MAKALAH SORTING DALAM PEMROGRAMAN

BAHASA C



Oleh :

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

BAB I

PENDAHULUAN

1.1 Latar Belakang

Salah satu konsep yang sangat berguna dalam ilmu komputer adalah satu bentuk struktur data yang disebut dengan stack (tumpukan). Dalam ilmu komputer sering sekali istilah stack (tumpukan) disebutkan. Dengan ini kami coba mengenali apa itu stack dan bagaimana uraian-uraiannya menurut 5w+1h.

Struktur data adalah karakteristik yang terkait dengan sifat dan cara  penyimpanan sekaligus penggunaan atau pengaksesan data. Karakteristik tersebut mengikat dan sekaligus secara timbal balik dipengaruhi oleh algoritma yang mengakses data tersebut, sehingga disebutkan Algoritma dan Struktur Data merupakan satu kesatuan. Salah satu teknik dasar tentang struktur data adalah stack (tumpukan) yang lebih lanjut akan dibahas dalam makalah ini.

1.2 Rumusan Masalah

* Apa itu Stack?
* Kenapa bisa tersusun menjadi stack?
* Siapa pengguna Stack?
* Mengapa aplikasi Stack digunakan?
* Dimana mendeklarasikan Stack?
* Kapan operasi pada Stack digunakan?
* Bagaimana kelebihan dan kekurangan Stack?
* Bagaimana contoh program stack?

1.3 Tujuan Penulisan

Sejalan dengan rumusan masalah di atas, makalah ini disusun dengan tujuan untuk mengetahui:

* Untuk mengetahui definisi stack.
* Untuk mengetahui apa yang ada dalam stack.
* Untuk mengetahui siapa pengguna stack.
* Untuk mengetahui tujuan penggunaan aplikasi stack.
* Untuk mengetahui mendeklarasikan stack.
* Untuk mengetahui penggunaan operasi stack.
* Untuk mengetahui kelebihan dan kekurangan stack.
* Untuk mengetahui contoh program stack.

1.3 Manfaat

Manfaat yang diperoleh dari makalah tersebut adalah sebagai berikut:

1.Pembaca

* Menjadikan pembaca makalah semata-mata agar pembaca memahami dasar-dasar pemrograman.
* Untuk memotivasi pembaca untuk membuat makalah yang lebih baik dari Ini.

2.Pengarang

* Untuk memberikan semangat pada mata kuliah yang sedang diambil saat ini.
* Sebagai titik awal untuk melakukan pembelajaran agar pembelajaran selanjutnya dapat dilakukan dengan lebih baik lagi.

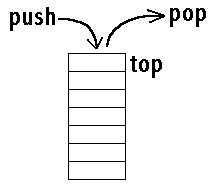
BAB II

PEMBAHASAN

2.1 Pengertian Stack

Stack merupakan jenis linked list yang menerapkan konsep LIFO (Last In Fist Out) artinya elemen dari struktur (node) yang dimasukkan pertama kali ke dalam rangkaian akan dikeluarkan (diproses) terakhir kali sedangkan elemen yang masuk terakhir akan diproses / dikeluarkan pertama.

 Dengan kata lain setiap transaksi atau aktifitas yang ada dalam stack dilakukan pada kepalanya (head) saja. Pada stack hanya ada 2 fungsi utama yaitu Push atau menambah/memasukkan node (struktur) ke dalam stack, dan Pop atau mengeluarkan/menghapus node dari stack.



cara kerja stack dapat di ilustrasikan sebagai berikut:

| D | ---> atas

| C |

| B |

| A | ---> bawah

Anggaplah gambar diatas sebagai tumpukan data, D sebagai node(struktur) yang atas (terkhir dimasukkan) dan A sebagai node yang bawah (pertamakali dimasukkan). Jika kita melakukan operasi Push (menambah), misalnya E, maka penambahan akan dilakukan pada nilai yang paling atas, yaitu D, sehingga menjadi:

| E | ---> atas

| D |

| C |

| B |

| A | ---> bawah

Sehingga nilai E menjadi nilai atas yang baru, dan jika dilakukan operasi pop, maka stack akan menjadi seperti gambar yang pertama diatas, yaitu nilai D menjadi nilai atas, dan nilai E akan dihapus dari tumpukan (stack).

Ada dua macam penyajian stack menggunakan array, yaitu Single stack dan Double stack.

