LAPORAN PRAKTIKUM

MODUL VI
STACK

Disusun oleh:
Syafanida Khakiki
NIM: 2311102005

Dosen Pengampu:
Muhammad Afrizal Amrustian, S. Kom., M. Kom

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I

TUJUAN PRAKTIKUM

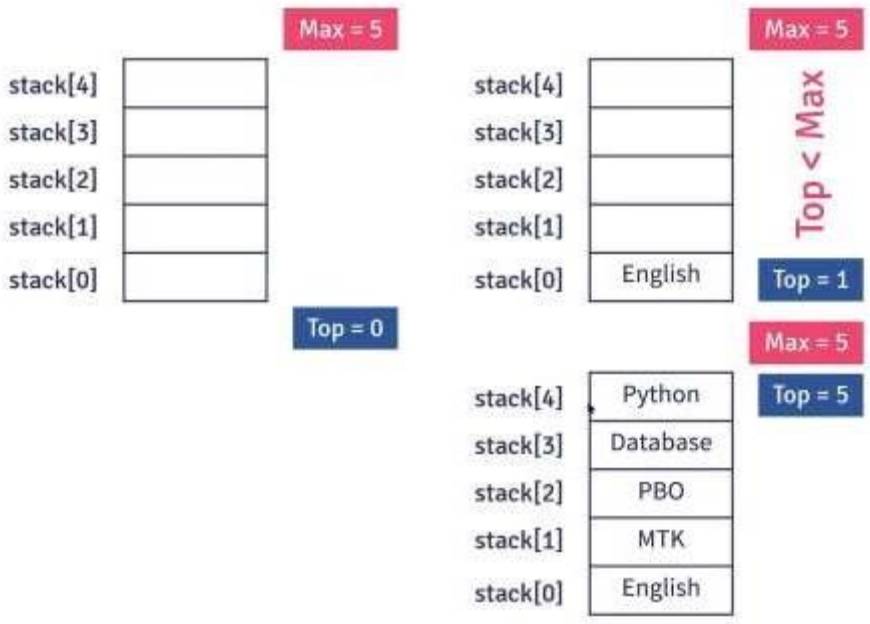
1. Mampu memahami konsep stack pada struktur data dan algoritma
2. Mampu mengimplementasikan operasi-operasi pada stack
3. Mampu memecahkan permasalahan dengan solusi stack

BAB II
DASAR TEORI

a. Pengertian Stack

Stack adalah struktur data sederhana yang digunakan untuk menyimpan data (mirip dengan Linked Lists). Dalam tumpukan, urutan kedatangan data penting. Sebuah tumpukan piring di kafetaria adalah contoh bagus dari tumpukan. Piring ditambahkan ke tumpukan saat mereka dibersihkan dan ditempatkan di bagian atas. Ketika sebuah piring dibutuhkan, diambil dari bagian atas tumpukan. Piring pertama yang ditempatkan di tumpukan adalah yang terakhir digunakan.

Definisi: Sebuah tumpukan adalah daftar terurut di mana penyisipan dan penghapusan dilakukan di satu ujung, disebut atas. Elemen terakhir yang dimasukkan adalah yang pertama dihapus. Oleh karena itu, disebut daftar Last in First out (LIFO).



Operasi pada stack melibatkan beberapa fungsi dasar yang dapat dilakukan padastruktur data ini. Berikut adalah beberapa operasi umum pada stack:

a. Push (Masukkan): Menambahkan elemen ke dalam tumpukan pada posisi paling atas atau ujung.
a. Pop (Keluarkan): Menghapus elemen dari posisi paling atas atau ujung tumpukan. Top (Atas): Mendapatkan nilai atau melihat elemen teratas pada tumpukan tanpa menghapusnya.
b. IsEmpty (Kosong): Memeriksa apakah tumpukan kosong atau tidak.

c. IsFull (Penuh): Memeriksa apakah tumpukan penuh atau tidak (terutama pada implementasi tumpukan dengan kapasitas terbatas).
d. Size (Ukuran): Mengembalikan jumlah elemen yang ada dalam tumpukan.
e. Peek (Lihat): Melihat nilai atau elemen pada posisi tertentu dalam tumpukan tanpa menghapusnya.
f. Clear (Hapus Semua): Mengosongkan atau menghapus semua elemen dari tumpukan.
g. Search (Cari): Mencari keberadaan elemen tertentu dalam tumpukan.

