

**LAPORAN PRAKTIKUM STRUKTUR DATA
DAN ALGORITMA MODUL 6
“Stack”**



DISUSUN OLEH:
Farrell Edric Kelvianto
2311102079
S1 IF-11-B

DOSEN:

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

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

A. Dasar Teori

a. Pengertian Stack

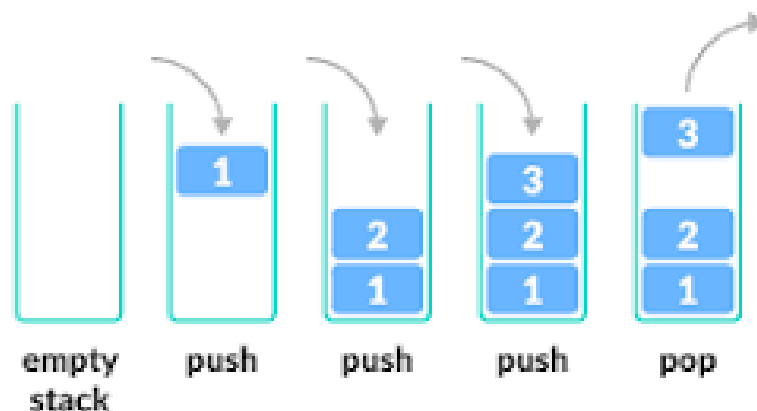
Stack adalah tumpukan dari suatu data, gambarannya itu seperti kita menaruh bola kedalam keranjang yang menjadikan tumpukan dalam suatu benda tersebut. Stack ini sifatnya yaitu FILO (First In Last Out) gambarannya seperti ketika kita memasukan bola dipaling terakhir atau kalau dalam Stack ini memiliki sebuah batas kapasitas dalam suatu data dan sama halnya seperti bucket/keranjang ini juga memiliki kapasitasnya masing – masing jadi ketika di masukan awal bolanya kedalam keranjang ini disebut dengan first in sampai ditumpukan akhir atau yang batas dari kapasitas keranjang tersebut lalu itulah yang dinamakan Last Out karena yang diambil adalah bola paling terakhir untuk mencapai dasar atau mengambil bola paling awal masuk.

b. Operasi Basic Stack

Dalam operasi stack ini berbeda dengan teknik hash table seperti kemarin disini ada beberapa operasi khusus untuk stack sesuai

- a. Push (masukan): berfungsi untuk menambahkan elemen kedalam tumpukan pada posisi paling atau ujung.
- b. Pop (Keluarkan): Mengapus elemen paling atas atau ujung dari tumpukan.
- c. IsEmpty (Kosong): memeriksa apakah tumpukan ini kosong atau tidak.
- d. Top (Atas): Berfungsi untuk melihat tumpukan teratas tanpa menghapusnya.
- e. IsFull (Penuh): Memeriksa apakah tumpukan ini terdeteksi penuh atau tidak.
- f. Size (Ukuran) : Berfungsi untuk mengembalikan jumlah elemen yang ada pada tumpukan elemen.
- g. Peek (Lihat) : Berfungsi untuk melihat nilai atau element pada posisi yang tertentu.
- h. Clear (Hapus Semua) : Mengosongkan atau menghapus semua elemen dari tumpukan.
- i. Search (Cari): Berfungsi untuk mencari keberadaan elemen yang tertentu

- Sebagai berikut adalah gambaran dari suatu stack yang berisikan sebuah tumpukan:



Guided

Guided 1

```
#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");
    pushArrayBuku ("Bahasa Asing");
    cetakArrayBuku();
    cout << "\n";
}
}

```

Screenshot Output

Deskripsi Program

Cara kerja program ini adalah seperti sebelum – sebelumnya hanya saja di program ini menggunakan operasi yaitu:

