LAPORAN PRAKTIKUM

Modul 7

Stack

Disusun Oleh:

Yogi Hafidh Maulana - 2211104061

SE06-02

Dosen :

Wahyu Andi

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

A. Tujuan

- Memahami konsep dasar stack sebagai struktur data dengan prinsip LIFO (Last-In, First-Out).
- Mengimplementasikan operasi dasar stack, seperti push, pop, peek, dan display, baik dalam bentuk array maupun linked list.
- Menerapkan stack dalam pemecahan masalah sederhana, seperti menentukan palindrom dan melakukan pembalikan kata menggunakan stack.
- Mampu mengidentifikasi dan menangani kondisi overflow dan underflow dalam implementasi stack.
- Mampu menganalisis manfaat stack dalam berbagai aplikasi komputasi, seperti manajemen memori, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi.

B. Landasan Teori

Stack atau tumpukan adalah salah satu struktur data linear yang bekerja berdasarkan prinsip Last-In, First-Out (LIFO), di mana elemen terakhir yang masuk ke dalam stack akan menjadi elemen pertama yang keluar. Analoginya dapat diumpamakan sebagai tumpukan piring, di mana piring terakhir yang ditumpuk akan menjadi piring pertama yang diambil. Struktur ini sangat berguna dalam berbagai konteks pemrograman karena memudahkan akses terhadap elemen terakhir yang dimasukkan.

Pada implementasinya, stack memiliki beberapa operasi dasar, yaitu:

1. Push: Menambahkan elemen baru ke posisi paling atas dalam stack.
2. Pop: Menghapus elemen teratas dari stack, mengembalikannya ke keadaan sebelum elemen tersebut ditambahkan.
3. Peek atau Top: Mengambil elemen teratas tanpa menghapusnya, untuk mengetahui nilai elemen tersebut.
4. isEmpty: Memeriksa apakah stack kosong atau tidak.
5. isFull: Memeriksa apakah stack sudah penuh atau tidak (dalam kasus implementasi array dengan kapasitas terbatas).

Dalam ilmu komputer, stack memiliki banyak aplikasi praktis, terutama karena kemampuannya untuk menangani data secara efisien melalui prinsip LIFO. Contoh aplikasinya meliputi:

- Manajemen Memori: Stack digunakan dalam penyimpanan alamat memori untuk variabel dan fungsi saat suatu program dieksekusi.
- Evaluasi Ekspresi Aritmatika: Stack digunakan untuk menyusun dan menghitung ekspresi infix, postfix, atau prefix dalam kalkulator atau compiler.
- Pengelolaan Panggilan Fungsi (Call Stack): Stack menyimpan konteks fungsi yang dipanggil secara bertumpuk, memastikan bahwa fungsi yang terakhir dipanggil diselesaikan terlebih dahulu.

Terdapat dua metode implementasi stack, yaitu menggunakan array dan linked list. Dalam implementasi array, ukuran stack terbatas oleh kapasitas array itu sendiri, sehingga jika stack penuh, kondisi "overflow" terjadi. Sebaliknya, dalam implementasi linked list, stack tidak memiliki batas tetap, sehingga ukuran stack hanya dibatasi oleh kapasitas memori.

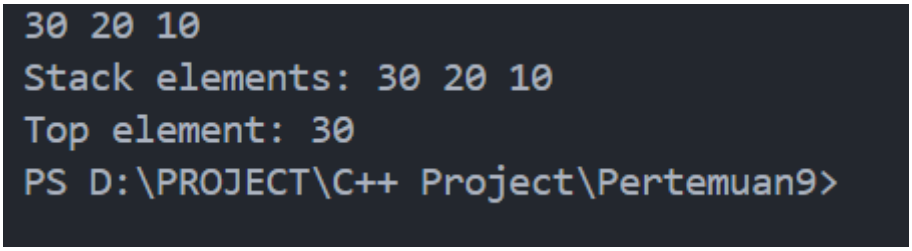
C. Guided

1) Guided 1

Code:

```cpp
#include <iostream>
#define MAX 100
using namespace std;

class stack
{
private:
    int top;
    int arr[MAX];

public:
    stack()
    {
        top = -1;
    }

    bool isFull()
    {
        return top == MAX - 1;
    }

    bool isEmpty()
    {
        return top == -1;
    }

    void push(int x)
    {
        if (isFull())
        {
            cout << "Stack Overflow" << endl;
            return;
        }
        arr[++top] = x;
    }

    int pop()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return -1;
        }
        return arr[top--];
    }

    int peek()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return -1;
        }
        return arr[top];
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Stack Underflow" << endl;
            return;
        }
        for (int i = top; i >= 0; i--)
        {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main()
{
    stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.display();
    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";
    return 0;
}
```