2.2 single stack

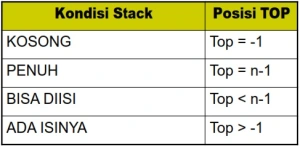
Single stack dapat dipresentasikan menggunakan array satu dimensi. Prinsip yang digunakan single stack adalah LIFO (Last In First Out). Sedangkan proses pada single stack ada :

Awal (Inisialisasi)

Push (Simpan, masuk, insert, tulis)

Pop (Ambil, keluar, delete, baca)

Kondisi single Stack ditentukan oleh posisi atau isi TOP



Deklarasi UmumBentuk deklarasi single stack:

#define MAX 50

#define true 1

#define false 0

Char stack[MAX];

Int top;

Operasi-Operasi Single Stack

* Fungsi Init
* Fungsi Full
* Fungsi empty
* Fungsi clear
* Fungsi Push
* Fungsi Po

/\*operasi-operasi single stack\*/

#include<stdio.h>

#include<conio.h>

#define MAX 50

#define true 1

#define false 0

char stack[MAX];

int top;

void init(void);

int full (void);

int empty (void);

char pop (void);

void clear (void);

void push (char info);

void baca (void);

main()

{

char pilih,elm;

printf("----Demo Operasi Single Stack----\n");

init();

do

{

printf("operasi single stack:\n");

printf("[1] PUSH\n");

printf("[2] POP\n");

printf("[3] CLEAR\n");

printf("[4] BACA\n");

printf("[5] SELESAI...\n");

printf(" Pilihan : ");scanf("%s",&pilih);

switch(pilih)

{

case '1': printf("PUSH ");scanf("%s",&elm);push(elm);break;

case '2': elm=pop();printf("POP %c\n",elm);break;

case '3': clear();break;

case '4': baca();break;

case '5': break;

default : printf("Salah pilih...\n");

}

printf("\n");

}while(pilih!='5');

getche();

}

void init(void)

{

top=0;

}

void push(char info)

{

if(full()!=true)

{ top++;

stack[top]=info; }

else printf("Stack overflow...\n");

}

char pop(void)

{

char info;

if(empty()!=true)

{ info=stack[top];

top--;

return(info);

}

else printf("Stack underflow...\n");

}

void clear(void)

{

top=0;

}

int full(void)

{

if(top==MAX) return(true);

else return(false);

}

int empty(void)

{

if(top==0) return(true);

else return(false);

}

void baca(void)

{

int i;

printf("isi stack : ");

if(top>0)

{

for(i=1;i<=top;i++)

printf("%c ",stack[i]);

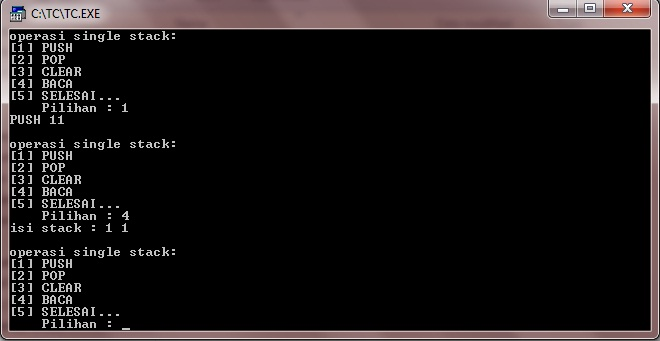
}

else printf("(kosong)");

printf("\n");

