LAPORAN PRAKTIKUM

MODUL VI
STACK

Disusun oleh:
Maulana Ghani Rolanda
NIM: 2311102012

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

a. Mampu memahami konsep stack pada struktur data dan algoritma

b. Mampu mengimplementasikan operasi-operasi pada stack

c. Mampu memecahkan permasalahan dengan solusi stack

BAB II

DASAR TEORI

Stack adalah struktur data yang beroperasi berdasarkan prinsip **LIFO** (Last In, First Out). Ini berarti bahwa elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Berikut adalah operasi dasar yang dapat dilakukan pada stack:

1. **Push**: Menambahkan elemen ke puncak stack.
2. **Pop**: Menghapus elemen dari puncak stack.
3. **Peek**: Mengakses elemen di puncak stack tanpa menghapusnya.
4. **IsEmpty**: Memeriksa apakah stack kosong.

Implementasi Stack

Stack dapat diimplementasikan menggunakan array atau linked list:

1. **Array-based Stack**:
    - **Kelebihan**: Akses elemen cepat karena elemen disimpan dalam lokasi memori yang berdekatan.
    - **Kekurangan**: Ukuran stack tetap (statis) dan perlu didefinisikan sebelumnya.
2. **Linked List-based Stack**:

- **Kelebihan**: Ukuran stack dinamis dan dapat berubah sesuai kebutuhan.
- **Kekurangan**: Memerlukan memori tambahan untuk menyimpan pointer ke elemen berikutnya.

BAB III

GUIDED

1. Guided 1

Source code

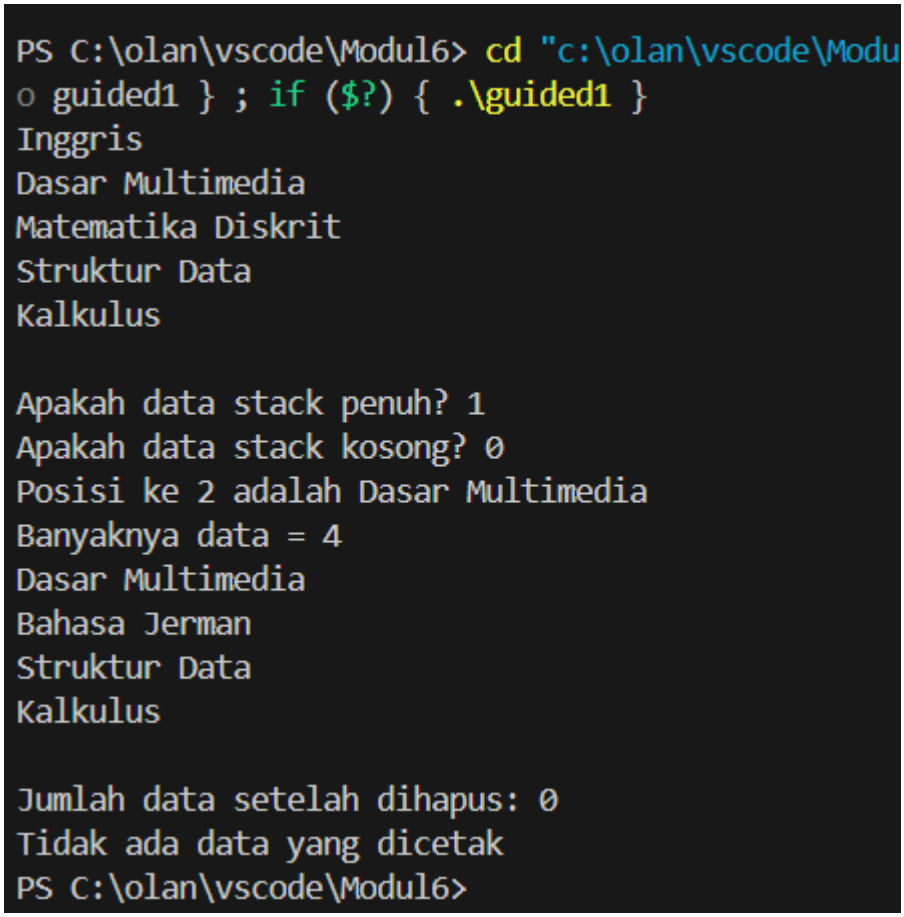
```
#include <iostream>
using namespace std;

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;
}
```

Screenshoot program



Deskripsi program

Program di atas adalah implementasi struktur data **stack** menggunakan array dalam bahasa C++. Stack ini memiliki beberapa fungsi dasar untuk menambah, menghapus, melihat, dan mengubah elemen, serta memeriksa status stack.