Output:



Deskripsi Program:

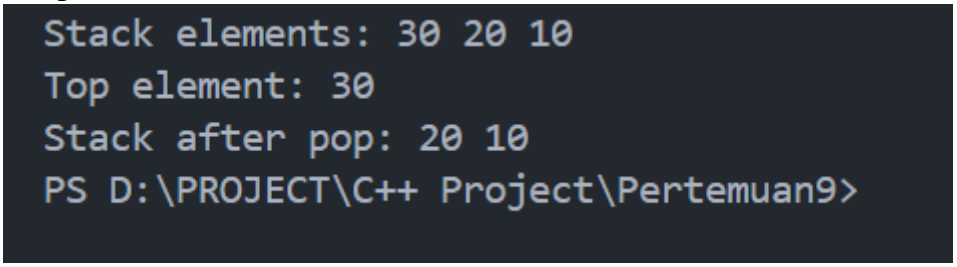
Kode di atas adalah implementasi stack menggunakan struktur data array dalam C++. Kelas stack memiliki variabel top untuk melacak indeks elemen teratas di stack dan arr sebagai tempat penyimpanan data dengan ukuran maksimum MAX (100). Kelas ini memiliki beberapa metode: push untuk menambah elemen ke atas stack, pop untuk menghapus elemen teratas, peek untuk melihat elemen teratas tanpa menghapusnya, dan display untuk menampilkan semua elemen dalam stack dari atas ke bawah. Jika stack penuh, metode push akan mengeluarkan pesan "Stack Overflow", dan jika kosong, pop dan peek akan mengeluarkan "Stack Underflow". Pada main, kita membuat objek stack, menambah beberapa elemen (10, 20, 30), menampilkan elemen di stack, dan mencetak elemen teratas dengan peek.

2) Guided 2

Code:

```cpp
#include <iostream>

using namespace std;

class Node
{
public:
    int data;
    Node *next;
    Node(int value)
    {
        data = value;
        next = nullptr;
    }
};

class Stack
{
private:
    Node *top;

public:
    Stack()
    {
        top = nullptr;
    }

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

    void push(int x)
    {
        Node *newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }

    void pop()
    {
        if (isEmpty())
        {
            cout << "Stack underflow\n";
            return;
        }
        Node *temp = top;
        top = top->next;
        delete temp;
    }

    int peek()
    {
        if (isEmpty())
        {
            cout << "Stack is empty\n";
            return -1;
        }
        return top->data;
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Stack is empty\n";
            return;
        }
        Node *current = top;
        while (current != nullptr)
        {
            cout << current->data << " ";
            current = current->next;
        }
        cout << "\n";
    }
};

int main()
{
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    cout << "Stack after pop: ";
    s.display();

    return 0;
}
```

Output:



Deskripsi Program:

Kode di atas adalah implementasi stack menggunakan struktur data linked list dalam C++. Kelas Node merepresentasikan satu elemen di stack, menyimpan data (data) dan pointer (next) ke node berikutnya. Kelas Stack memiliki pointer top yang menunjuk ke elemen teratas stack. Metode push menambah elemen baru di atas stack dengan mengalokasikan node baru, mengarahkannya ke elemen teratas saat ini, dan mengubah top ke node baru tersebut. Metode pop menghapus elemen teratas dengan memindahkan top ke elemen di bawahnya dan menghapus node sebelumnya untuk mencegah kebocoran memori. Metode peek menampilkan data pada elemen teratas tanpa menghapusnya, dan display menampilkan semua elemen dari atas ke bawah. Pada main, stack diisi dengan nilai 10, 20, dan 30, lalu menampilkan elemen-elemen di dalamnya, menampilkan elemen teratas, dan menghapus elemen teratas sebelum menampilkan kembali isi stack.