- isFull() ini berfungsi untuk mengindikasikan bahwa jika tumpukan penuh maka akan return.
- isEmpty() berfungsi untuk mengindikasikan jika tumpukan kosong maka program akan return.
- pushArrayBuku() berfungsi untuk menambahkan elemen pada tumpukan atau stack.
- popArrayBuku () berfungsi untuk menghapus elemen paling atas.
- peekArrayBuku () berfungsi untuk mengecek data dengan posisi tertentu.
- ChangeArrayBuku () berfungsi untuk mengubah data melalui posisi tertentu.
- DestroyArrayBuku () berfungsi untuk menghapus semua buku menjadikannya kosong kembali.
- CetakArrayBuku () berfungsi menampilkan dari isi stack tersebut dan mengubahnya menjadi output.

Untuk membentuk program stack ini memiliki susunan yang mirip di guided ini cara kerja program diatas adalah user diminta untuk menambahkan setiap tumpukannya lalu di dimasukan kedalam array dan menggunakan beberapa operasi basic seperti diatas.

B. Tugas Unguided 1

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

using namespace std;

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

// Mengecek apakah stack penuh
bool isFull()
{
    return (top == maksimal);
}

// Mengecek apakah stack kosong
bool isEmpty()
{
```

```

        return (top == 0);
    }

    // Menambahkan elemen ke stack
    void pushPalindrom(string Palindrom)
    {
        if (isFull())
        {
            cout << "Data Telah Penuh" << endl;
        }
        else
        {
            arrayPalindrom[top] = Palindrom;
            top++;
        }
    }

    // Menghapus elemen dari stack
    void popPalindrom()
    {
        if (isEmpty())
        {
            cout << "Tidak ada data yang dihapus" << endl;
        }
        else
        {
            arrayPalindrom[top - 1] = "";
            top--;
        }
    }

    // Melihat elemen pada posisi tertentu dari stack
    void peekPalindrom(int posisi)
    {
        if (isEmpty())
        {
            cout << "Suku Kata Palindrom Tidak Ditemukan" << endl;
        }
        else
        {
            int index = top;
            for (int i = 1; i <= posisi; i++)
            {
                index--;
            }
        }
    }

```

```

        cout << "Posisi ke " << posisi << " adalah " << arrayPalindrom[index]
<< endl;
    }
}

// Menghitung jumlah elemen dalam stack
int countstack()
{
    return top;
}

// Mengubah elemen pada posisi tertentu dari stack
void changePalindrome(int posisi, string palindrome)
{
    if (posisi > top)
    {
        cout << "Posisi melebihi jangkauan " << endl;
    }
    else
    {
        int index = top;
        for (int i = 1; i <= posisi; i++)
        {
            index--;
        }
        arrayPalindrom[index] = palindrome;
    }
}

// Menghapus semua elemen dalam stack
void destroyArraybuku()
{
    for (int i = top; i >= 0; i--)
    {
        arrayPalindrom[i] = "";
    }
    top = 0;
}

bool isPalindrome(const string& word)
{
    int left = 0;
    int right = word.length() - 1;

```



```

    while (left < right)
    {
        if (word[left] != word[right])
        {
            return false;
        }
        left++;
        right--;
    }
    return true;
}

bool Palindrome(const string& sentence)
{
    string pembersihKata;

    for (char c : sentence)
    {
        if (isalpha(c))
        {
            pembersihKata += tolower(c);
        }
    }

    return isPalindrome(pembersihKata);
}

int main()
{
    string kalimat1 = "Killin of demand";
    string kalimat2 = "ini";