LATIHAN KELAS - UNGUIDED

1. **Unguided 1**

   Source code

```cpp
#include <iostream>
#include <stack>
#include <string>
using namespace std;

// Fungsi untuk menghapus spasi dan mengubah huruf menjadi huruf kecil
string preprocessString(const string &str) {
    string result;
    for (char ch : str) {
        if (isalnum(ch)) {
            result += tolower(ch);
        }
    }
    return result;
}

// Fungsi untuk mengecek apakah string adalah palindrom
bool isPalindrome(const string &str) {
    string processedStr = preprocessString(str);
    stack<char> charStack;

    // Masukkan semua karakter ke dalam stack
    for (char ch : processedStr) {
        charStack.push(ch);
    }

    // Periksa apakah string sama ketika dibaca dari depan dan dari stack
    for (char ch : processedStr) {
        if (ch != charStack.top()) {
            return false;
        }
        charStack.pop();
```

```
    }

    return true;
}

int main() {
    string input;
    char choice;

    do {
        // Input kalimat dari pengguna
        cout << "Masukkan kalimat: ";
        getline(cin, input);

        // Periksa apakah kalimat adalah palindrom
        if (isPalindrome(input)) {
            cout << "Kalimat tersebut adalah palindrom" << endl;
        } else {
            cout << "Kalimat tersebut bukan palindrom" << endl;
        }

        // Tanya pengguna apakah ingin mengecek kalimat lain
        cout << "Apakah Anda ingin mengecek kalimat lain? (y/n):
";
        cin >> choice;
        cin.ignore();  // Mengabaikan karakter newline setelah
pilihan

    } while (choice == 'y' || choice == 'Y');

    return 0;
}
```

Screenshoot program

```
PS C:\olan\vscode\Modul6> cd "c:\olan\vscode\Modul6\" ; if
nnerFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan kalimat: telkom
Kalimat tersebut bukan palindrom
Apakah Anda ingin mengecek kalimat lain? (y/n): y
Masukkan kalimat: ketek
Kalimat tersebut adalah palindrom
```

Cara Kerja Program

1. **Looping untuk Pengecekan Berulang**:
    - Program menggunakan do-while loop untuk mengulangi proses pengecekan hingga pengguna memutuskan untuk berhenti.
    - Setelah setiap pengecekan, program menanyakan pengguna apakah ingin mengecek kalimat lain.
2. **Input dan Preprocessing**:
    - Program meminta pengguna untuk memasukkan sebuah kalimat.
    - Fungsi preprocessString membersihkan string dari spasi dan karakter non-alfanumerik, serta mengubah semua huruf menjadi huruf kecil.
3. **Menggunakan Stack**:
    - Program menggunakan stack (charStack) untuk menyimpan karakter-karakter dari string yang telah diproses.
    - Semua karakter dari string yang telah diproses dimasukkan ke dalam stack satu per satu.
4. **Pemeriksaan Palindrom**:
    - Program membandingkan setiap karakter dari string yang telah diproses dengan karakter yang diambil dari puncak stack.
    - Jika semua karakter sama saat dibandingkan, maka string tersebut adalah palindrom.

- Jika ditemukan perbedaan, program segera mengembalikan false.

5. **Output**:
   - Jika string tersebut adalah palindrom, program mencetak "Kalimat tersebut adalah palindrom".
   - Jika tidak, program mencetak "Kalimat tersebut bukan palindrom".
6. **Mengulangi atau Keluar**:
   - Setelah setiap pengecekan, program meminta pengguna untuk memasukkan 'y' atau 'Y' jika ingin mengecek kalimat lain, atau karakter lain untuk keluar dari program.
   - cin.ignore() digunakan untuk mengabaikan karakter newline yang tertinggal di buffer setelah membaca pilihan pengguna.

Dengan cara ini, pengguna dapat terus memeriksa apakah kalimat yang dimasukkan adalah palindrom atau tidak tanpa harus menjalankan kembali program.