BAB III
GUIDED

1. **Guided 1**

   Source code :

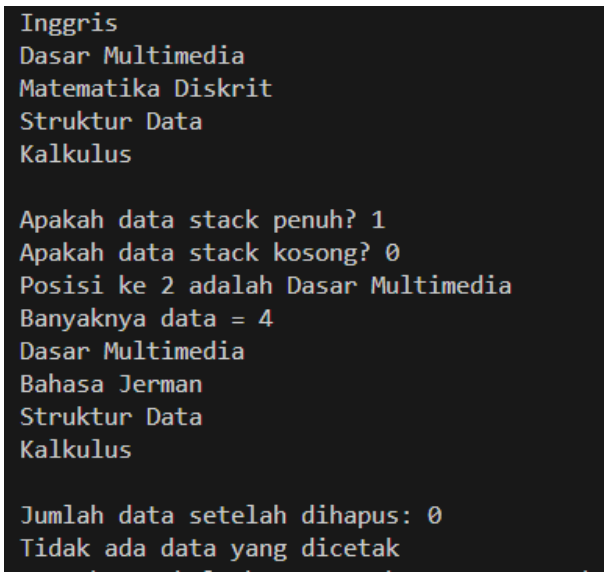
```
#include <iostream>
using namespace std;
string arrayBuku[5];
int maksimal = 5, top = 0;
bool isFull()
{
    return (top == maksimal);
}
bool isEmpty()
{
    return (top == 0);
}
void pushArrayBuku(string data)
{
    if (isFull())
    {
        cout << "Data telah penuh" << endl;
    }
    else
    {
        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;
}
```

}

Output :



Deskripsi program :

Program ini mengimplementasikan tumpukan (stack) untuk menyimpan data buku menggunakan array dalam bahasa C++. Stack adalah struktur data LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan adalah yang pertama kali dikeluarkan.

Komponen Program:

- **Variabel:**
  - `string arrayBuku[5]`: Array string untuk menyimpan data buku, dengan kapasitas maksimum 5 buku.
  - `int maksimal = 5`: Menetapkan nilai maksimum elemen dalam array.
  - `int top = 0`: Menginisialisasi variabel top untuk melacak posisi elemen teratas dalam stack.
- **Fungsi:**
  - `isFull()`: Memeriksa apakah stack penuh (top == maksimal).
  - `isEmpty()`: Memeriksa apakah stack kosong (top == 0).
  - `pushArrayBuku(string data)`: Menambahkan data buku baru ke dalam stack.
    - Memeriksa apakah stack penuh.
    - Jika tidak penuh, menambahkan data buku ke array di posisi `top` dan memperbarui `top`.
    - Jika penuh, menampilkan pesan bahwa stack penuh.
  - `popArrayBuku()`: Menghapus data buku teratas dari stack.
    - Memeriksa apakah stack kosong.
    - Jika tidak kosong, menghapus data buku di posisi `top-1`, memperbarui `top`, dan menampilkan pesan bahwa data telah dihapus.
    - Jika kosong, menampilkan pesan bahwa stack kosong.
  - `peekArrayBuku(int posisi)`: Menampilkan data buku pada posisi tertentu dalam stack.
    - Memeriksa apakah stack kosong.
    - Jika tidak kosong, menghitung indeks data berdasarkan posisi (top - posisi) dan menampilkan data buku pada indeks tersebut.

        - Jika kosong, menampilkan pesan bahwa stack kosong.
    - `countStack()`: Menghitung jumlah data buku dalam stack (nilai `top`).
    - `changeArrayBuku(int posisi, string data)`: Mengubah data buku pada posisi tertentu dalam stack.
        - Memeriksa apakah posisi melebihi jumlah data dalam stack.
        - Jika tidak, menghitung indeks data berdasarkan posisi (top - posisi) dan mengubah data buku pada indeks tersebut.
        - Jika posisi melebihi, menampilkan pesan bahwa posisi melebihi data yang ada.
    - `destroyArraybuku()`: Menghapus semua data buku dari stack.
        - Mengatur semua elemen array menjadi string kosong.
        - Mengatur `top` menjadi 0.
    - `cetakArrayBuku()`: Mencetak semua data buku dalam stack dari atas ke bawah.
        - Memeriksa apakah stack kosong.
        - Jika tidak kosong, mengiterasi array dari top-1 ke 0 dan menampilkan data buku pada setiap indeks.
        - Jika kosong, menampilkan pesan bahwa stack kosong.
- **Fungsi Utama (`main()`):**
    - Memasukkan 5 data buku ke dalam stack menggunakan `pushArrayBuku()`.
    - Mencetak semua data buku dalam stack menggunakan `cetakArrayBuku()`.
    - Memeriksa apakah stack penuh dan kosong menggunakan `isFull()` dan `isEmpty()`.
    - Menampilkan data buku pada posisi 2 menggunakan `peekArrayBuku()`.
    - Menghapus data buku teratas dari stack menggunakan `popArrayBuku()`.
    - Menghitung jumlah data buku dalam stack menggunakan `countStack()`.
    - Mengubah data buku pada posisi 2 menjadi "Bahasa Jerman" menggunakan `changeArrayBuku()`.
    - Mencetak semua data buku dalam stack menggunakan `cetakArrayBuku()`.
    - Menghapus semua data buku dari stack menggunakan `destroyArraybuku()`.
    - Mencetak jumlah data buku setelah dihapus dan stack kosong menggunakan `top` dan `cetakArrayBuku()`.

