MAKALAH ALGORITMA
TENTANG
STACK (TUMPUKAN)

Angga Kresnabayu – 140810160001

Dzakia Rayhana – 140810160015

Syafira Fitra Annisa – 140810160047

Shofiyyah Nadhiroh – 140810160057

Patricia Joanne – 140810160065

Dosen Pembimbing:

Erick Paulus, M.Kom.

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

DEPARTEMEN ILMU KOMPUTER

UNIVERSITAS PADJADJARAN

2017

KATA PENGANTAR

Puji dan syukur tim penyusun panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan kekuatan tim penyusun untuk menyusun makalah algoritma tentang stack atau tumpukan.

Makalah ini disusun sebagai salah satu tugas mata kuliah Struktur Data berdasarkan informasi yang didapat dan hasil diskusi sederhana dan pembelajaran di kelas yang telah dilakukan oleh tim penyusun.

Tim penyusun mengucapkan terima kasih kepada dosen yang telah membimbing tim penyusun, baik itu secara langsung maupun tidak langsung serta Universitas Padjadjaran yang telah menyediakan fasilitas WiFi untuk menunjang tim penyusun menuntaskan makalah ini.

Tim penyusun menyadari bahwa makalah ini masih banyak kekurangan dan kelemahannya, baik dalam isi maupun sistematikanya. Oleh sebab itu, tim penyusun sangat mengharapkan kritik dan saran yang membangun dalam upaya menyempurnakan makalah ini. Tim penyusun juga mengharapkan agar makalah ini dapat memberikan manfaat bagi pembacanya.

Jatinangor, 2 Mei 2017

Tim Penyusun

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Salah satu konsep yang sangat berguna di dalam ilmu komputer adalah satu bentuk struktur data yang disebut dengan tumpukan. Dalam hal ini kami mencoba mengenali mengapa stack (tumpukan) sangat berguna dan memainkan peranan penting dalam pemrograman dan bahasa pemrograman.

1.2. Tujuan

Adapun tujuan dari penyusunan makalah ini adalah untuk mengetahui sejauh mana pemahaman kami mengenai stack.

1.3. Rumusan Masalah

1. Apa yang dimaksud dengan Stack (Tumpukan)?
2. Apa definisi dari Stack (Tumpukan)?
3. Bagaimana cara mendeklarasikan Stack (Tumpukan)?
4. Bagaimana cara menjalankan operasi pada Stack (Tumpukan)? (Push, Pop, Size, Empty, Full)

BAB II
ISI

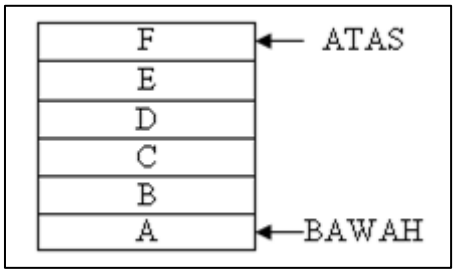
2.1. Teori Umum

1. Pengertian Stack

Stack pada Struktur Data adalah sekumpulan data yang seolah-olah diletakkan di atas data yang lain atau suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (top) saja.

Stack dapat diilustrasikan dengan 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.

Stack bersifat LIFO (*Last In First Out*) artinya data yang terakhir masuk ke dalam stack akan menjadi yang pertama keluar dari stack.



2. Definisi Stack

Cara mendefinisikan stack ada dua cara, yaitu menggunakan array dan list berkait. Yang membedakan keduanya adalah pada isi fungsi operasinya.

- Definisikan konstanta maxElemen untuk menyimpan maksimum isi stack
- Definisikan stack dengan menggunakan struct
- Buatlah variabel array data sebagai implementasi stack
- Deklarasikan operasi-operasi/function dan buat implementasinya

3. Deklarasi Stack

- Menggunakan Array

```
#include<iostream>
using namespace std;

const int maxElemen = 255;
```

```
struct stack{
    int isi[maxElemen];
    int top;
};
stack s;

void createStack(stack& s){
    s.top=-1;
}
```

- Menggunakan List Berkait

```
#include <iostream.h>
using namespace std;

const int maxElmt=255;

struct elmtList{
    char isi;
    elmtList*next;
};

typedef elmtList* pointer;
typedef pointer list;

void createStack(list&top){
    top=NULL;
}

void createElmt(pointer&pBaru,char k){
    pBaru=new elmtList;
    pBaru->isi=k;
    cout<<pBaru->isi;
    pBaru->next=NULL;
}
```

4. Inisialisasi Stack

   Pada mulanya isi top dengan -1, karena dimulai dari 0, yang berarti stack adalah kosong. Top adalah suatu variabel penanda dalam STACK yang menunjukkan elemen teratas Stack sekarang. Top of Stack akan selalu bergerak hingga mencapai maxElemen sehingga menyebabkan stack penuh.