2. Unguided 2

Source code

```
#include <iostream>
#include <stack>
#include <sstream>
#include <string>
using namespace std;

// Fungsi untuk membalikkan sebuah kata menggunakan stack
string reverseWord(const string &word) {
    stack<char> charStack;
    for (char ch : word) {
        charStack.push(ch);
    }
```

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

    return reversedWord;
}

// Fungsi untuk membalikkan setiap kata dalam kalimat
string reverseWordsInSentence(const string &sentence) {
    stringstream ss(sentence);
    string word;
    string reversedSentence;

    // Memproses setiap kata dalam kalimat
    while (ss >> word) {
        reversedSentence += reverseWord(word) + " ";
    }

    // Menghapus spasi ekstra di akhir kalimat
    if (!reversedSentence.empty()) {
        reversedSentence.pop_back();
    }

    return reversedSentence;
}

int main() {
    string input;

    // Input kalimat dari pengguna
    cout << "Masukkan kalimat (minimal 3 kata): ";
    getline(cin, input);

    // Mengecek apakah kalimat memiliki minimal 3 kata
    stringstream ss(input);
    int wordCount = 0;
    string word;
    while (ss >> word) {
        wordCount++;
```

```
    }

    if (wordCount < 3) {
        cout << "Kalimat harus memiliki minimal 3 kata." << endl;
    } else {
        // Membalikkan setiap kata dalam kalimat
        string reversedSentence = reverseWordsInSentence(input);
        cout << "Kebalikan: " << reversedSentence << endl;
    }

    return 0;
}
```

Screenshoot program

```
PS C:\olan\vscode\Modul6> cd "c:\olan\vscode\Modul6\" ; if ($?) { g
 if ($?) { .\M6-Unguided2 }
Masukkan kalimat (minimal 3 kata): institut telkom purwokerto
Kebalikan: tutitsni moklet otrekowrup
```

Deskripsi program

Program ini meminta pengguna untuk memasukkan sebuah kalimat dan kemudian memeriksa apakah kalimat tersebut memiliki minimal tiga kata. Jika memenuhi syarat, program akan membalikkan setiap kata dalam kalimat menggunakan stack. Pertama, program memecah kalimat menjadi kata-kata menggunakan stringstream. Setiap kata kemudian dibalik dengan memasukkan karakter-karakter kata tersebut ke dalam stack dan mengeluarkannya satu per satu untuk membentuk kata yang dibalik. Setelah semua kata diproses, kata-kata yang dibalik digabungkan kembali menjadi kalimat baru yang kemudian ditampilkan kepada pengguna.

BAB IV

KESIMPULAN

Dari praktikum ini, kita dapat menyimpulkan beberapa hal penting mengenai konsep dan implementasi stack pada struktur data.

Pemahaman Konsep Stack

Stack adalah struktur data yang mengikuti prinsip LIFO (Last In, First Out), di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Operasi dasar pada stack meliputi push (menambahkan elemen), pop (menghapus elemen), peek (mengakses elemen di puncak tanpa menghapusnya), dan isEmpty (memeriksa apakah stack kosong).

Implementasi Stack

Stack dapat diimplementasikan menggunakan array atau linked list. Implementasi array memberikan akses elemen yang cepat tetapi dengan ukuran yang tetap, sedangkan linked list memungkinkan ukuran dinamis dengan biaya tambahan untuk penyimpanan pointer.

Praktikum Guided

Implementasi stack menggunakan array dalam bahasa C++ mengajarkan bagaimana mengelola data dengan operasi dasar stack. Program dapat menambah, menghapus, mengubah, dan mencetak elemen dalam stack serta memeriksa status penuh atau kosongnya stack.

Latihan Unguided

Mengecek Palindrom: Program menggunakan stack untuk mengecek apakah suatu kalimat adalah palindrom, dengan membersihkan kalimat dari karakter non-alfanumerik dan mengubahnya menjadi huruf kecil. Program ini

mengilustrasikan penggunaan stack untuk memecahkan masalah pemrosesan string secara efisien.

Membalikkan Kata dalam Kalimat: Program membalikkan setiap kata dalam kalimat menggunakan stack, yang menunjukkan bagaimana stack dapat digunakan untuk manipulasi string dan pemrosesan kata.

Pemecahan Masalah dengan Stack

Praktikum ini membuktikan bahwa stack adalah alat yang berguna untuk berbagai masalah algoritma dan pemrograman, seperti pemrosesan string, pengecekan palindrom, dan pembalikan kata. Stack memberikan solusi yang efisien dan intuitif untuk masalah yang melibatkan urutan dan pembalikan data.

Kesimpulan Akhir

Secara keseluruhan, praktikum ini memberikan pemahaman yang mendalam tentang konsep stack dan aplikasinya dalam pemrograman, serta bagaimana struktur data ini dapat digunakan untuk memecahkan masalah nyata dengan cara yang efisien dan efektif.

DAFTAR PUSTAKA

Geeks For Geeks 2 Mei, 2024. Stack Data Structure, Diakses pada 22 Mei 2024 dari https://www.geeksforgeeks.org/stack-data-structure/