

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

MODUL VI



DISUSUN OLEH :

Nama: Fahrur Rizqi

Nim : 2311102059

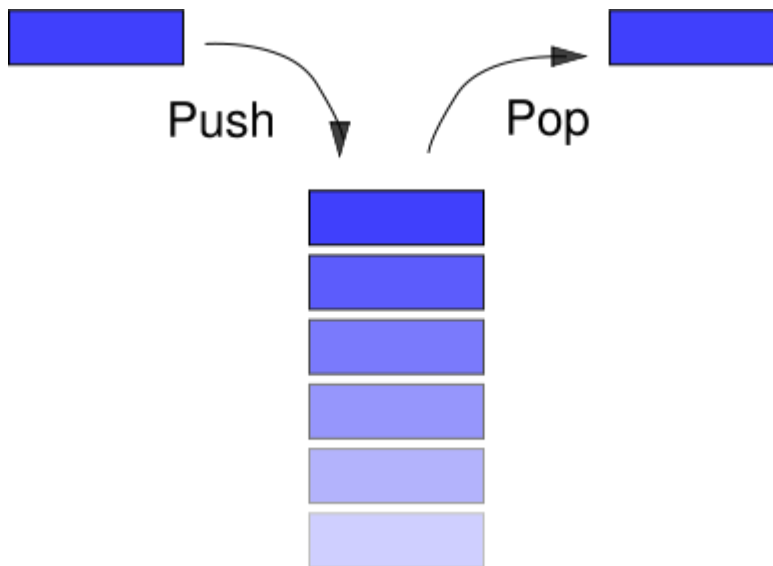
Dosen

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

Stack adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO (Last In First Out). 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, maka dilakukan operasi push. Dan untuk memindahkan dari tempat yang atas tersebut, maka dilakukan operasi pop.



1) Pada awalnya dibuat sebuah struct 'struct STACK', kemudian struct stack dijadikan menjadi sebuah tipe data dari variabel stackbaru. Dan beberapa prototype fungsi diantaranya :

- void clear()
- void print()
- void pop()
- void push(float dta)
- bool isempty()
- bool isfull()

2) Masuk ke fungsi main, deklarasikan variabel. Kemudian masuk ke perulangan do-while dengan kondisi ketika ulang == 'y' || ulang 'Y'. Pada perulangan ini akan ditampilkan beberapa menu pilihan yang dapat dipilih oleh user :

```
printf("Menu : ");  
  
puts("\n1. Pop stack");  
  
puts("2. Push stack");  
  
puts("3. Cetak");
```

```
puts("4. Bersihkan stack");
```

```
puts("5. Exit");
```

/*note = puts(put string), kegunaannya sama dengan printf ataupun cout, hanya saja yang dicetak hanyalah berupa STRING.*/

3) Jika input user== '1' maka program akan menjalankan seluruh pernyataan yang ada pada kondisi pertama, pop(); //panggil fungsi pop()

Fungsi POP :

```
void pop()
{
    if(isempty() == true) //panggil fungsi isempty(), jika kondisi benar pernyataan
    dijalankan
    {
        cout<<"Data telah kosong!";
    }
    else
    {
        cout<<"Data yang diambil : " <<stackbaru.data[stackbaru.top]<<endl;
        stackbaru.top--;
    }
}
```

Pertama kali yang akan dilakukan program adalah mengecek apakah stack dalam keadaan kosong, jika benar maka akan tercetak string pada layar "Data telah kosong!". Jika bernilai false maka data pada posisi teratas akan diambil, dan kemudian nilai stackbaru.top didecrement sehingga posisi teratas pada stack berganti dengan data dibawah top sebelumnya.

```
ulang = 'y';
```

Setelah fungsi pop() dijalankan, selanjutnya variabel ulang disetel dengan Nilai 'y' Sehingga setelah memilih menu pop() program akan secara otomatis mengulangi prosesnya.

```
getch(); //akan meminta input dari keyboard sehingga akan menghentikan
```

program sementara sampai adanya input keyboard.

4) Jika input user == '2', maka pernyataan yang berada di dalam kondisi ke dua akan dijalankan. Saat masuk dalam pernyataan pada kondisi kedua maka akan tercetak String pada layar dan kemudian user diminta untuk mengisi data yang akan ditambahkan dalam tumpukan stack.

```
cout<<"\nTambah Data";

        cout<<"\n----- ";

        cout<<"\nData yang akan disimpan di stack : ";

        cin>>dta;
```

Data yang telah diinputkan disimpan dalam memori, dan digunakan untuk pemanggilan fungsi push(data).

```
push(dta); /*panggil fungsi push(dta)--dta sesuai dengan data ynag diinput*/
```

Fungsi PUSH :

```
void push(float dta)

{

    if(isfull() == true) /*panggil fungsi isempty(), jika kondisi benar pernyataan
dijalankan*/

    {

        puts("Maaf, stack penuh");

    }

    else{

        stackbaru.top++;

        stackbaru.data[stackbaru.top]=dta;

    }

}
```

Pertama kali akan dicek apakah stack dalam keadaan penuh, jika true maka akan tercetak string pada layar "Maaf, stack penuh". Jika bernilai salah maka stackbaru.top akan diincrement kemudian data yang tadi diinputkan ditambahkan pada stack.

```
ulang = 'y';
```

Setelah fungsi push() dijalankan, selanjutnya variabel ulang disetel dengan Nilai 'y' Sehingga setelah memilih menu push() program akan secara otomatis mengulangi prosesnya.