5. Operasi Stack

   Operasi-operasi yang biasanya tredapat pada **Stack** yaitu:

   - **Push**: digunakan untuk menambah item pada tumpukan stack paling atas

- **Pop**: digunakan untuk mengambil item pada tumpukan stack paling atas
- **Empty**: digunakan untuk mengecek apakah stack sudah kosong
- **Full**: digunakan untuk mengecek apakah stack sudah penuh
- **Sizeof**: digunakan untuk mengukur panjang stack

2.2. Operasi Stack dengan Array

1. Jenis-jenis operasi

- Push

  Ada 2 kasus yang perlu ditangani yaitu:

  1. Stack sudah penuh dengan ciri 'nilai Atas = maxElemen-1', maka tidak ada penambahan elemen
  2. Stack belum penuh, maka langkah yang dilakukan adalah :
     - Menaikkan posisi 'Atas' ke atas (Atas++)
     - Menyisipkan elemen baru ke posisi 'Atas' yang baru tersebut

```
void push(stack& T, char X){
    if (T.atas == maxElemen-1){
        cout << "TUMPUKAN SUDAH PENUH" << endl;
    }
    else {
        T.Atas = T.Atas+1;
        T.Isi[T.Atas] = X;
    }
}
```

- Pop

  Ada 2 kasus yang harus ditangani yaitu:

  1. Stack sudah kosong, dengan ciri 'nilai Atas = 0', maka tidak ada penghapusan
  2. Stack masih ada isi, maka langkah yang dilakukan adalah menurunkan posisi 'Atas' ke bawah (Atas--)

```
void pop(stack& T){
    if (T.Atas == 0){
        cout << "TUMPUKAN SUDAH KOSONG" << endl;
    }
    else {
        T.Atas = T.Atas-1;
    }
}
```

- Empty

```
int empty (stack T){
    if (T.Atas == -1)
        return 1;
    else
        return 0;
}
```

- Full

```
int full (stack T){
    if (T.Atas == maxElemen-1)
        return 1;
    else
        return 0;
}
```

- Sizeof

```
int SizeOf(const char *x){
    int panjang = strlen(x);
    return panjang;
}
```

2. Contoh pengimplementasian

- Membalikkan kalimat

```
/*
                PROGRAM MEMBALIK KALIMAT
*****************************************************/
#include <iostream.h>
#include <string.h>
using namespace std;

const int maxElemen = 255;
struct stack{
    char Isi[maxElemen];
    int Atas;
};
stack T;

void createStack(stack& T){
    T.Atas = -1;
}
int full (stack T){
    if (T.Atas == maxElemen-1)
        return 1;
    else
        return 0;
}
```

```
int empty (stack T){
    if (T.Atas == -1)
        return 1;
    else
        return 0;
}
int SizeOf(const char *x){
    int panjang = strlen(x);
    return panjang;
}
void push(stack& T, char x){
    if (T.Atas == maxElemen-1){
        cout << "TUMPUKAN SUDAH PENUH" << endl;
    }
    else {
        T.Atas = T.Atas+1;
        T.Isi[T.Atas] = x;
    }
}
```

```
char pop(stack& T){
    char hasil;
    if (T.Atas < 0){
        cout << "TUMPUKAN SUDAH KOSONG" << endl;
    }
    else {
        hasil = T.Isi[T.Atas];
        T.Atas = T.Atas-1;
    }
    return hasil;
}
```

```
main(){
    createStack (T);
    char kalimat[maxElemen];
    cout << "\n\nTUMPUKAN UNTUK MEMBALIK KALIMAT";
    cout << "\n---------------------------------" << endl;
    cout << "Isikan sembarang kalimat : " << endl;
    cin.getline(kalimat,maxElemen);
    cout << "\nKalimat Asli : " << endl << kalimat;
    cout << "\n\nSetelah dibalik : " << endl;
    for (int i=0;i<SizeOf(kalimat);i++){
        push (T,kalimat[i]);
    }
    for (int i=0;i<SizeOf(kalimat);i++){
        cout << pop(T);
    }
}
```

```
TUMPUKAN UNTUK MEMBALIK KALIMAT
--------------------------------
Isikan sembarang kalimat :
AKU ANAK SOLEH

Kalimat Asli :
AKU ANAK SOLEH

Setelah dibalik :
HELOS KANA UKA

Terminated with return code 0
Press any key to continue ...
```