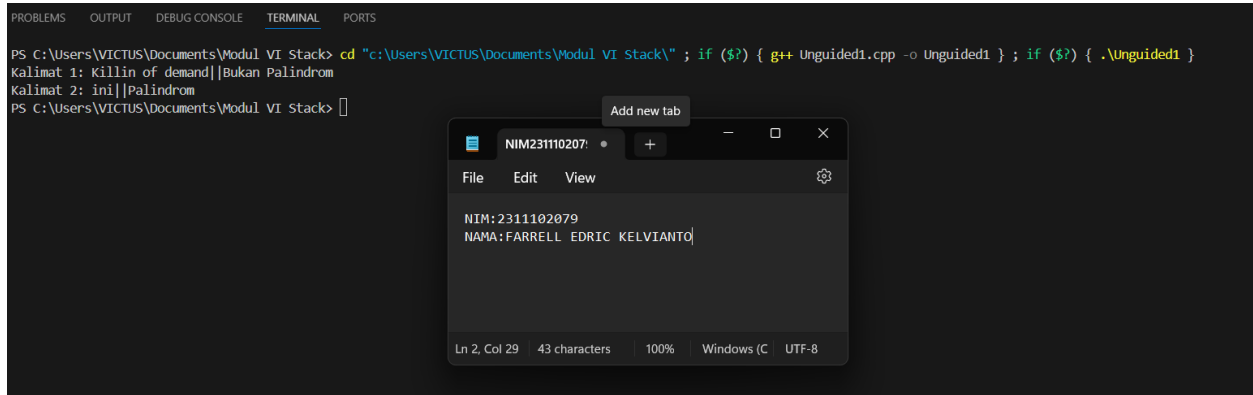
    cout << "Kalimat 1: " << kalimat1 << "||" << (Palindrome(kalimat1) ?
"Palindrom" : "Bukan Palindrom") << endl;
    cout << "Kalimat 2: " << kalimat2 << "||" << (Palindrome(kalimat2) ?
"Palindrom" : "Bukan Palindrom") << endl;

    return 0;
}

```

Screenshot Output

Soal No. 1 menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\VICTUS\Documents\Modul VI Stack> cd "c:\Users\VICTUS\Documents\Modul VI Stack\" ; if ($?) { g++ Unguided1.cpp -o Unguided1 } ; if ($?) { .\Unguided1 }
Kalimat 1: Killin of demand|Bukan Palindrom
Kalimat 2: ini|Palindrom
PS C:\Users\VICTUS\Documents\Modul VI Stack>
```

NIM2311102079

NAMA: FARRELL EDRIK KELVIAN TO

Ln 2, Col 29 43 characters 100% Windows (C UTF-8

Deskripsi Program

Cara kerja program ini adalah seperti program pada umumnya sebenarnya hanya disini memaksakan untuk melakukan implementasi stack dari beberapa proses stacknya disini yang menjadi stacknya itu adalah tumpukan katanya. Dan disini saya akan jelaskan bagaimana setiap operasi di atas:

- `isFull()` berfungsi untuk memberikan batas maksimal dan jika ujung sudah maksimal maka akan berhenti.
- `isEmpty()` ini berfungsi untuk mengindikasikan apakah stack masih kosong dan jika kosong maka dia akan mereturn.
- `pushPalindrom()` untuk menumpuk kata – kata yang diindikasikan sebagai kata palindrom atau bukan.
- `popPalindrom()` ini berfungsi untuk menghapus data paling atas.
- `peekPalindrom()` berfungsi untuk mencari kata – kata yang ditumpuk tadi dengan posisi tertentu.
- `ChangePalindrom()` ini berfungsi untuk mengubah kata palindrom sebelumnya menjadi kata baru.
- `DestroyArrayBuku()` ini berfungsi untuk menghapus semua data didalam array.
- `IsPalindrom()` berfungsi untuk mengecek apakah mengandung palindrome atau bukan dengan cara melihat dari kekiri ke kanan.
- `Palindrome()` ini berfungsi untuk memeriksa dari karakter ke karakter dan dipaling bawah itu fungsinya untuk mengabaikan seperti huruf kapital simbol ataupun angka.