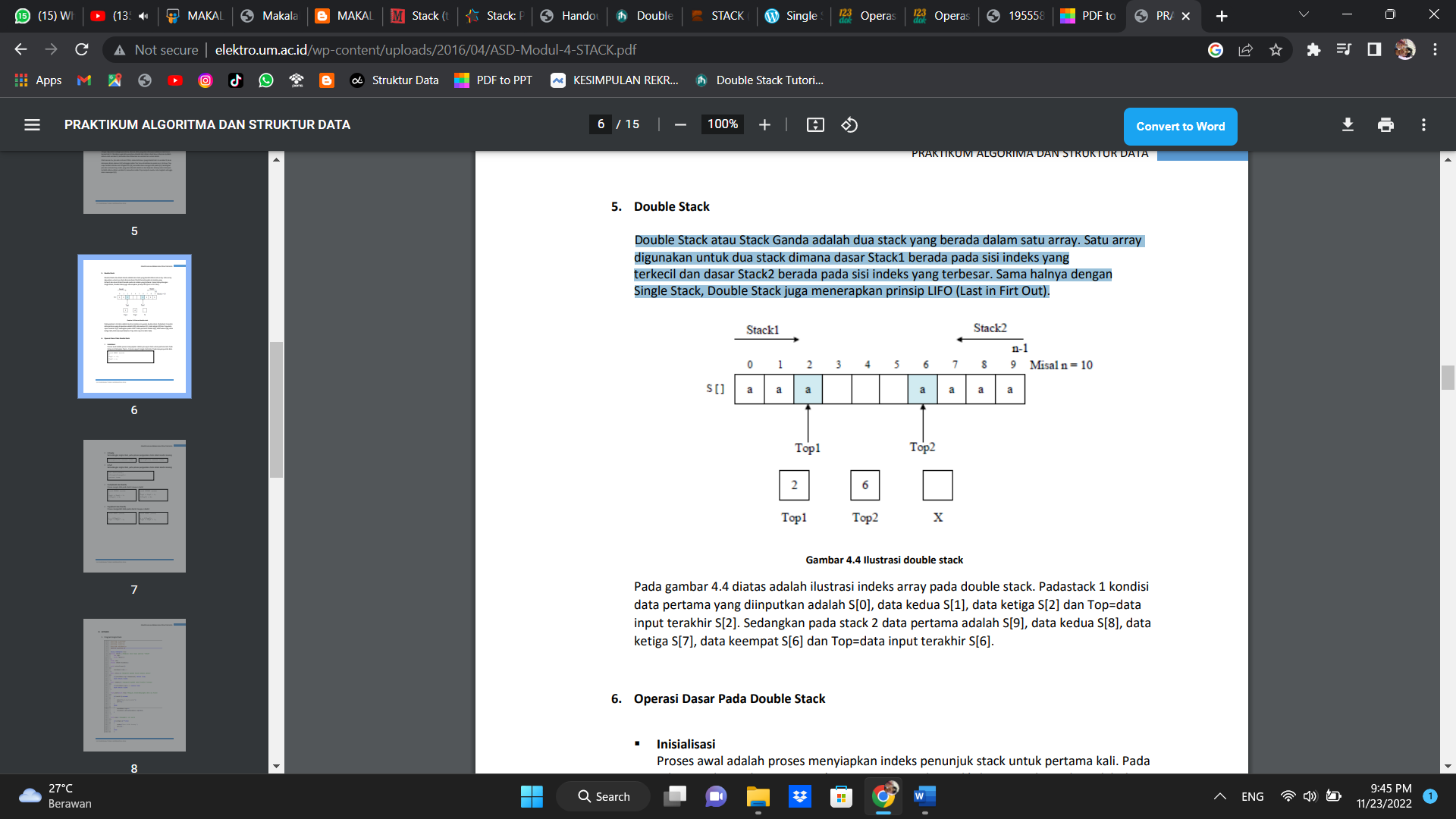
}

Output :



2.3 Double Stack

Double Stack atau Stack Ganda adalah dua stack yang berada dalam satu array. Satu array digunakan untuk dua stack dimana dasar Stack1 berada pada sisi indeks yang terkecil dan dasar Stack2 berada pada sisi indeks yang terbesar. Sama halnya dengan Single Stack, Double Stack juga menerapkan prinsip LIFO (Last in Firt Out).



diatas adalah ilustrasi indeks array pada double stack. Padastack 1 kondisi data pertama yang diinputkan adalah S[0], data kedua S[1], data ketiga S[2] dan Top=data input terakhir S[2]. Sedangkan pada stack 2 data pertama adalah S[9], data kedua S[8], data ketiga S[7], data keempat S[6] dan Top=data input terakhir S[6].

