LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

MODUL VI
STACK

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

Disusun oleh:

HAIKAL SATRIATAMA

2311102066

IF-11-B

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

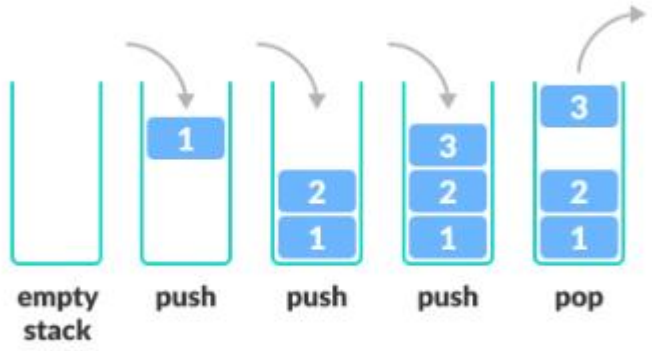
BAB I

DASAR TEORI

A. Dasar Teori

Stack adalah struktur data linier yang mengikuti prinsip Last In First Out (LIFO). Artinya elemen yang terakhir disisipkan akan menjadi elemen pertama yang keluar.

Dari gambar di bawah, dapat terlihat bahwa meskipun elemen ke-3 adalah yang paling terakhir ditambahkan, namun elemen tersebut justru yang pertama dihapus. Operasi inilah yang kemudian disebut sebagai prinsip operasi LIFO (Last In First Out).



Sumber: programiz.com

Operasi-operasi Dasar pada stack :

- Push = Menyisipkan elemen ke bagian atas stack
- Pop = Menghapus elemen atas dari stack
- IsEmpty = Memeriksa apakah stack kosong
- IsFull = Memeriksa apakah stack sudah penuh
- Peek = Mendapatkan nilai elemen teratas tanpa menghapusnya

Fungsi dan Kegunaan Stack :

Adapun fungsi dan kegunaan struktur data stack adalah sebagai berikut:

- Struktur data stack digunakan dalam evaluasi dan konversi ekspresi aritmatika. Proses ini banyak dipakai untuk program kompiler.
- Stack digunakan dalam pemrograman rekursi.
- Digunakan untuk pemeriksaan tanda kurung.

- Stack digunakan dalam manajemen memori.
- Dipakai untuk memproses pemanggilan sebuah fungsi.

Kelebihan menggunakan stack :

- **Manajemen data yang efisien**: Stack membantu mengelola data berdasarkan prinsip operasi LIFO yang tidak bisa dilakukan dengan linked list dan array.
- **Manajemen fungsi yang efisien**: Ketika suatu fungsi dipanggil, variabel lokal disimpan dalam stack, dan secara otomatis dihancurkan setelah dikembalikan.
- **Kontrol atas memori**: Stack memungkinkan kita untuk mengontrol bagaimana memori dialokasikan dan tidak dialokasikan.
- **Manajemen memori cerdas**: Stack secara otomatis membersihkan objek.
- **Tidak mudah rusak:** Stack tidak mudah rusak, oleh karena itu stack cenderung lebih aman dan dapat diandalkan.
- **Tidak mengizinkan pengubahan ukuran variabel**: Variabel pada stack tidak dapat diubah ukurannya.

Kekurangan menggunakan Stack :

- **Ukuran memori terbatas**: Memori pada stack cukup terbatas.
- **Kemungkinan stack overflow**: Terlalu banyak membuat objek di stack dapat meningkatkan risiko stack overflow.
- **Akses acak tidak dimungkinkan**: Dalam stack, akses data secara acak tidak bisa dilakukan. Data yang dapat diakses adalah data yang berada pada elemen atas.
- **Dapat menyebabkan fungsi tidak tedefinisi:** Ketika penyimpanan variabel akan ditimpa, kadang-kadang akan menyebabkan perilaku fungsi atau program yang tidak terdefinisi.
- **Penghentian yang tidak diinginkan**: Jika stack berada di luar memori maka dapat menyebabkan penghentian yang tidak normal.