5) Jika input user=='3', maka program akan menjalankan seluruh pernyataan yang berada di dalam kondisi ketiga.

```
print(); /*panggil fungsi untuk mencetak data dalam stack*/
```

Fungsi PRINT :

```
printf("\nData yang terdapat dalam stack : \n");  
printf("-----\n");  
for(int i=0; i<=stackbaru.top; i++)  
{  
    cout<<stackbaru.data[i]<<"  ";  
}
```

Dengan memanfaatkan perulangan for, fungsi ini akan mencetak seluruh data yang berada di dalam stack.

```
cout<<"\n\nUlang ? (y/t)";  
cin>>ulang;
```

Setelah data tercetak pada layar, selanjutnya akan ditampilkan string "Ulang ?(y/n)". Jika input user adalah 'y' || 'Y' maka program akan dijalankan lagi mulai awal (diulang), tapi jika input user == 'n' || 'N' maka akan keluar dari perulangan dan selanjutnya keluar dari program.

6) Jika Input user == '4', maka seluruh pernyataan yang berada dalam kondisi keempat akan dijalankan.

```
clear(); //panggil fungsi untuk membersihkan stack
```

Fungsi CLEAR :

```
void clear()  
{  
    stackbaru.top = -1;  
    printf("\nSekarang stack kosong");  
}
```

Saat fungsi ini dipanggil maka posisi stackbaru.top diinisialisasi berada pada -1. Seperti halnya mereset ulang suatu stack yang membuat isinya akan hilang.

```
cout<<"\n\nUlang ? (y/t)";
```

```
cin>>ulang;
```

Setelah data dalam stack dibersihkan, selanjutnya akan ditampilkan string “Ulang?(y/n)”. Jika input user adalah ‘y’ || ‘Y’ maka program akan dijalankan lagi mulai awal (diulang), tapi jika input user == ‘n’ || ‘N’ maka akan keluar dari perulangan dan selanjutnya keluar dari program.

7) Jika input user == ‘5’ maka pernyataan yang berada pada kondisi kelima akan dijalankan

```
exit(0);
```

Pernyataan diatas digunakan untuk keluar dari program. Jadi jika kita memilih menu 5 maka kita akan keluar dari program.

Penjelasan fungsi boolean isfull() dan bool isempty()

//fungsi boolean untuk mengetahui apakah stack penuh

```
bool isfull()
```

```
{
```

```
    if(stackbaru.top == maxstack)
```

```
        return true;
```

```
    else
```

```
        return false;
```

```
}
```

Untuk mengetahui apakah suatu stack sedang penuh adalah dengan membandingkan stackbaru.top dengan maxstack, jika kondisi benar maka stack dalam posisi penuh, dan sebaliknya.

//fungsi boolean untuk mengetahui apakah stack kosong

```
bool isempty()
```

```
{
```

```
    if(stackbaru.top == -1)
```

```
        return true;
```

```
    else
```

```
        return false;
```

```
}
```

Untuk mengetahui apakah suatu stack dalam keadaan kosong adalah dengan membandingkan `stackbaru.top` dengan `-1`, jika kondisi benar maka stack dalam posisi kosong, dan sebaliknya.

B. Guided

Guided 1

Sourcode :

```
#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;
}

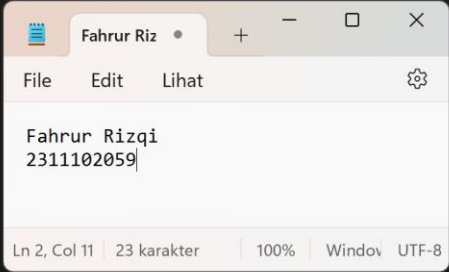
```

Output :