Contoh Program Double Stack:

#include<stdio.h>

#define MAX 5

//Declaration of Double Stack

**typedef** **struct**

{

**int** top1;

**int** top2;

**int** ele[MAX];

}DStack;

//Initialization of Double Stack

**void** init( DStack \*s )

{

    s->top1 = -1;

    s->top2 = MAX;

}

//Push Operation on Stack1

**void** pushA( DStack \*s, **int** item )

{

**if**( s->top2 == s->top1 + 1 )

    {

**printf**("\nStack Overflow Stack1");

**return**;

    }

    s->top1++;

    s->ele[s->top1] = item;

**printf**("\nInserted item in Stack1 : %d",item);

}

//Push Operation on Stack2

**void** pushB( DStack \*s, **int** item )

{

**if**( s->top2 == s->top1 + 1 )

    {

**printf**("\nStack Overflow Stack2");

**return**;

    }

    s->top2--;

    s->ele[s->top2] = item;

**printf**("\nInserted item in Stack2 : %d",item);

}

//Pop Operation on Stack1

**int** popA( DStack \*s, **int** \*item )

{

**if**( s->top1 == -1 )

    {

**printf**("\nStack Underflow Stack1");

**return** -1;

    }

    \*item = s->ele[s->top1--];

**return** 0;

}

//Pop Operation on Stack2

**int** popB( DStack \*s, **int** \*item )

{

**if**( s->top2 == MAX )

    {

**printf**("\nStack Underflow Stack2");

**return** -1;

    }

    \*item = s->ele[s->top2++];

**return** 0;

}

**int** main()

{

**int** item = 0;

    DStack s;

    init(&s);

    pushA( &s, 10);

    pushA( &s, 20);

    pushA( &s, 30);

    pushB( &s, 40);

    pushB( &s, 50);

    pushB( &s, 60);

**if**( popA(&s, &item) == 0 )

**printf**("\nDeleted item From Stack1 : %d",item);

**if**( popA(&s, &item) == 0 )

**printf**("\nDeleted item From Stack1 : %d",item);

**if**( popA(&s, &item) == 0 )

**printf**("\nDeleted item From Stack1 : %d",item);

**if**( popB(&s, &item) == 0 )

**printf**("\nDeleted item From Stack2 : %d",item);

**if**( popB(&s, &item) == 0 )

**printf**("\nDeleted item From Stack2 : %d",item);

**if**( popB(&s, &item) == 0 )

**printf**("\nDeleted item From Stack2 : %d",item);

**printf**("\n");

**return** 0;

}

A. Pendeklarasian stack

Proses pendeklarasian stack adalah proses pembuatan struktur stack dalam memori. Karena stack dapat direpresentasikan dalam 2 cara, maka pendeklarasian stack pun ada 2 yaitu pendeklarasian stack menggunakan array dan single linked list, hanya pada bagain ini hanya dibahas stack menggunakan array :

Pendeklarasian stack yang menggunakan array.

Suatu stack memiliki beberapa bagian yaitu

* top yang menunjuk posisi data terakhir (top)
* elemen yang berisi data yang ada dalam stack. Bagian ini lah yang berbentuk array.
* maks\_elemen yaitu variable yang menunjuk maksimal banyaknya elemen dalam stack.

B. Inisialisasi

Inisialisasi stack adalah proses pembuatan suatu stack kosong. Adapun langkah-langkah

proses tersebut berdasarkan jenis penyimpanannya adalah :

**Inisialisasi stack yang menggunakan array.**

Proses inisialisasi untuk stack yang menggunakan array adalah dengan mengisi nilai

field top dengan 0 (nol) jika elemen pertama diawali dengan nomor 1.

C. Operasi IsEmpty

Operasi ini digunakan untuk memeriksa apakah stack dalam keadaan kosong. Operasi ini penting dilakukan dalam proses pop. Ketika suatu stack dalam keadaan kosong, maka proses pop tidak bisa dilakukan.

Adapun langkah-langkah operasi ini adalah :

Operasi IsEmpty pada stack yang menggunakan array.

Operasi ini dilakukan hanya dengan memeriksa field top. Jika top bernilai 0 (untuk elemen yang dimulai dengan index 1), maka berarti stack dalam keadaan empty (kosong) yang akan me-return-kan true (1) dan jika tidak berarti stack mempunyai isi dan me-return-kan nilai false (0)

D. Operasi IsFull

Operasi ini berguna untuk memeriksa keadaan stack apakah sudah penuh atau belum. Operasi ini akan menghasilkan nilai true (1) jika stack telah penuh dan akan menghasilkan nilai false (0) jika stack masih bisa ditambah. Langkah-langkah untuk operasi ini adalah:

Operasi IsFull pada stack yang menggunakan array.