D. Unguided

1) Unguided 1

Code:

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

using namespace std;

bool isPalindrome(const string &sentence)
{
    stack<char> charStack;
    string cleanedSentence;

    for (char ch : sentence)
    {
        if (isalnum(ch))
        {
            cleanedSentence += tolower(ch);
            charStack.push(tolower(ch));
        }
    }

    string reversedSentence;
    while (!charStack.empty())
    {
        reversedSentence += charStack.top();
        charStack.pop();
    }

    return cleanedSentence == reversedSentence;
}

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

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

    return 0;
}
```

Output:

```
Masukkan kalimat: ini
Kalimat tersebut adalah palindrom
PS D:\PROJECT\C++ Project\Pertemuan9>
```

Deskripsi Code:

Program ini bekerja untuk memeriksa apakah sebuah kalimat adalah palindrom atau tidak, dengan menggunakan struktur data stack. Pertama, program membersihkan kalimat masukan dengan menghapus spasi dan tanda baca, lalu mengubah semua huruf menjadi huruf kecil agar seragam. Setelah itu, setiap karakter dimasukkan ke dalam stack. Karena stack bekerja dengan prinsip LIFO (Last In, First Out), karakter yang diambil dari stack akan keluar dalam urutan terbalik, sehingga membentuk kalimat terbalik. Program kemudian membandingkan kalimat asli yang sudah dibersihkan dengan kalimat terbalik tersebut. Jika keduanya sama, maka kalimat tersebut adalah palindrom; jika tidak, kalimat tersebut bukan palindrom.

2) Unguided 2

Code:

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

using namespace std;

string reverseSentence(const string &sentence)
{
    stack<char> charStack;
    string reversedSentence;

    for (char ch : sentence)
    {
        charStack.push(ch);
    }

    while (!charStack.empty())
    {
        reversedSentence += charStack.top();
        charStack.pop();
    }

    return reversedSentence;
}

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

    string result = reverseSentence(sentence);

    cout << "Hasil: " << result << endl;

    return 0;
}
```

Output:

```
Masukkan kalimat: Telkom Purwokerto
Hasil: otrekowruP mokleT
PS D:\PROJECT\C++ Project\Pertemuan9>
```

Deskripsi Code:
Program ini menggunakan stack untuk membalik setiap kata dalam kalimat yang dimasukkan. Pertama, program meminta pengguna untuk memasukkan kalimat. Lalu, setiap kata dalam kalimat dipisah menggunakan `stringstream`. Setiap huruf dalam kata tersebut didorong (push) ke dalam stack satu per satu. Karena stack menyimpan data secara bertumpuk, maka ketika huruf-huruf tersebut dikeluarkan (pop) dari stack, urutannya akan terbalik, sehingga menghasilkan kata yang terbalik. Setelah setiap kata terbalik, program menambahkannya ke kalimat hasil dengan spasi sebagai pemisah antar kata. Hasil akhirnya adalah kalimat dengan setiap kata terbalik sesuai urutan aslinya, yang kemudian ditampilkan ke layar.

E. Kesimpulan

Stack adalah struktur data fundamental yang bekerja berdasarkan prinsip Last-In, First-Out (LIFO), di mana elemen terakhir yang ditambahkan akan menjadi elemen pertama yang dikeluarkan. Melalui operasi dasar seperti push, pop, dan peek, stack menyediakan cara yang efisien untuk mengelola data, terutama dalam konteks yang memerlukan pengolahan data secara berurutan atau sementara. Implementasi stack dapat dilakukan menggunakan array maupun linked list, dengan kelebihan dan kekurangan masing-masing, seperti keterbatasan kapasitas pada array dan fleksibilitas pada linked list. Stack memiliki aplikasi yang luas, termasuk dalam manajemen memori, pengolahan ekspresi aritmatika, dan pengelolaan panggilan fungsi. Pemahaman yang kuat tentang konsep dan implementasi stack sangat penting bagi mahasiswa, karena konsep ini tidak hanya sering digunakan dalam pemrograman tetapi juga dalam pengembangan berbagai sistem dan algoritma komputer.