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
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
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6>
```



Deskripsi :

Program ini merupakan implementasi sederhana dari struktur data stack menggunakan array. Dalam program tersebut, terdapat fungsi-fungsi dasar untuk operasi pada stack seperti menambahkan data (push), menghapus data (pop), melihat data pada posisi tertentu (peek), dan menghitung jumlah data dalam stack. Program juga menyediakan fungsi-fungsi untuk memeriksa apakah stack sudah penuh atau kosong, serta untuk mengubah atau menghapus data dalam stack. Selain itu, terdapat fungsi main() yang melakukan serangkaian operasi stack seperti menambah, menghapus, dan mencetak data dalam stack. Kode ini memberikan penggunaan dasar dari struktur data stack menggunakan array dalam bahasa C++.

C. Unguided

Unguided 1

Sourcode

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

using namespace std;

string removeNonAlphanumeric(string str) {
    string result = "";
    for (char c : str) {
        if (isalnum(c)) {
            result += tolower(c);
        }
    }
}
```

```

    }
    return result;
}

bool isPalindrome(string str) {
    stack<char> charStack;
    int length = str.length();

    for (int i = 0; i < length / 2; i++) {
        charStack.push(str[i]);
    }

    for (int i = (length + 1) / 2; i < length; i++) {
        if (charStack.top() != str[i]) {
            return false;
        }
        charStack.pop();
    }

    return true;
}

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

    string cleanedInput = removeNonAlphanumeric(input);

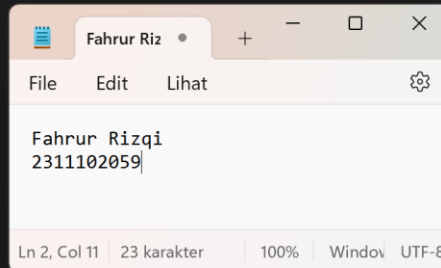
    if (isPalindrome(cleanedInput)) {
        cout << "Kalimat tersebut adalah palindrom." << endl;
    } else {
        cout << "Kalimat tersebut bukan palindrom." << endl;
    }

    return 0;
}

```

Output:

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6\" ; if ($?) { gcc tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }  
Masukkan kalimat: saya makan nasi  
Kalimat tersebut bukan palindrom.  
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6> 
```



Deskripsi :

Program ini merupakan sebuah alat sederhana untuk mengecek apakah sebuah kalimat adalah palindrom. Dengan menerima input kalimat dari pengguna, program membersihkan kalimat tersebut dari karakter-karakter non-alfabet dan mengubahnya menjadi huruf kecil. Kemudian, ia membandingkan setiap karakter dari kalimat dengan karakter-karakter dalam stack, dimulai dari belakang. Jika tidak ada perbedaan, program menyatakan bahwa kalimat tersebut adalah palindrom; jika tidak, kalimat tersebut bukan palindrom. Selanjutnya, hasil pengecekan ditampilkan kepada pengguna.

Unguided 2

Sourcode

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

    struct tumpukan
    {
        string isi[192]; int atas;
    };

    void push(tumpukan &T, const string &x)
    {
        if (T.atas == 192)
        {
            cout << "Tumpukan Sudah Penuh"; return;
        }

        T.atas++; T.isi[T.atas] = x;
    }

    string pop(tumpukan &T)
    {
        if (T.atas == 0)
        {
            cout << "Tumpukan sudah kosong";
            return "";
        }

        string hasil = T.isi[T.atas]; T.atas--;
        return hasil;
    }
```

```
int main()
{
    string kalimat; tumpukan T; T.atas = 0;

    cout << "Masukkan kata: "; getline(cin, kalimat);

    cout << "\nData asli: " << kalimat;

    for (char c : kalimat)
    {
        push(T, string(1, c));
    }

    cout << "\nData: "; while (T.atas > 0)
    {
        cout << pop(T);
    }

    cout << "\n\n";

    return 0;
}
```

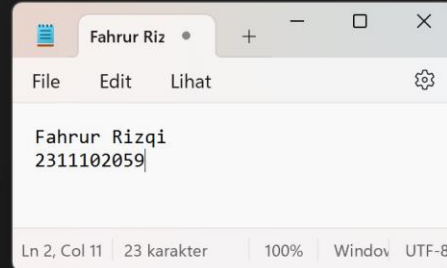
Output:

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }  
Masukkan kata: saya makan nasi
```

Data asli: saya makan nasi

Data: isan nakam ayas

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 6> █
```



Deskripsi :

Program ini merupakan program yang membalikkan urutan kata-kata dalam sebuah kalimat. Pengguna diminta memasukkan kalimat, kemudian program memeriksa apakah kalimat tersebut memiliki minimal 3 kata. Jika iya, program membalikkan urutan kata-kata dan menampilkannya; jika tidak, tampilkan pesan error. Proses pembalikan urutan kata-kata dilakukan dengan menggunakan stack untuk menyimpan karakter-karakter dari kalimat, yang kemudian diambil kembali dari stack untuk membentuk kalimat yang dibalik.

Kesimpulan

Stack, atau tumpukan, adalah struktur data yang menggunakan prinsip Last-In-First-Out (LIFO), di mana elemen terakhir yang dimasukkan adalah yang pertama kali diambil. Dalam pemrograman, stack digunakan untuk menyimpan dan mengelola data dengan operasi dasar seperti push (menambahkan elemen), pop (menghapus elemen teratas), dan peek (melihat elemen teratas tanpa menghapusnya). Stack sering digunakan dalam parsing ekspresi matematika, penanganan rekursi, evaluasi ekspresi postfix, dan pengelolaan memori. Keunggulan stack adalah kecepatan operasi push dan pop yang konstan serta kemudahan implementasinya dalam berbagai konteks pemrograman.

Referensi

Unknown. (2014). Stack pada C++.
Diakses pada tanggal 19 mei 2024
[Stack pada C++ - nblognlife](#)