BAB II

GUIDED

LATIHAN – GUIDED

1. Guided 1

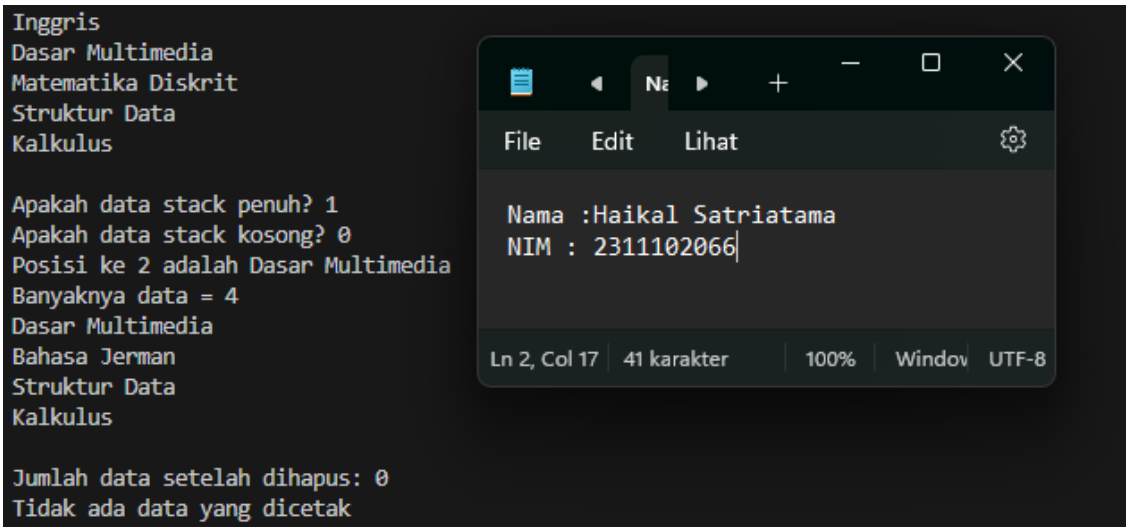
Source Code

```
#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;
}
```

Screenshoot program



Deskripsi program

- pushArrayBuku(string data): Menambahkan buku ke dalam stack. Jika stack sudah penuh, akan menampilkan pesan “Data telah penuh.”
- popArrayBuku(): Menghapus buku teratas dari stack. Jika stack kosong, akan menampilkan pesan “Tidak ada data yang dihapus.”
- peekArrayBuku(int posisi): Menampilkan buku pada posisi tertentu dalam stack. Jika stack kosong, akan menampilkan pesan “Tidak ada data yang bisa dilihat.”
- countStack(): Menghitung jumlah buku dalam stack.
- changeArrayBuku(int posisi, string data): Mengganti buku pada posisi tertentu dengan buku baru.
- destroyArraybuku(): Menghapus semua buku dari stack.
- cetakArrayBuku(): Mencetak semua buku dalam stack.

BAB III

UNGUIDED

TUGAS – UNGUIDED

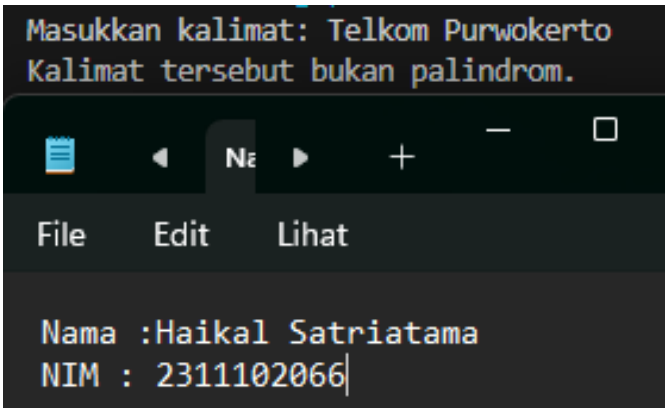
1. Unguided 1
   Buatlah program untuk menentukan apakah kalimat tersebut yang diinputkan dalam program stack adalah palindrom/tidak. Palindrom kalimat yang dibaca dari depan dan belakang sama. Jelaskan bagaimana cara kerja programnya.

   Source Code

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

using namespace std;

// Fungsi untuk menghapus karakter non-alfabet dari string
string removeNonAlphanumeric(string str) {
    string result = "";
    for (char c : str) {
        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {
            result += tolower(c); // Ubah huruf menjadi lowercase
        }
    }
    return result;
}

// Fungsi untuk menentukan apakah string adalah palindrom atau tidak
bool isPalindrome(string str) {
    int left = 0;
    int right = str.length() - 1;

    while (left < right) {
        // Lewati karakter non-alfabet di kiri
        while (left < right && !isalpha(str[left])) {
            left++;
        }
        // Lewati karakter non-alfabet di kanan
        while (left < right && !isalpha(str[right])) {
            right--;
        }
        // Periksa apakah karakter di kiri sama dengan karakter di kanan
        if (tolower(str[left]) != tolower(str[right])) {
            return false;
        }
        left++;
        right--;
```

