

LAPORAN PRAKTIKUM

MODUL 6

STACK



Disusun oleh :

Diva Octaviani

NIM : 2311102006

Dosen Pengampu:

Wahyu Andi Saputra, S. Pd., M. Eng.

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

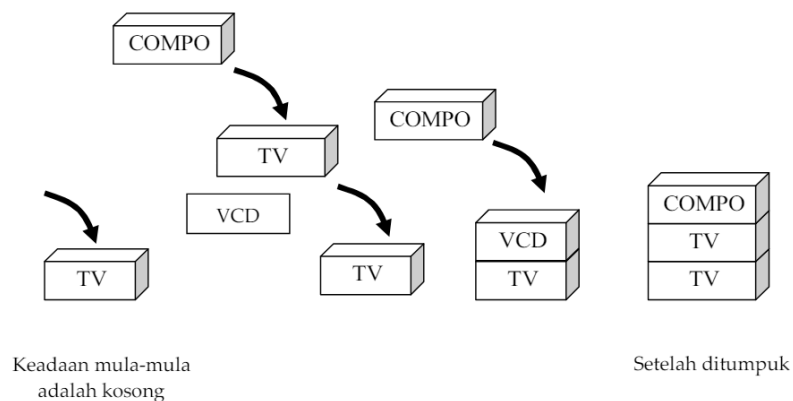
1. Mampu memahami konsep stack pada struktur data dan algoritma.
2. Mampu mengimplementasikan operasi-operasi pada stack.
3. Mampu memecahkan permasalahan dengan solusi stack.

BAB II

DASAR TEORI

1. Pengertian Stack

Stack merupakan sebuah kumpulan data yang diletakkan di atas data lainnya, seperti sebuah tumpukan. Dengan demikian, stack merupakan salah satu struktur data yang menerapkan prinsip LIFO (Last In First Out), dimana elemen yang terakhir disimpan dalam stack, menjadi elemen yang pertama diambil.



Gambar 1. Ilustrasi stack

Pada gambar di atas, jika ingin mengambil sesuatu dari tumpukan maka harus mengambil benda paling atas terlebih dahulu, yakni compo. Misalnya jika VCD langsung diambil, compo akan jatuh. Prinsip stack ini bisa diterapkan dalam pemrograman. Pada C++, ada dua cara penerapan prinsip stack yaitu dengan array dan linked list.

2. Operasi pada Stack

a) Create : untuk membuat sebuah stack kosong.

```
struct STACK {  
    int top;  
    float data[5];  
};  
float dta;  
struct STACK stackbaru;
```

b) IsEmpty : untuk memeriksa apakah stack kosong. Tanda bahwa sebuah stack kosong adalah Top bernilai kurang dari nol (-1).

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

- c) IsFull : untuk memeriksa apakah stack yang ada sudah penuh. Stack akan penuh jika puncak stack terletak tepat dibawah jumlah maksimum yang dapat ditampung stack (Top = MAX_STACK-1).

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

- d) Push : untuk menambahkan item pada tumpukan paling atas. Operasi ini tidak dapat dilakukan jika stack dalam keadaan penuh.

```
void push(float dta) {
    if (isfull()==false) {
        puts("stack penuh");
    } else {
        stackbaru.top++;
        stackbaru.data[top]=dta;
    }
}
```

- e) Pop : untuk mengambil item teratas dari stack dengan syarat stack tidak dalam kondisi kosong.

```
void pop() {
    if (isempty()==false) {
        cout<<"data kosong";
    } else {
        cout<<"data yang terambil :
"<<stackbaru.data[top]<<endl;
        stackbaru.top--;
    }
}
```

- f) Clear : untuk mengosongkan stack dengan cara mengeset Top dengan -1. Jika Top bernilai kurang dari nol maka stack dianggap kosong.

```
void clear () {  
    top=-1  
}
```

g) Retrieve : untuk melihat nilai yang berada pada posisi tumpukan teratas.

