NAMA : RANITA SARI
NPM : 175100004
PROGRAM STUDI : SISTEM INFOMASI
MATA UJI : STRUKTUR DATA
DOSEN : ARIE SETYA PURA,S.KOM.,M.T.I

ARTIKEL STRUKTUR DATA | STACK (TUMPUKAN)

Artikel 1

Pengertian stack/tumpukan:

Kumpulan items yang teratur dimana items baru akan dimasukkan ke dan sebuah items akan dikeluarkan dari satu ujung yang sama, yaitu dari TOP sebuah stack. Struktur data linier dimana hanya bagian TOP-nya saja yang bisa diakses. Bersifat LIFO = Last In First Out. Bisa diimplementasikan menggunakan array atau Linked List.

Ilustrasi Stack

Terdapat dua buah kotak yang ditumpuk, kotak yang satu akan ditumpuk diatas kotak yang lainnya. Jika kemudian stack 2 kotak tadi, ditambah kotak ketiga, keempat, kelima, dan seterusnya, maka akan diperoleh sebuah stack kotak yang terdiri dari N kotak

OPERASI PADA STACK

2 operasi dasar yang bisa dilaksanakan pada sebuah stack, yaitu:

ž Operasi Push (menyisipkan data)

memasukkan data ke dalam stack

ž Operasi Pop (menghapus data)

menghapus elemen yang terletak pada posisi paling atas dari sebuah stack

langkah-langkah :

1. buat stack (stack) – create

ž membuat sebuah stack baru yang masih kosong

ž spesifikasi:

– tujuan : mendefinisikan stack yang kosong

– input : stack

– syarat awal : tidak ada

– output stack : – (kosong)

- syarat akhir : stack dalam keadaan kosong

2. stack kosong (stack) – empty

ž fungsi untuk menentukan apakah stack dalam keadaan kosong atau tidak

ž spesifikasi:

- tujuan : mengecek apakah stack dalam keadaan kosong

- input : stack

- syarat awal : tidak ada

- output : boolean

- syarat akhir : stack kosong bernilai true jika stack dalam keadaan kosong

3. stack penuh (stack) – full

ž fungsi untuk memeriksa apakah stack yang ada sudah penuh

ž spesifikasi:

- tujuan : mengecek apakah stack dalam keadaan penuh

- input : stack

- syarat awal : tidak ada

- output : boolean

- syarat akhir : stack penuh bernilai true jika stack dalam keadaan penuh

4. push (stack, info baru)

ž menambahkan sebuah elemen kedalam stack.

ž spesifikasi:

- tujuan : menambahkan elemen, info baru pada stack pada posisi paling atas

- input : stack dan Info baru

- syarat awal : stack tidak penuh

- output : stack

- syarat akhir : stack bertambah satu elemen

5. pop (stack, info pop)

ž mengambil elemen teratas dari stack

ž spesifikasi:

- tujuan : mengeluarkan elemen dari stack yang berada pada posisi paling atas

- input : stack

- syarat awal : stack tidak kosong

- output : stack dalam info pop

syarat akhir : stack berkurang satu elemen

contoh source code tentang progarn yang mampu melakukan pengecekan terhadap sebuah string yang di dalamnya mengandung satu atau beberapa pasang tanda delimiter yaitu tanda kurung biasa buka dan tutup '()', tanda kurung kurawal buaka dan tutup '{}',serta tanda kurung siku buka dan tutup '[']'. Program mengeluarkan pernyataan

'benar', jika struktur string sudah benar dan sebaliknya menguarkn pernyataan 'salah' jika struktur string salah. (ada tanda delimiter yang tidak mempunyai pasangan).
Program harus mengimplementasikan struktur data stack dengan menggunakan array.

Contoh:

input:a{bc[d]e}f(g) output: BENAR

input:a{bc[d]e}f(g) outpur: SALAH

```
#include "stdio.h"
```

```
char data[50];char temp;
```

```
char stack[50];int awal;
```

```
int taruh;int benar=1;
```

```
void main()
```

```
{
```

```
char jwb;
```

```
do
```

```
{
```

```
printf("\tprogram delimiter checker\n\n");
```

```
printf("Input data anda: ");
```

```
scanf("%s",&data);
```

```
for(int i=0;i<=100;i++)
```

```
{
```

```
if(data[i]=='{' || data[i]=='(' || data[i]=='[')
```

```
{
```

```
stack[awal+1]=data[i]; awal++;
```

```
}
```

```
else if(data[i]=='}' || data[i]==')' || data[i]==']')
```

```
{
```

```
temp=stack[awal];stack[awal]=NULL;awal--;
```

```
if(temp=='{')
```

```
{ taruh='}'; }
```

```
else if(temp=='(')
```

```
{ taruh=')'; }
```

```
else if(temp=='[')
```

```
{ taruh=']'; }
```

```
if(taruh!=data[i])
```

```
{ benar=0; }
```

```
}
```

```
}
```

```
if(benar==1)
```

```
{
```

```

printf("\nhasil output: BENAR\n");
}
else
{
printf("\nhasil output : SALAH\n");
}
printf("apakah anda ingin mencoba lagi ?? [Y/N] : \n"); scanf("%s",&jwb);
printf("\n");
}while ((jwb=='y') || (jwb=='Y'));
}

```

semoga bermanfaat

Artikel 2

1. Jelaskan tentang stack, uraikan dengan deskripsi (fungsi2 dasar yg dimiliki), permodelan, dan contoh nyata penerapannya?