2.3. Operasi Stack dengan List Berkait

1. Jenis-jenis operasi

- Push

  Ada 2 kasus yang perlu ditangani yaitu:

  1. Stack masih kosong
  2. Stack belum penuh, penambahan dilakukan di paling atas

```
void push(list&top, pointer pBaru){
    if (top==NULL)
        top=pBaru;
    else{
        pBaru->next=top;
        top=pBaru;
    }
}
```

- Pop

  Ada 3 kasus yang perlu ditangani yaitu:

  1. Stack kosong, tidak ada yang dihapus
  2. Stack berisi hanya satu nilai
  3. Stack berisi lebih dari satu nilai

```
void pop(list&top, pointer&pHapus){
    if (top==NULL){
        pHapus=NULL;
        cout<<"stack kosong, tidak ada yang dihapus"<<endl;
    }
    else if(top->next==NULL){
        pHapus=top;
        top=NULL;
    }
    else{
        pHapus=top;
        top=top->next;
        pHapus->next=NULL;
    }
    cout<<pHapus->isi;
}
```

- Sizeof

```
int panjangElmt (char kalimat[maxElmt]){
    return strlen(kalimat);
}
```

2. Contoh pengimplementasian
    - Membalikkan kalimat

```
#include <iostream.h>
using namespace std;

const int maxElmt=255;

struct elmtList{
    char isi;
    elmtList*next;
};

typedef elmtList* pointer;
typedef pointer list;

void createStack(list&top){
    top=NULL;
}

void createElmt(pointer&pBaru,char k){
    pBaru=new elmtList;
    pBaru->isi=k;
    cout<<pBaru->isi;
    pBaru->next=NULL;
}
```

```
void push(list&top, pointer pBaru){
    if (top==NULL)
        top=pBaru;
    else{
        pBaru->next=top;
        top=pBaru;
    }
}

void pop(list&top, pointer&pHapus){
    if (top==NULL){
        pHapus=NULL;
        cout<<"stack kosong, tidak ada yang dihapus"<<endl;
    }
    else if(top->next==NULL){
        pHapus=top;
        top=NULL;
    }
    else{
        pHapus=top;
        top=top->next;
        pHapus->next=NULL;
    }
    cout<<pHapus->isi;
}
```

```
main(){
    list top;
    pointer s,pHapus;
    char kalimat[maxElmt];

    cout<<"Program pembalik kalimat dengan List Berkait"<<endl;
    cout<<"============================================"<<endl<<endl;

    cout<<"Masukkan kalimat yang akan dibalik: ";
    gets(kalimat);

    createStack(top);

    cout<<endl<<"Sebelum dibalik: ";
    for (int i=0; i<strlen(kalimat); i++){
        createElmt(s,kalimat[i]);
        push(top,s);
    }
    cout<<endl<<"Sesudah dibalik: ";
    for (int i=0; i<strlen(kalimat); i++){
        pop(top,pHapus);
    }
}
```

```
Program pembalik kalimat dengan List Berkait
============================================

Masukkan kalimat yang akan dibalik: aku kamu kita

Sebelum dibalik: aku kamu kita
Sesudah dibalik: atik umak uka

Terminated with return code 0
Press any key to continue ...
```

BAB III
PENUTUP

3.1. Kesimpulan

Stack pada Struktur Data adalah sekumpulan data yang seolah-olah diletakkan di atas data yang lain atau suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (top) saja. **Stack** bersifat LIFO (*Last In First Out*) artinya data yang terakhir masuk ke dalam stack akan menjadi yang pertama keluar dari stack.

Cara mendeklarasikannya ada dua cara yaitu menggunakan array dan list berkait. Untuk penginisialisasian stack harus dimulai dari -1 dan pengoperasiannya terdiri dari **push** untuk menambah item pada tumpukan stack paling atas, **pop** untuk mengambil item pada tumpukan stack paling atas, **empty** untuk mengecek apakah stack sudah kosong, **full** untuk mengecek apakah stack sudah penuh, dan **sizeof** digunakan untuk mengukur panjang stack.

3.2. Saran

Pembelajaran tentang stack ini masih harus lebih diperdalam lagi agar dapat diterapkan untuk contoh-contoh lain.

DAFTAR PUSTAKA

http://blog-arul.blogspot.co.id/2012/01/stack-pada-struktur-data.html

Contoh Makalah.docx