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

3. Pointer sebagai Penunjuk Stack

Selain menggunakan indeks, untuk menunjuk sebuah Top atau posisi teratas dari stack, dapat juga digunakan pointer sebagai berikut.

a) Membuat Stack dengan Pointer

```
int S[10], *Top, *BatasAtas  
Top = &S[-1];  
BatasAtas = &S[10];
```

b) Proses Push

```
if (Top < BatasAtas)  
    Top++;  
*Top = X;  
else  
    printf("Stack Penuh");
```

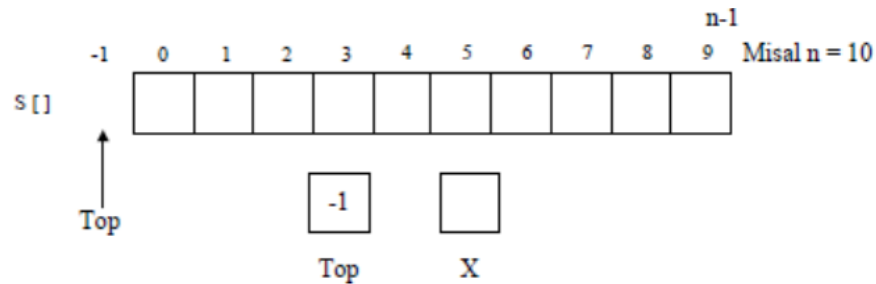
c) Proses Pop

```
if (Top > &A[-1])  
    X= *Top;  
    Top --;  
else  
    printf("Stack Kosong");
```

4. Representasi Proses Stack

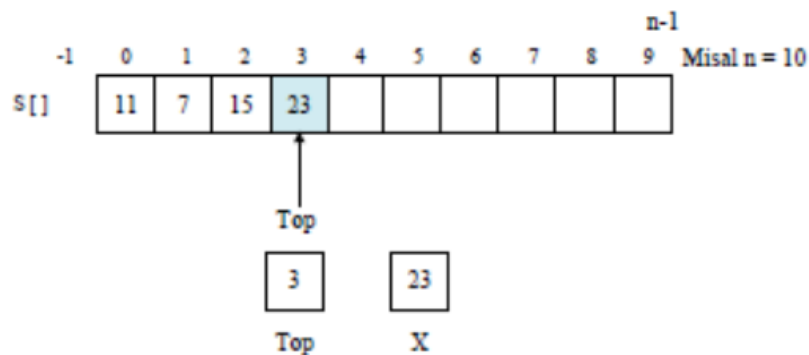
Stack adalah salah satu dari contoh struktur data yang terdiri dari satu collection, yang juga menerapkan prinsip LIFO. Bila stack tersebut

menggunakan array satu dimensi, maka stack tersebut dapat diilustrasikan sebagai berikut :



Gambar 2. Ilustrasi stack satu dimensi menggunakan indeks array

Pada gambar di atas, diilustrasikan ada sebuah indeks array yang masih kosong pada awal pembuatan stack dimana $n[10]$, variabel Top berada pada -1 yang menunjukkan indeks masih dalam keadaan kosong.



Gambar 3. Ilustrasi stack ketika sudah diisi data

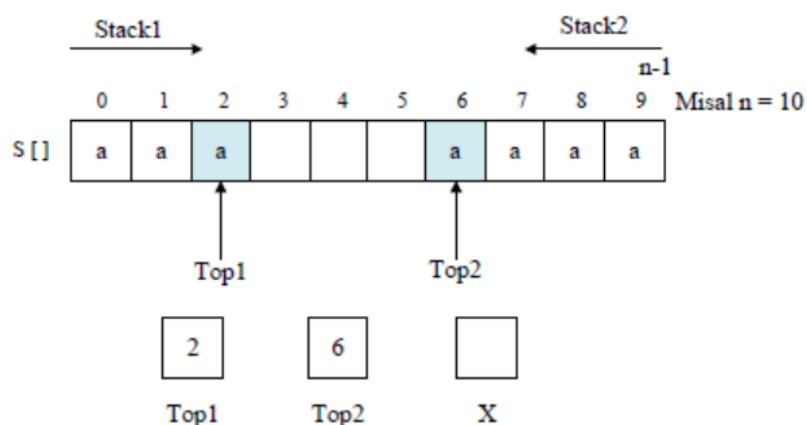
Gambar tersebut adalah keadaan ketika stack sudah diisi data. Pada kondisi ini data pertama yang diinputkan adalah $S[0]=11$, data kedua $S[1]=7$, data ketiga $S[2]=15$, data keempat $S[3]=23$. Kondisi Top sudah berubah menjadi data yang terakhir diinputkan (data keempat) sehingga $Top[3] X=23$.