Operasi ini akan memberikan nilai true (1) jika field top sama dengan field maks\_elemen (untuk array yang elemennya dimulai dari posisi 1).

E. Operasi Push

Operasi push adalah operasi dasar dari stack. Operasi ini berguna untuk menambah suatu elemen data baru pada stack dan disimpan pada posisi top yang akan mengakibatkan posisi top akan berubah. Langkah operasi ini adalah :

a. Operasi push pada stack yang menggunakan array.

Langkah operasi push dalam array adalah dengan :

* Periksa apakah stack penuh (isfull). Jika bernilai false/0 (tidak penuh) maka proses push dilaksanakan dan jika pemeriksaan ini bernilai true/1 (stack penuh), maka proses push digagalkan.
* Proses push-nya sendiri adalah dengan menambah field top dengan 1, kemudian elemen pada posisi top diisi dengan elemen data baru.

Untuk lebih jelas, perhatikan lagi gambar 4 mengenai representasi stack dengan array

F. Operasi Pop

Operasi pop adalah salah satu operasi paling dasar dari stack. Operasi ini berguna untuk mengambil elemen terakhir (top) dan kemudian menghapus elemen tersebut sehingga posisi top akan berpindah. Operasi ini biasanya dibuat dalam bentuk function yang mereturn-kan nilai sesuai data yang ada di top.

Operasi pop pada stack yang menggunakan array.

Langkah operasi pop pada stack yang menggunakan array adalah terlebih dahulu memeriksa apakah stack sedang keadaan kosong, jika tidak kosong maka data diambil pada posisi yang ditunjuk oleh posisi top, kemudian simpan dalam variable baru dengan nama data, kemudian posisi top0.

2.4 Stack dengan Array

Sesuai dengan sifat stack, pengambilan / penghapusan delemen dalam stack harus dimulai dari elemen teratas.

Operasi-operasi pada Stcak dengan Array

* Konstruktor

Fungsi ini membuat sebuah stack baru yang masih kosong. Konsepnya adalah bahwa Top menunjukkan elemen stack teratas. Jika Top bernilai -1, berarti tumpukan kosong.

* IsFul

Fungsi ini memeriksa apakah stack yang ada sudah penuh. Stack penuh jika stack penuh jika puncak stack terdapat tepat dibawah jumlah maksimum yang dapat ditampung stack atau dengan kata lain Top = MAX\_STACK -1.

* Push

Fungsi ini menambahkan sebuah elemen ke dalam stack dan tidak bias dilakukan lagi jika stack sudah penuh.

* IsEmpty

Fungsi menentukan apakah stack kosong atau tidak. Tanda bahwa stack kosong adalah Top bernilai kurang dari nol.

* Pop

Fungsi ini mengambil elemen teratas dari stack dengan syarat stack tidak boleh kosong.

* Clear

Fungsi ini mengosongkan stack dengan cara mengeset Top dengan -1. Jika Top bernilai kurang dari nol maka stack dianggap kosong

2.5 Double Stack dengan Array

Metode ini adalah teknik khusus yang dikembangkan untuk menghemat pemakaian memori dalam pembuatan dua stack dengan array. Intinya adalah penggunaan hanya sebuah array untuk menampung dua stack.

Tampak jelas bahwa sebuah array dapat dibagi untuk dua stack, stack 1 bergerak ke atas dan stack 2 bergerak ke bawah. Jika Top1 (elemen teratas dari Stack 1) bertemu dengan Top 2 (elemen teratas dari Stack 2) maka double stack telah penuh.

Implementasi double stack dengan array adalah dengan memanfaatkan operasioperasi yang tidak berbeda jauh dengan operasi single stack dengan array.

Operasi-operasi Double Stack Array

* Konstruktor

Fungsi ini membuat stack baru yang masih kosong. Top[0] diset dengan -1 dan Top[1] diset dengan MAX\_STACK.

* IsFull

Fungsi ini memeriksa apakah double stack sudah penuh. Stack dianggap penuh jika Top[0] dan Top[1] bersentuhan sehingga stack tida memiliki ruang kosong. Dengan kata lain, (Top[0] + 1) > Top[1].

* Push

Fungsi ini memasukkan sebuah elemen ke salah satu stack.

* IsEmpty

Fungsi memeriksa apakah stack pertama atau stack kedua kosong. Stack pertamadianggap kosong jika puncak stack bernilai kurang dari nol, sedangkan stack kedua dianggap kosong jika puncak stack sama atau melebihi MAX\_STACK.