Unguided 2

```
#include <iostream>
#include <cstring>

using namespace std;

const int MAX = 100;

void pembalikKata(char* sentence) {
    char stack[MAX];
    int top = -1;
    int length = strlen(sentence);

    for (int i = 0; i < length; ++i) {
        if (top >= (MAX - 1)) {
            cout << "Stack Overflow\n";
            return;
        } else {
            stack[++top] = sentence[i];
        }
    }

    for (int i = 0; i < length; ++i) {
        if (top < 0) {
            cout << "Stack Underflow\n";
            return;
        } else {
            sentence[i] = stack[top--];
        }
    }
}

int main() {
    while (true){
        char kalimat[MAX];

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

        int word_count = 1;
        char temp[MAX];
        strcpy(temp, kalimat);
```

```

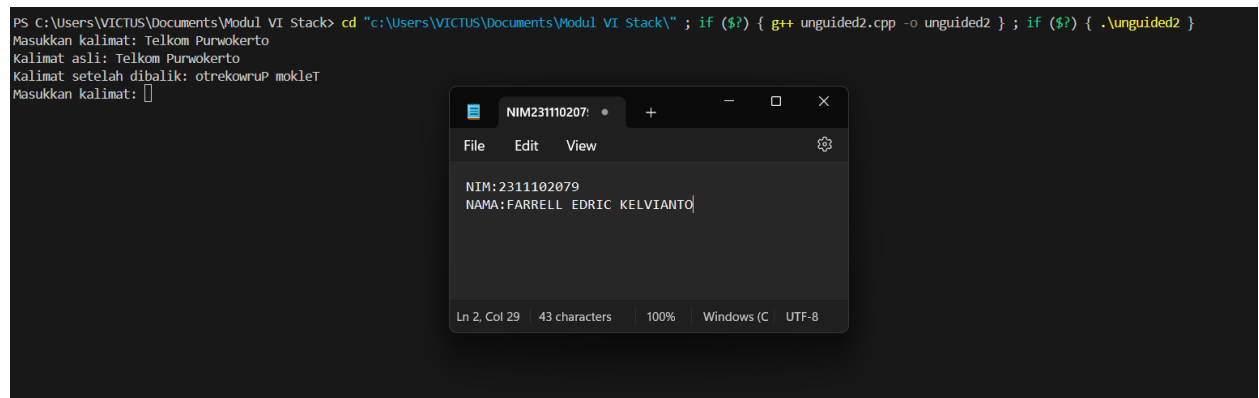
char* token = strtok(temp, " ");
while (token != nullptr) {
    ++word_count;
    token = strtok(nullptr, " ");
}

if (word_count <= 3 && word_count <= 2) {
    cerr << "Kalimat harus mengandung minimal 3 kata." << endl;
    return 1;
}

cout << "Kalimat asli: " << kalimat << endl;
pembalikKata(kalimat);
cout << "Kalimat setelah dibalik: " << kalimat << endl;
}
return 0;
}

```

Screenshot Output



Deskripsi Program

Nah untuk program yang ini berbeda dengan sebelumnya tetapi untuk implementasi stacknya tetap masuk. Disini terdapat sebuah 1 prosedur yaitu untuk implementasi stacknya dan beserta membaca length dari kata yang akan dibalik. Untuk membatasi maksnya disini menggunakan char Max yang bersikan 100 untuk batasnya disertai const supaya tidak terjadi manipulasi data. Disini yang sebagai pembalik katanya terdapat pada fungsi mainnya mnegggunakan sizeof untuk mengukur size kata dari inputan lalu mengindikasikan melalui char dan strtok ini berfungsi untuk membagi – bagi kata yang akan dibuat untuk reverse katanya, nah disini saya membatasi 3 kata seperti pada soalnya.

Kesimpulan

Kesimpulannya adalah stack merupakan tumpukan seperti dengan analogi bola yang ditumpukan kedalam ranjang bola dan pasti setiap ranjang memiliki kapasitas untuk diisi nah sama seperti stack jika stack telah mencapai maksimum maka dia tidak bisa diisi kembali ini berguna untuk mencegah alokasi yang tidak terkendali. Kesimpulan di nomor 1 dan 2 adalah walaupun temanya beda tetapi untuk susunan program untuk menggunakan stacknya tetap sama.

Refrensi

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