Variabel Top digunakan sebagai indeks untuk menunjuk nomor elemen array yang berisi nilai stack yang berada paling kanan atau Top, yang ditunjukkan dengan angka 3. Variabel X bertipe integer digunakan sebagai perantara, dimana data yang akan disimpan kedalam stack harus berasal dari X. Demikian juga data yang baru diambil dari dalam stack harus diterima terlebih dahulu oleh variabel X, kemudian baru diberikan ke variabel lain untuk diolah.

Oleh karena itu, jika ada instruksi PUSH, maka data baru (yang diambil dari isi variabel X) akan disimpan dalam elemen S[4] sehingga indeks Top harus diarahkan ke posisi no.4. Artinya, Top maju terlebih dahulu satu langkah ke S[4], kemudian baru mengisi nilai pada S[4]. Sedangkan jika ada instruksi Pop, maka yang akan diambil adalah isi dari S[3] dan datanya akan disimpan terlebih dahulu dalam variabel X, kemudian indeks Top menjadi mundur satu langkah sehingga akan menunjuk S[2].

5. 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 First Out).



Gambar 4. Ilustrasi double stack

Gambar di atas adalah ilustrasi indeks array pada double stack. Pada stack 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].

6. Operasi Dasar Pada Double Stack

a) Inisialisasi

Proses awal adalah proses menyiapkan indeks penunjuk stack untuk pertama kali. Pada tahap ini, ditetapkan $Top1 = -1$ (sama seperti single stack) dan $Top2 = \text{banyak jumlah data}$.

```
void AWAL (void)
{
    Top1 = -1;
    Top2 = n;
}
```

b) Is Empty

Sama dengan single stack, operasi ini merupakan proses pengecekan stack dalam kondisi kosong.

```
if(top1 == -1) return true;
```

```
if(top2 == n) return true;
```

c) Is Full

Sama dengan single stack, operasi ini yaitu proses pengecekan stack dalam kondisi kosong.

```
int full(void){
    if(top1+1 >= top2){
        return true;
    }
}
```

d) Push (Stack1 dan Stack2)

Proses mengisi data pada stack1 maupun stack2.

```
void PUSH1 (void)
{
    Top1 = Top1 + 1;
    S[Top1] = X;
}
```

```
void PUSH2 (void)
{
    Top2 = Top2 - 1;
    S[Top2] = X;
}
```


e) Pop (Stack1 dan Stack2)

Proses mengambil data pada stack1 maupun stack2.

```
void POP1 (void)
{
X = S[Top1];
Top1 = Top1 - 1;
}
```

```
void POP2 (void)
{
X = S[Top2];
Top2 = Top2 + 1;
}
```

BAB III

GUIDED

1. Guided

Source code

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

string arrayBuku[5];
int maksimal = 5, top = 0;

bool isFull()
{
    return (top == maksimal);
}

bool isEmpty()
{
    return (top == 0);
}

void pushArrayBuku(string data)
{
    if (isFull())
    {
        cout << "Data telah penuh" << endl;
    }
    else
    {
        arrayBuku[top] = data;
        top++;
    }
}
```

```

}

void popArrayBuku()
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang dihapus" << endl;
    }
    else
    {
        arrayBuku[top - 1] = "";
        top--;
    }
}

void peekArrayBuku(int posisi)
{
    if (isEmpty())
    {
        cout << "Tidak ada data yang bisa dilihat" << endl;
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        cout << "Posisi ke " << posisi << " adalah " <<
arrayBuku[index] << endl;
    }
}

int countStack()

```

```
{
    return top;
}

void changeArrayBuku(int posisi, string data)
{
    if (posisi > top)
    {
        cout << "Posisi melebihi data yang ada" << endl;
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        arrayBuku[index] = data;
    }
}

void destroyArraybuku()
{
    for (int i = top; i >= 0; i--)
    {
        arrayBuku[i] = "";
    }
    top = 0;
}