* Pop

Fungsi ini mengeluarkan elemen teratas dari salah satu stack

* Clear

Fungsi ini mengosongkan salah satu stack.

Contoh program:

#include “stdio.h”

void main()

{

int stack[100];

int top=-1;

int pilih, i;

do

{

printf(“MENU\n”);

printf(“1. PUSH\n2. POP\n3. VIEW\n4. EXIT\n”);

printf(“Pilih = “); scanf(“%d”, &pilih);

switch(pilih)

{

case 1://push

if (top > 100)

printf(“Stack penuh!\n”);

else

{ printf(“Data = “); scanf(“%d”, &stack[top+1]);

top++;

}

break;

case 2://pop

if (top < 0)

printf(“Stack kosong!\n”);

else

{

printf(“Data keluar = %d\n”, stack[top]);

top–;

}

break;

case 3://view

for(i=top; i>=0; i–)

printf(“%d “, stack[i]);

printf(“\n”);

break;

case 4:

printf(“Exit…\n”);

break;

}

}while (pilih!=4);

}

2.6 Stack dengan Single Linked List

Selain implementasi stack dengan array seperti telah dijelasnkan sebelumnya, ada cara lain untuk mengimplementasi stack dalam C++, yakni dengan single linked list. Keunggulannya dibandingkan array tebtu saja adalah penggunaan alokasi memori yang dinamis sehingga menghindari pemborosan memori. Misalnya saja pada stack dengan array disediakan tempat untuk stack berisi 150 elemen, sementara ketika dipakai oleh user stack hanya diisi 50 elemen, maka telah terjadi pemborosan memori untuk sisa 100 elemen, yang tak terpakai. Dengan penggunaan linked list maka tempat yang disediakan akan sesuai dengan banyaknya elemen yang mengisi stack. Oleh karena itu pula dalam stack dengan linked list tidak ada istilah full, sebab biasanya program tidak menentukan jumlah elemen stack yang mungkin ada (kecuali jika sudah dibatasi oleh pembuatnya). Namun demikian sebenarnya stack ini pun memiliki batas kapasitas, yakni dibatasi oleh jumlah memori yang tersedia.

Operasi-operasi untuk Stack dengan Linked List

* Konstruktor

Fungsi ini membuat stack baru yang kosong. Stack adalah kosong jika Top tidak menunjuk apa pun (bernilai NULL).

* IsEmpty

Fungsi memeriksa apakah stack yang adamasih kosong.

* Push

Fungsi memasukkan elemen baru ke dalam stack. Push di sini mirip dengan insert dalam single linked list biasa.

* Pop

Fungsi ini mengeluarkan elemen teratas dari stack.

* Clear

Fungsi ini akan menghapus stack yang ada.

**Contoh program stack:**

Contoh 1:

#include <stdio.h>

#include <conio.h>

#define max 5

typedef struct {

int top;

int data[max+1];

}stack;

stack tumpukan;

void createEmpty();

int IsEmpty();

int IsFull();

void push(int x);

void pop();

main(){

int lagi;

int input;

int pilih;

createEmpty();

pilih = 0;

while (pilih != 5){

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf(" MENU UTAMA\n");

//Printf dan Puts memiliki Fungsi yang sama

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

puts("1: Cek kondisi Stack");

puts("2: Tambah data (Pust)");

puts("3: Keluarkan isi stack (Pop)");

puts("4: Kosongkan stack");

puts("5: Keluar");

printf("Pilihan Anda: ");

scanf("%d",&pilih);

switch(pilih){

case 1:

if (IsEmpty() == 1)

puts("Stack masih kosong");

else if ((IsEmpty() == 0) && (IsFull() == 0))

puts("Stack sudah terisi (belum penuh)");

else

puts("Stack sudah penuh");

break;

case 2:

if (IsFull() == 1)

puts("Stack Penuh.");

else

{

printf("Masukkan data: ");

scanf("%d",&input);

push(input);

printf("Top baru: %d\n",tumpukan.data[tumpukan.top]);

printf("IsFull: %d\n",IsFull());

printf("IsEmpty: %d\n",IsEmpty());

}

break;

case 3:

printf("Top yang dikeluarkan bernilai: %d \n",tumpukan.data[tumpukan.top]);

pop();

break;

case 4:

while (IsEmpty() == 0)