BAB IV
UNGUIDED

1. Unguided 1

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

Source code :

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;
bool isPalindrome(string kalimat)
{
    stack<char> stackKarakter;
    int length = kalimat.length();
    for (int i = 0; i < length / 2; i++)
    {
        stackKarakter.push(kalimat[i]);
    }
    int i = (length + 1) / 2;
    while (i < length)
    {
        if (kalimat[i] != stackKarakter.top())
        {
            return false; // Tidak palindrom
        }
        stackKarakter.pop();
        i++;
    }
    return true; // Palindrom
}
int main()
{
    string kalimat;
    cout << "Masukkan kalimat: ";
    getline(cin, kalimat);
    if (isPalindrome(kalimat))
    {
        cout << "Kalimat tersebut adalah palindrom." << endl;
    }
    else
```

```
    {
        cout << "Kalimat tersebut bukan palindrom." << endl;
    }
    return 0;
}
```

Output :

```
Masukkan kalimat: ibu ratna antar ubi
Kalimat tersebut adalah palindrom.
```

```
Masukkan kalimat: saya suka
Kalimat tersebut bukan palindrom.
```

Deskripsi Program :

Program ini mengimplementasikan fungsi untuk memeriksa apakah sebuah kalimat merupakan **palindrom** atau tidak menggunakan struktur data **stack** dalam bahasa C++. Palindrom adalah kalimat yang dibaca sama baik dari depan maupun belakang.

Komponen Program:

- **Fungsi `isPalindrome(string kalimat)`:**
  - **Parameter:** String `kalimat` yang ingin diperiksa.
  - **Deskripsi:**
    - Menginisialisasi stack karakter kosong (`stack<char> stackKarakter`).
    - Menentukan panjang kalimat (`length = kalimat.length()`).
    - Melakukan iterasi setengah dari panjang kalimat:
      - Memasukkan karakter kalimat pada posisi `i` ke dalam stack (`stackKarakter.push(kalimat[i])`).
    - Menentukan indeks awal untuk pemeriksaan karakter sisa (`i = (length + 1) / 2`).
    - Melakukan iterasi dari `i` hingga akhir kalimat:
      - Membandingkan karakter kalimat pada `i` dengan karakter teratas di stack (`kalimat[i] != stackKarakter.top()`).
        - Jika tidak sama, kembalikan `false` (kalimat bukan palindrom).
        - Jika sama, "pop" karakter teratas dari stack (`stackKarakter.pop()`).
    - Jika iterasi selesai tanpa ketidakcocokan, kembalikan `true` (kalimat palindrom).
- **Fungsi Utama (`main()`):**
  - Mendeklarasikan variabel string `kalimat`.

- Meminta pengguna untuk memasukkan kalimat melalui `cout << "Masukkan kalimat: ";` dan `getline(cin, kalimat);`.
- Memanggil fungsi `isPalindrome(kalimat)` untuk memeriksa kalimat.
- Menampilkan hasil pemeriksaan (palindrom atau bukan) menggunakan `cout`.

2. 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?

Source Code :

```cpp
#include <iostream>
#include <stack>
#include <string>
using namespace std;
string reverseLetters(string sentence)
{
    stack<char> letters;
    string reversedSentence = "";
    for (char letter : sentence)
    {
        letters.push(letter);
    }
    while (!letters.empty())
    {
        reversedSentence += letters.top();
        letters.pop();
    }
    return reversedSentence;
}
int main()
{
    string sentence;
    cout << "Masukkan kalimat: ";
    getline(cin, sentence);
    string reversedSentence = reverseLetters(sentence);
    cout << "\nHasil: " << reversedSentence << endl;
    return 0;
}
```

Output :