void cetakArrayBuku()
{
    if (isEmpty())
    {
```

```

        cout << "Tidak ada data yang dicetak" << endl;
    }
    else
    {
        for (int i = top - 1; i >= 0; i--)
        {
            cout << arrayBuku[i] << endl;
        }
    }
}

int main()
{
    pushArrayBuku("Kalkulus");
    pushArrayBuku("Struktur Data");
    pushArrayBuku("Matematika Diskrit");
    pushArrayBuku("Dasar Multimedia");
    pushArrayBuku("Inggris");
    cetakArrayBuku();
    cout << "\n";
    cout << "Apakah data stack penuh? " << isFull() << endl;
    cout << "Apakah data stack kosong? " << isEmpty() << endl;
    peekArrayBuku(2);
    popArrayBuku();
    cout << "Banyaknya data = " << countStack() << endl;
    changeArrayBuku(2, "Bahasa Jerman");
    cetakArrayBuku();
    cout << "\n";
    destroyArraybuku();
    cout << "Jumlah data setelah dihapus: " << top << endl;
    cetakArrayBuku();
    return 0;
}

```

Screenshoot program

```
PS D:\Praktikum Struktur Data\Modul 6> cd "d:\Praktikum Strukt  
r Data\Modul 6\" ; if ($?) { g++ guided.cpp -o guided } ; if ($  
?) { .\guided }  
Inggris  
Dasar Multimedia  
Matematika Diskrit  
Struktur Data  
Kalkulus  
  
Apakah data stack penuh? 1  
Apakah data stack kosong? 0  
Posisi ke 2 adalah Dasar Multimedia  
Banyaknya data = 4  
Dasar Multimedia  
Bahasa Jerman  
Struktur Data  
Kalkulus  
  
Jumlah data setelah dihapus: 0  
Tidak ada data yang dicetak
```

Deskripsi program

Program di atas merupakan implementasi struktur data stack menggunakan array yang memiliki fungsi untuk menambahkan, menghapus, melihat, mengubah, dan menghitung buku dalam tumpukan. Array statis dengan jumlah maksimal 5 elemen digunakan untuk menyimpan daftar buku. Dalam fungsi main, program menerapkan operasi pada stack, seperti memasukkan buku ke stack, mencetak daftar buku, memeriksa kondisi stack penuh/kosong, melihat buku pada posisi tertentu, menghapus buku teratas, menghitung jumlah buku, mengubah judul buku, serta menghapus semua buku dari stack.

LATIHAN KELAS – UNGUIDED

Unguided 1

Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

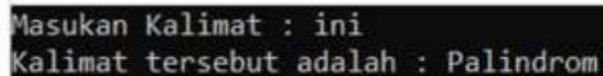
Contoh:

Kalimat : ini

Kalimat tersebut adalah polindrom

Kalimat : telkom

Kalimat tersebut adalah bukan polindrom



```
Masukan Kalimat : ini
Kalimat tersebut adalah : Palindrom
```

Source code

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

bool isPalindrome(string str)
{
    stack<char> charStack;
    int length = str.length();
    int i, mid = length / 2;

    for (i = 0; i < mid; i++)
    {
```

```

        charStack.push(str[i]);
    }

    for (i = mid + length % 2; i < length; i++)
    {
        if (charStack.top() != str[i])
        {
            return false;
        }
        charStack.pop();
    }

    return true;
}

int main()
{
    string kalimat;
    cout << "Masukkan Kalimat : ";
    getline(cin, kalimat);

    if (isPalindrome(kalimat))
    {
        cout << "Kalimat tersebut adalah : Palindrom";
    }
    else
    {
        cout << "Kalimat tersebut adalah : Bukan Palindrom";
    }

    return 0;
}

```


Screenshoot program

```
PS D:\Praktikum Struktur Data\Modul 6> cd "d:\Praktikum Struktur Data\Modul 6\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }

Masukkan Kalimat : ini
Kalimat tersebut adalah : Palindrom
PS D:\Praktikum Struktur Data\Modul 6> 
```

Deskripsi program

Program di atas bekerja dengan meminta pengguna memasukkan sebuah kalimat, kemudian menghilangkan semua karakter non-alfabet dan mengubahnya ke huruf kecil. Setelah itu, program menggunakan stack untuk membalikkan urutan karakter dan membandingkannya dengan urutan asli. Jika kedua urutan tersebut sama, maka kalimat tersebut adalah palindrom.