{

printf("Top yang dikeluarkan bernilai: %d \n",tumpukan.data[tumpukan.top]);

pop();

}

puts("Stack sudah kosong");

break;

case 5:

puts("---Terimakasih || MARKIJAR.Com---");

break;

}

}

}

void createEmpty(){

tumpukan.top = 0;

}

void push(int x){

tumpukan.top = tumpukan.top + 1;

tumpukan.data[tumpukan.top] = x;

}

void pop(){

tumpukan.top = tumpukan.top - 1;

}

int IsEmpty(){

if (tumpukan.top == 0)

return 1;

else

return 0;

}

int IsFull(){

if (tumpukan.top == max)

return 1;

else

return 0;

}

Output :



Contoh 2:

#include "stdio.h";

#include "ctype.h";

#include "stdlib.h"

#define SIZE 40

int stack[SIZE];

int top=-1;

int pop();

void push(int);

int evaluate(int , int o,char );

void push(int n)

{

if(top == SIZE-1)

{

printf("Stack is full\n");

return;

}

else

{

top=top+1;

stack[top]=n;

printf("Pushed element is %d\n",n);

}

}

int pop()

{

int n;

if(top == -1)

{

printf("Stack is empty\n");

return;

}

else

{

n=stack[top];

top=top-1;

printf("The poped element is %d\n",n);

return(n);

}

}

int evaluate(int op1, int op2,char ch)

{

printf("op1=%d op2=%d ch=%c\n",op1,op2,ch);

int n;

if (op1 < op2)

{

n=op1;

op1=op2;

op2=n;

}

if(ch=='+')

n=op1+op2;

else if(ch=='-')

n=op1-op2;

else if(ch=='\*')

n=op1\*op2;

else if(ch=='/')

n=op1/op2;

else if(ch=='%')

n=op1%op2;

else

{

printf("The operator is not identified\n");

exit(0);

}

printf("n=%d\n",n);

return(n);

}

int main()

{

char str[50],ch,ch1;

int i=0,n,op1,op2;

printf("Enter the Postfix string\n");

scanf("%s",str);

ch=str[i];

while(ch!='\0')

{

printf("%c",ch);

if(ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5')//

if(isdigit(ch))

{

n=ch-'0';

push(n);

}

else

{

op1=pop();

op2=pop();

n=evaluate(op1,op2,ch);

push(n);

}

ch=str[++i];

}

printf("%d\n",pop());

return;

}

Contoh 3:

#include<stdio.h>

#include<conio.h>

#define MAX 50

#define true 1

#define false 0

char stack[MAX];

int top;

void init(void);

int full (void);

int empty (void);

char pop (void);

void clear (void);

void push (char info);

void baca (void);

main()

{

char pilih,khol;

printf("----Demo Operasi Single Stack----\n");

init();

do

{

printf("operasi single stack:\n");

printf("[1] PUSH\n");

printf("[2] POP\n");

printf("[3] CLEAR\n");

printf("[4] BACA\n");

printf("[5] SELESAI...\n");

printf(" Pilihan : ");scanf("%s",&pilih);

switch(pilih)

{

case '1': printf("PUSH ");scanf("%s",&khol);push(khol);break;

case '2': khol=pop();printf("POP %c\n",khol);break;

case '3': clear();break;

case '4': baca();break;

case '5': break;

default : printf("Salah pilih...\n");

}

printf("\n");

}while(pilih!='5');

getche();

}

void init(void)

{

top=0;

}

void push(char info)

{

if(full()!=true)

{ top++;

stack[top]=info; }

else printf("Stack overflow...\n");

}

char pop(void)

{

char info;

if(empty()!=true)

{ info=stack[top];

top--;

return(info);

}

else printf("Stack underflow...\n");

}

void clear(void)

{

top=0;

}

int full(void)

{

if(top==MAX) return(true);

else return(false);

}

int empty(void)

{

if(top==0) return(true);

else return(false);

}

void baca(void)

{

int i;

printf("isi stack : ");

if(top>0)

{

for(i=1;i<=top;i++)

printf("%c ",stack[i]);