Jawab:

STACK adalah salah satu list linear dalam struktur data yang digunakan untuk menyimpan dan mengambil data dengan konsep LIFO (Last In First Out). Dimana dalam stack ini kumpulan data yang masuk diletakkan di atas data yang lain. Dan berdasar konsep LIFO maka data yang terakhir kali disimpan dalam stack akan menjadi data yang pertama kali diambil. Dalam prosesnya, untuk memasukkan sebuah data ke dalam stack atau dengan kata lain ke bagian atas dari sebuah tumpukan digunakan perintah push. Dan untuk memindahkan data dari tempat tersebut digunakan perintah pop. Sedangkan dalam penyajiannya, stack bisa memakai array atau linked list.

Dari gambar di atas bisa dilihat bahwa data di dalam stack hanya bisa dimasukkan atau diambil melalui bagian ujung saja/top.

Operasi-operasi dasar pada stack :

a. Cek Stack kosong (Isempyt)

Fungsi yang melakukan pengecekan apakah stack dalam kondisi kosong.

```
public int isempty()
```

```
if(posisi==0)
```

```
System.out.println("Data Kosong");
```

```
return 1;
```

```
else
```

```
System.out.println("Data ada");
```

```
return 0;
```

b. Cek Stack penuh (full)

Fungsi yang melakukan pengecekan apakah stack dalam kondisi penuh atau tidak.

```
public int full()

if(posisi==MAX)

System.out.println("Stack Penuh");

return 0;

else

return 1;
```

c. Operasi Push

Operasi push dalam stack adalah operasi yang memasukkan elemen yang akan diletakkan pada posisi teratas dari tumpukan.

```
public void push (int data)

if(posisi<MAX_STACK)

isistack[++posisi] = data;
```

d. Operasi Pop

Operasi pop dalam stack adalah operasi untuk mengambil/menghapus elemen yang terletak pada posisi paling atas dari sebuah tumpukan.

```
public void pop()

int y=0;

if(posisi != 0)

isistack[posisi-]=y;
```

e. Cek posisi Teratas (Peek)

Operasi peek digunakan untuk mengecek posisi teratas dalam stack.

```
public void peek()
```

```
System.out.print("Posisi Atas= ");
```

```
if(posisi != 0)
```

```
System.out.print(isistack[posisi]);
```

```
System.out.println();
```

Penerapan stack :

Stack digunakan untuk menuliskan ungkapan menggunakan notasi tertentu (Notasi Polish). Biasanya ungkapan yang digunakan adalah ungkapan numeris. Sebagai contoh ungkapan $(A + B) * (C - D)$ apabila ditulis dengan menggunakan notasi Polish menjadi $* + A B - C D$.

Artikel 3

Pada saat ukuran stack, kalau kita teruskan menambah data lagi, akan terjadi overflow. Dengan demikian perlu data tambahan untuk mencatat posisi ujung stack. Dengan kebutuhan seperti ini, kita dapat menyajikan stack dengan menggunakan tipe data struktur (struct) yang terdiri dari dua field. Field pertama bertipe array untuk menyimpan elemen stack, medan kedua bertipe integer untuk mencatat posisi ujung stack. Sebagai ilustrasi, "Benda yang terakhir masuk ke dalam stack akan menjadi yang pertama keluar dari stack. Hal ini asemaam kebalikan dari postingan saya sebelumnya tentang queue

Saya akan membuat suatu program yang melakukan pembalikan kalimat dengan menggunakan stack

Contoh:

source code

```
#include <iostream.h>
```

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

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

```
#include <conio.h>
```

```
#define maxstack 200
```

```
struct STACK //membuat jenis data abstrak 'STACK'
```

```
{
```

```
int top;
```

```
char data[maxstack];
```

```
};
```

```

char dta[maxstack];
struct STACK stackbaru;

void inisialisasi()
{
    stackbaru.top = -1;
}

bool isfull() //menanyakan penuhkah?
{
    if (stackbaru.top == maxstack-1) return true;
    else return false;
}

bool isempty() //menanyakan kosongkah?
{
    if (stackbaru.top == -1) return true;
    else return false;
}

void push(char dta) //mengisi stack (menyiapkan data)
{
    if (isfull() == false)
    {
        stackbaru.top++;
        stackbaru.data[stackbaru.top]=dta;
    }
    else
    {
        puts ("\nMaaf Stack penuh");
    }
}

void pop() //mengambil isi stack
{
    while (isempty() == false)
    {
        cout<<stackbaru.data[stackbaru.top];
        stackbaru.top--;
    }
}

