("\nStack kosong!!");
    }
    else
    {
        printf("\nItem pada Stack adalah...\n");
        for(i=top;i>=0;--i)
            printf("%d\n",stack[----]);
    }
}
```

## TUGAS

1. Buatlah sebuah program dengan menggunakan stack yang berfungsi untuk menyimpan data nilai mahasiswa yang terdiri dari NIM, nama, dan nilai mahasiswa. Tambahkan beberapa fasilitas seperti sorting (pengurutan data), push (menambahkan data), pop (mengambil data), display (untuk menampilkan data stack). Tampilan dan menu sesuaikan dengan kreativitas masing-masing.
2. Buatlah program sederhana pembalik kata dengan menggunakan stack. Program terdiri dari 3 menu,
    1) Input Kata
    2) Balik Kata
    3) Exit

## DAFTAR PUSTAKA

1. Kernighan, Brian W, & Ritchie, Dennis M. 1988. The Ansi C Programming Language Second Edition, Prentice-Hall.
2. Cipta Ramadhani. 2015. Dasar Algoritma & Struktur Data. Yogyakarta: ANDI.