```
Masukkan kalimat: syafanida

Hasil: adinafays
```

Deskripsi Program :

Program ini mengimplementasikan fungsi untuk membalik urutan huruf dalam sebuah kalimat menggunakan struktur data "stack" dalam bahasa C++.

Fungsi `reverseLetters(string sentence)`:

- **Parameter:** Fungsi menerima string `sentence` sebagai input, yang mewakili kalimat yang ingin dibalik.
- **Nilai Kembali:** Fungsi mengembalikan string yang berisi kalimat yang dibalik.

Logika:

1. **Inisialisasi Stack:**
   - `stack<char> letters;`: Deklarasikan stack bernama `letters` dengan tipe `char` (untuk menyimpan karakter). Stack ini akan digunakan untuk menyimpan karakter kalimat sementara.
2. **Perulangan Kalimat dan Push Stack:**
   - `for (char letter : sentence)`: Loop ini mengiterasi setiap karakter (`letter`) dalam string `sentence`.
     - Di dalam loop:
       - `letters.push(letter);`: Mendorong karakter (`letter`) saat ini ke dalam stack `letters`. Ini secara efektif membangun stack di mana huruf pertama yang didorong berada di bawah dan huruf terakhir yang didorong berada di atas.
3. **Pop Stack dan Pembalikan Kalimat:**
   - `while (!letters.empty())`: Loop ini berlanjut selama stack `letters` tidak kosong (memiliki elemen).
     - Di dalam loop:
       - `reversedSentence += letters.top();`: Mengambil (menghapus) karakter teratas dari stack `letters` dan menambahkannya ke string `reversedSentence`. Karena karakter ditambahkan di akhir string dengan setiap iterasi, ini pada dasarnya membalik urutan karakter.
       - `letters.pop();`: Mengambil (menghapus) karakter teratas dari stack `letters`.
4. **Kembalikan Kalimat yang Dibalik:**
   - `return reversedSentence;`: Setelah loop menyelesaikan pemrosesan semua karakter, string `reversedSentence` akan berisi huruf dari kalimat input dalam urutan terbalik. Fungsi mengembalikan kalimat terbalik ini.

Fungsi Utama (`main()`):

1. **Input Kalimat:**
   - `string sentence;`: Deklarasikan variabel string bernama `sentence` untuk menyimpan kalimat yang disediakan pengguna.
   - `cout << "Masukkan kalimat: ";`: Meminta pengguna untuk memasukkan kalimat menggunakan cout.
   - `getline(cin, sentence);`: Membaca seluruh baris yang dimasukkan pengguna (termasuk spasi) dan menyimpannya dalam variabel `sentence` menggunakan getline.
2. **Pembalikan Kalimat:**
   - `string reversedSentence = reverseLetters(sentence);`: Memanggil fungsi `reverseLetters`, meneruskan `sentence` sebagai input. Fungsi mengembalikan kalimat yang dibalik, yang disimpan dalam variabel `reversedSentence`.
3. **Output:**
   - `cout << "\nHasil: " << reversedSentence << endl;`: Mencetak kalimat yang dibalik ke konsol menggunakan cout.

3.

BAB IV

KESIMPULAN

Stack atau tumpukan adalah struktur data linier yang mengikuti prinsip **Last In First Out (LIFO)**. Artinya, elemen yang terakhir dimasukkan (**push**) akan menjadi elemen pertama yang dikeluarkan (**pop**).

Karakteristik Stack

- **LIFO:** Elemen yang terakhir masuk akan menjadi elemen pertama yang keluar.
- **Akses:** Elemen hanya dapat diakses dari satu sisi, yaitu **puncak stack** (**top**).
- **Operasi Dasar:**
    - **Push:** Menambahkan elemen ke puncak stack.
    - **Pop:** Menghapus elemen dari puncak stack.
    - **IsEmpty:** Memeriksa apakah stack kosong.
    - **IsFull:** Memeriksa apakah stack sudah penuh.
    - **Peek:** Mendapatkan nilai elemen teratas tanpa menghapusnya.

Penggunaan Stack

Stack memiliki banyak aplikasi dalam pemrograman, antara lain:

- **Undo/Redo:** Menyimpan status program sebelumnya untuk memungkinkan undo/redo.
- **Evaluator Ekspresi:** Mengevaluasi ekspresi matematika dengan benar.
- **Parsing:** Memproses input teks atau kode secara berurutan.
- **Algoritma Pencocokan Pola:** Menemukan pola dalam teks atau data.
- **Mesin Virtual:** Menjalankan program dengan mensimulasikan prosesor.

Kelebihan Stack

- Sederhana dan mudah dipahami.
- Efisien untuk operasi akses, push, dan pop.
- Cocok untuk aplikasi yang membutuhkan urutan LIFO.

Kekurangan Stack

- Tidak efisien untuk operasi akses acak.
- Sulit untuk mengakses elemen di tengah stack.

DAFTAR PUSTAKA

Asisten Praktikum (2024). Stack. Learning Management System.