Unguided 2

Buatlah program untuk melakukan pembalikan terhadap kalimat menggunakan stack dengan minimal 3 kata. Jelaskan output program dan source codenya beserta operasi/fungsi yang dibuat?

Contoh

Kalimat : Telkom Purwokerto

Hasil : otrekowruP mokleT



Masukkan Kata Telkom Purwokerto
Datastack Array :
Data : otrekowruP mokleT

Source code

```
#include <iostream>
#include <stack>
#include <string>
#include <sstream>

using namespace std;

string reverseWords(const string &sentence)
{
    stringstream ss(sentence);
    string word, result;
    stack<char> charStack;

    while (ss >> word)
    {
        string reversedWord;

        for (char ch : word)
        {
```

```

        charStack.push(ch);
    }

    while (!charStack.empty())
    {
        reversedWord.push_back(charStack.top());
        charStack.pop();
    }

    if (!result.empty())
        result += " ";
    result += reversedWord;
}

return result;
}

int main()
{
    string input;
    cout << "Masukkan kalimat (minimal 3 kata) : ";
    getline(cin, input);

    stringstream ss(input);
    string word;
    int count = 0;
    while (ss >> word)
    {
        count++;
    }

    if (count < 3)
    {
        cout << "Kalimat harus terdiri dari minimal 3 kata.\n";
    }
}

```

```

    }
    else
    {
        string reversedSentence = reverseWords(input);
        cout << "Datastack Array : " << endl;
        cout << "Data : " << reversedSentence << endl;
    }

    return 0;
}

```

Screenshoot program

```

PS D:\Praktikum Struktur Data\Modul 6> cd "d:\Praktikum Struktur
Data\Modul 6\" ; if ($?) { g++ unguided2.cpp -o unguided2 } ; if
($?) { .\unguided2 }
Masukkan kalimat (minimal 3 kata) : Diva Octaviani 2311102006
Datastack Array :
Data : aviD inaivatc0 6002011132

```

Deskripsi program

Program di atas memiliki fungsi untuk membalik urutan huruf pada setiap kata dalam sebuah kalimat inputan pengguna menggunakan struktur data stack. Fungsi utama `reverseWords` menerima sebuah kalimat sebagai argumen dan mengembalikan kalimat baru dengan urutan huruf pada setiap kata yang telah dibalik. Dalam fungsi `main`, program meminta pengguna untuk memasukkan sebuah kalimat. Jika kalimat tersebut memiliki minimal 3 kata, program akan memanggil fungsi `reverseWords` dan menampilkan hasil pembalikan kata-kata tersebut. Jika kalimat kurang dari 3 kata, program akan menampilkan pesan error.

BAB IV

KESIMPULAN

Praktikum ini membahas implementasi struktur data stack, seperti konsep dasar stack, operasi-operasi pada stack, serta representasi proses stack. Pada bagian guided, disajikan contoh program sederhana untuk mengimplementasikan operasi-operasi dasar stack dalam menyimpan daftar buku menggunakan array statis. Selanjutnya, terdapat dua latihan unguided yang menerapkan konsep stack dalam menyelesaikan permasalahan nyata. Pertama, menentukan apakah sebuah kalimat merupakan palindrom atau bukan dengan memanfaatkan stack untuk membalikkan urutan karakter. Kedua, melakukan pembalikan urutan huruf pada setiap kata dalam kalimat inputan menggunakan stack. Keseluruhan praktikum ini bertujuan untuk memahami konsep stack dan mengimplementasikannya dalam memecahkan permasalahan sederhana terkait manipulasi data menggunakan prinsip LIFO (Last In First Out).

DAFTAR PUSTAKA

Asisten Praktikum, “*Modul 6 Stack*”, Learning Management System, 2024.

Asisten Praktikum, “*Praktikum Algoritma dan Struktur Data*”, Universitas Muhammadiyah Malang, 2016.

Fachrurrozi, M. dan Novi Yusliani, “*Modul Praktikum Struktur Data*”, Universitas Sriwijaya, 2006.