```



```

}

void print() //mencetak stack
{
    cout<<" ";
    for (int i=0; i<=stackbaru.top; i++)
    {
        cout<<stackbaru.data[i];
    }
}

void clear()
{
    stackbaru.top = -1;
}

void main()
{
    char kata[200];

    printf("-- PROGRAM PEMBALIK KALIMAT -- \n\n");
    printf("Masukkan kalimat yang Anda inginkan: \n");

    gets(kata);

    for(int i=0; kata[i]; i++)
        push(kata[i]);
    printf("----- \n\n");

    print();
    cout<<" menjadi ";
    pop();
    cout<<"\n";
    getch();
}

```

Tampilan

program sorting dengan menggunakan stack

source code;

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#define maxstack 6

//using namespace std;

int top=-1;
struct STACK
{
    int top;
    float data[4];
};

float dta;
struct STACK stackbaru;

bool isfull ()
{
    if (stackbaru.top == maxstack)
        return true;
    else return false;
}
bool isempty()
{
    if (stackbaru.top == -1)
        return true;
    else
```

```

return false;
}
void push (float dta)
{
if (isfull() == true)
{
cout<<"maaf, stack dalam keadaan penuh";
getche();
}else{
top++;
stackbaru.top++;
stackbaru.data[top] = dta;
}
}
void pop ()
{
if (isempty () == true){
cout<<"data pada stack dalam keadaan kosong";
getche();
}else {
cout<<"data yang diambil : "<<stackbaru.data[top]<<endl;
top--;
}
}
void print () // mencetak stack
{
for (int i =top; i>=0;i-){
cout<<stackbaru.data[i]<<" ";
}
}
void clear ()
{
top=-1;
}

void inisialisasi ()
{
stackbaru.top=-1;
}

```

```

void Sort (int Array[])
{
    int i,j,kecil,temp;
    for (i=0; i<5; i++)
    {
        {kecil=i;
        for (j=i+1; j<5; j++)
        {
            if(Array[kecil]>Array[j])
            {kecil =j;}
        }
        temp = Array [i];
        Array[i] = Array[kecil];
        Array[kecil] = temp;

    }
}
}

```

```

int main ()
{
    void inisialisasi ();
    char menu;
    char ulang;
    int data [5];
    int i,j;
    do {
        clrscr;
        printf("Menu : \n");
        puts("1. pop stack");
        puts("2. Push data dan sorting ascending");
        puts("3. cetak sorting descending");
        puts("4. Bersihkan stack");
        puts("5. Exit");

        cout<<"Menu pilihan anda : ";
        cin>>menu;

        if (menu == '1'){
            pop ();

```

```

getche ();
ulang == 'y';

}
else if (menu == '2')
{
cout<<"sorting ascending\n";
for(i=0; i<5; i++)
{
cout<<"input nilai ke "<<(i+1)<<" =";
cin>>data[i];
}
Sort(data);
cout<<"sorting ascending :";
for(j=0 ; j<5 ; j++)
{cout<<data[j];
push(data[j]);
}
cout<<"\n";
getche();

ulang = 'y';
clrscr;
}else if (menu == '3')
{
cout<<"sorting descending\n";
print ();
cout<<"\n";
getche();
}else if (menu == '4')
{
clear ();
cout<<"Ulang? (y/t)";
cin>>ulang;
}else if (menu == '5')
{

}
}while (ulang == 'Y' || ulang == 'y');
getche();

```

}
tampilan

```
D:\UM\KULIAH\SEMESTER 3\ASD PRAKTIKUM\MODUL 4\tugas2.exe
Menu :
1. pop stack
2. Push data dan sorting ascending
3. cetak sorting descending
4. Bersihkan stack
5. Exit
Menu pilihan anda : 2
sorting ascending
input nilai ke 1 =1
input nilai ke 2 =2
input nilai ke 3 =3
input nilai ke 4 =4
input nilai ke 5 =5
sorting ascending :12345
Menu :
1. pop stack
2. Push data dan sorting ascending
3. cetak sorting descending
4. Bersihkan stack
5. Exit
Menu pilihan anda : 3
sorting descending
5 4 3 2 1
Menu :
1. pop stack
2. Push data dan sorting ascending
3. cetak sorting descending
4. Bersihkan stack
5. Exit
Menu pilihan anda : _
```