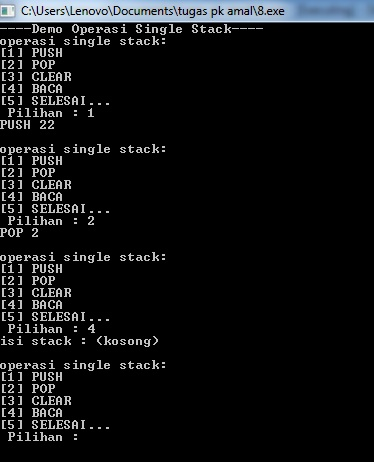
}

else printf("(kosong)");

printf("\n");

}

Output :



Kelebihan

* membantu mengelola data dengan metode LIFO
* secara otomatis membersihkan objek
* tidak mudah rusak
* ukuran variabel tidak dapat diubah
* mengontrol memori secara mandiri

Kekurangan

* memori stack sangat terbatas
* ada kemungkinan stack akan meluap atau overflow jika terlalu banyak objek
* tidak memungkinkan akses acak, karena harus mengeluarkan tumpukan paling atas terlebih dahulu untuk mengakses tumpukan paling bawah

BAB III

PENUTUP

**Kesimpulan**

Stack adalah suatu koleksi atau kumpulan item data yang terorganisasi dalam bentuk urutan linier, yang operasi pemasukan dan penghapusan datanya dilakukan pada salah satu sisinya. Pada stack, jarang sekali dilakukan traversal, karena keunikan stack justru pada operasi yang hanya menyangkut elemen TOP. Namun dibutuhkan traversal misalnya untuk mencetak isi stack. Pada stack, elemen yang diproses hanya elemen pada TOP. Maka hampir tidak pernah dilakukan search atau bersifat LIFO (Last In First Out). Operasi-operasi pada Stack :

Create(Stack)Operasi Create(Stack) digunakan untuk membuat suatu stack baru dengan nama stack, yang nilai elemen saat stack tersebut dibuat adalah NOEL(S) = 0, TOP(S) = NULL (tidak terdefinisikan).IsEmpty(Stack)Operasi ini merupakan operasi untuk mencek isi dari suatu stack dalam keadaan kosong atau berisi.Operasi ini memiliki kondisi boolean yaitu :

True jika stack tersebut kosong atau dapat dikatakan NOEL(S) = 0.

False jika stack tersebut tidak dalam kondisi kosong atau dapat dikatakan NOEL(S)

Push(Stack, Elemen)

Operasi ini merupakan operasi untuk menambahkan satu elemen dengan nilai X pada puncak suatu stack, sehingga posisi TOP(S) akan bernilai X, penerapan operasi push pasa suatu stack S akan berakibat overflow jika NOEL(S) dari stack tersebut telah bernilai maksimum

**Saran**

Penulis menyadari bahwa dalam pembuatan makalah ini masih banyak kesalahan dan kekurangan, baik dari segi penulisan maupun dari segi pemaparan. Maka dari itu pemakalah menerima dan mengharapkan kritik dan saran dari pembaca yang sifatnya membangun, agar makalah ini dapat menjadi lebih baik dan menjadi rujukan bagi pembaca nantinya.

**DAFTAR PUSTAKA**

1. <https://www.includehelp.com/c/>
2. <http://aditajulian05.blogspot.com/2017/04/makalah-struktur-data-stack.html>
3. <http://1412170016-mkholilar.blogspot.com/2018/05/stack-tumpukan-bahasa-c.html>
4. [https://www.markijar.com/2015/05/stack-tumpukan-dalam-bahasa c.html#:~:text=Stack%20merupakan%20jenis%20linked%20list,terakhir%20akan%20diproses%20%2F%20dikeluarkan%20pertama](https://www.markijar.com/2015/05/stack-tumpukan-dalam-bahasa%20c.html#:~:text=Stack%20merupakan%20jenis%20linked%20list,terakhir%20akan%20diproses%20%2F%20dikeluarkan%20pertama).
5. <https://afteracademy.com/blog/stack-and-its-basic-operations>
6. <https://www.academia.edu/8782802/CONTOH_PROGRAM_STACK_include>
7. <https://bluejundi.wordpress.com/2011/05/01/contoh-program-stack-memakai-array-bahasa-c/>
8. <https://daismabali.com/artikel_detail/64/1/Mengenal-Konsep-Stack-dalam-Struktur-Data-Beserta-Ilustrasi-Dan-Contoh-Pemrogramannya.html>