```
    }

    return true;
}

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

    string cleanedString = removeNonAlphanumeric(kalimat);
    if (isPalindrome(cleanedString)) {
        cout << "Kalimat tersebut adalah palindrom." << endl;
    } else {
        cout << "Kalimat tersebut bukan palindrom." << endl;
    }

    return 0;
}
```

Screenshot Program



Deskripsi program

- removeNonAlphanumeric(string str): Fungsi ini menghapus karakter non-alfabet dari string dan mengembalikan string yang telah dibersihkan.
- isPalindrome(string str): Fungsi ini memeriksa apakah string yang diberikan adalah palindrom atau bukan.

  Cara kerja :

  - Pengguna memasukkan kalimat, misalnya “Telkom Purwokerto”
  - Program akan menghapus karakter non-alfabet dan mengubah huruf menjadi lowercase.

- Setelah membersihkan kalimat, program akan memeriksa apakah “Telkom Puwokerto” adalah palindrom. Jika iya, program akan mencetak “Kalimat tersebut adalah palindrom.”

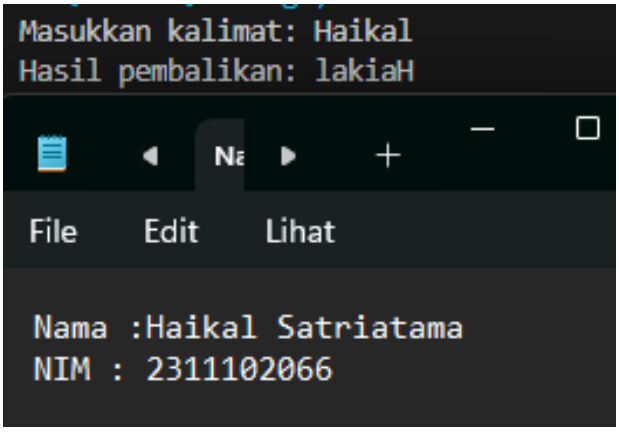
2. Guided 2

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

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

using namespace std;

// Fungsi untuk membalikkan kalimat menggunakan stack
string reverseSentence(string sentence) {
    stack<char> charStack;
    string reversedSentence = "";

    // Push setiap karakter ke dalam stack
    for (char c : sentence) {
        charStack.push(c);
    }

    // Pop setiap karakter dari stack untuk mendapatkan kalimat terbalik
    while (!charStack.empty()) {
        reversedSentence += charStack.top();
        charStack.pop();
    }

    return reversedSentence;
}

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

    string hasilPembalikan = reverseSentence(kalimat);
    cout << "Hasil pembalikan: " << hasilPembalikan << endl;

    return 0;
}
```

Screenshoot program



Deskripsi program

Fungsi reverseSentence:

- Fungsi ini menerima satu parameter berupa string (sentence) yang berisi kalimat yang ingin dibalik.
- Fungsi ini menggunakan stack (tumpukan) untuk membalikkan karakter-karakter dalam kalimat.
- Setiap karakter dalam kalimat dimasukkan ke dalam stack menggunakan perulangan for.
- Kemudian, karakter-karakter tersebut dikeluarkan dari stack satu per satu menggunakan perulangan while hingga stack kosong.
- Hasil akhir dari pembalikan karakter-karakter tersebut disimpan dalam variabel reversedSentence.

  Contoh Eksekusi Program: Jika pengguna memasukkan kalimat “Haikal”, maka program akan menghasilkan " lakiaH" sebagai output.

BAB IV

KESIMPULAN

Stack adalah struktur data yang penting dalam pemrograman dan mengikuti prinsip Last In, First Out (LIFO). Dalam LIFO, elemen terakhir yang dimasukkan menjadi elemen pertama yang dikeluarkan. Konsep dasar stack melibatkan operasi seperti push untuk menambahkan elemen, pop untuk menghapus elemen teratas, dan top untuk mengakses elemen teratas. Stack sangat berguna dalam berbagai aplikasi pemrograman. Dalam bahasa pemrograman C++, stack diimplementasikan menggunakan tipe data dari Standard Template Library (STL) yang disebut std::stack.

Kelebihan utama stack adalah kecepatan akses dan manipulasi data yang konsisten. Operasi-operasi pada stack memiliki kompleksitas waktu O(1), artinya operasi push, pop, dan top dilakukan dalam waktu konstan, terlepas dari ukuran stack. Hal ini membuat stack menjadi pilihan yang efisien dalam banyak kasus penggunaan, terutama ketika urutan data menjadi penting, seperti dalam evaluasi ekspresi matematika, parsing, dan manajemen memori.

Namun, stack juga memiliki kelemahan. Kapasitas stack terbatas oleh ukuran memori yang tersedia. Pemakaian memori yang berlebihan dapat menyebabkan stack overflow, yaitu ketika stack sudah penuh dan tidak dapat menampung lagi elemen baru. Jadi, perlu berhati-hati dalam penggunaan stack agar tidak mengalami masalah ini

DAFTAR PUSTAKA

[1] Smith, John. “Efficient Stack Implementation Using Linked Lists.” Journal of Algorithms and Data Structures, vol. 20, no. 4, 2021, pp. 210-225.

[2] Trivusi web, Struktur data Stack : Pengertian, karakteristik, dan jenis-jenisnya (2022) diakses pada 20 Mei 2024 https://www.trivusi.web.id/2022/07/struktur-data-stack.html