LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL VII

STACK

Disusun Oleh :
Muhammad Fathammubina
NIM : 103112430188

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

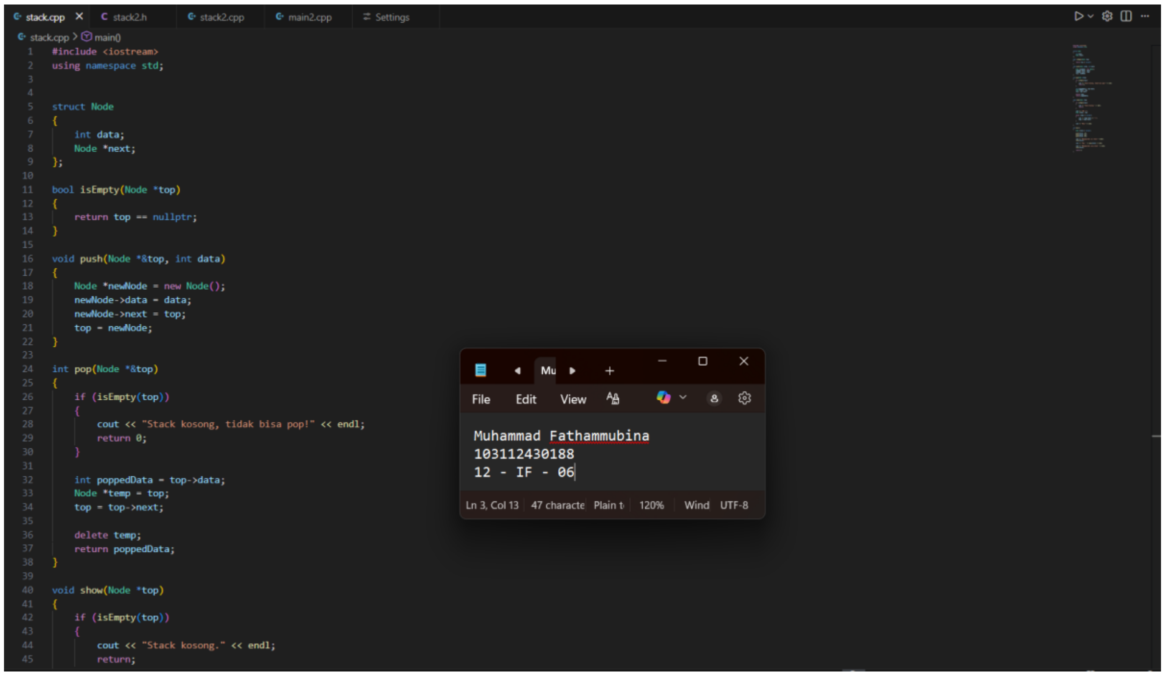
A. Dasar Teori

Stack adalah struktur data linear yang bekerja dengan prinsip LIFO (Last In First Out), yaitu elemen yang terakhir masuk akan menjadi elemen pertama yang keluar. Stack hanya memiliki satu akses yaitu pada bagian atas yang disebut TOP, sehingga semua operasi seperti penambahan data (push) dan penghapusan data (pop) dilakukan melalui TOP. Stack dapat diimplementasikan menggunakan pointer (linked list) maupun array, di mana implementasi pointer bersifat dinamis sedangkan array memiliki batasan ukuran tertentu. Selain operasi dasar, stack juga dapat dilengkapi fungsi tambahan seperti pengecekan kosong/penuh, pembalikan urutan (balikStack), pengurutan saat memasukkan data (pushAscending), hingga membaca input karakter berturut-turut (getInputStream). Struktur data stack banyak digunakan dalam berbagai proses komputasi, seperti pemanggilan fungsi, backtracking, dan fitur undo/redo karena sifatnya yang efektif dalam mengelola data secara berurutan.

B. Guided

Guided 1

stack.cpp



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


struct Node
{
    int data;
    Node *next;
};

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

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

int pop(Node *&top)
{
    if (isEmpty(top))
    {
        cout << "Stack kosong, tidak bisa pop!" << endl;
        return 0;
    }

    int poppedData = top->data;
    Node *temp = top;
    top = top->next;

    delete temp;
    return poppedData;
}

void show(Node *top)
{
    if (isEmpty(top))
    {
        cout << "Stack kosong." << endl;
        return;
```

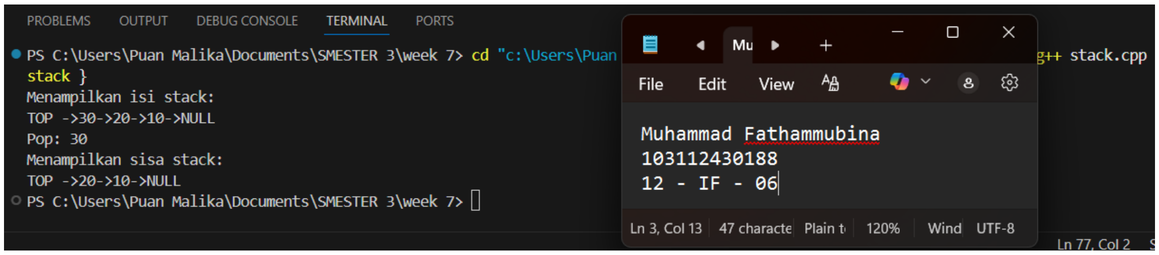
```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

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


struct Node
{
    int data;
    Node *next;
};
```

```cpp
bool isEmpty(Node *top)
{
    return top == nullptr;
}

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

int pop(Node *&top)
{
    if (isEmpty(top))
    {
        cout << "Stack kosong, tidak bisa pop!" << endl;
        return 0;
    }

    int poppedData = top->data;
    Node *temp = top;
    top = top->next;

    delete temp;
    return poppedData;
}

void show(Node *top)
{
    if (isEmpty(top))
    {
        cout << "Stack kosong." << endl;
        return;
    }

    cout << "TOP ->";
    Node *temp = top;

    while (temp != nullptr)
    {
        cout << temp->data << "->";
        temp = temp->next;
    }

    cout << "NULL" << endl;
}
```

```cpp
int main()
{
    Node *stack = nullptr;

    push(stack, 10);
    push(stack, 20);
    push(stack, 30);

    cout << "Menampilkan isi stack:" <<endl;
    show(stack);

    cout << "Pop: " << pop(stack) << endl;

    cout << "Menampilkan sisa stack:" << endl;
    show(stack);

    return 0;
}
```

Screenshots Output



```
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 7> cd "c:\Users\Puan                g++ stack.cpp
stack }
Menampilkan isi stack:
TOP ->30->20->10->NULL
Pop: 30
Menampilkan sisa stack:
TOP ->20->10->NULL
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 7>
```

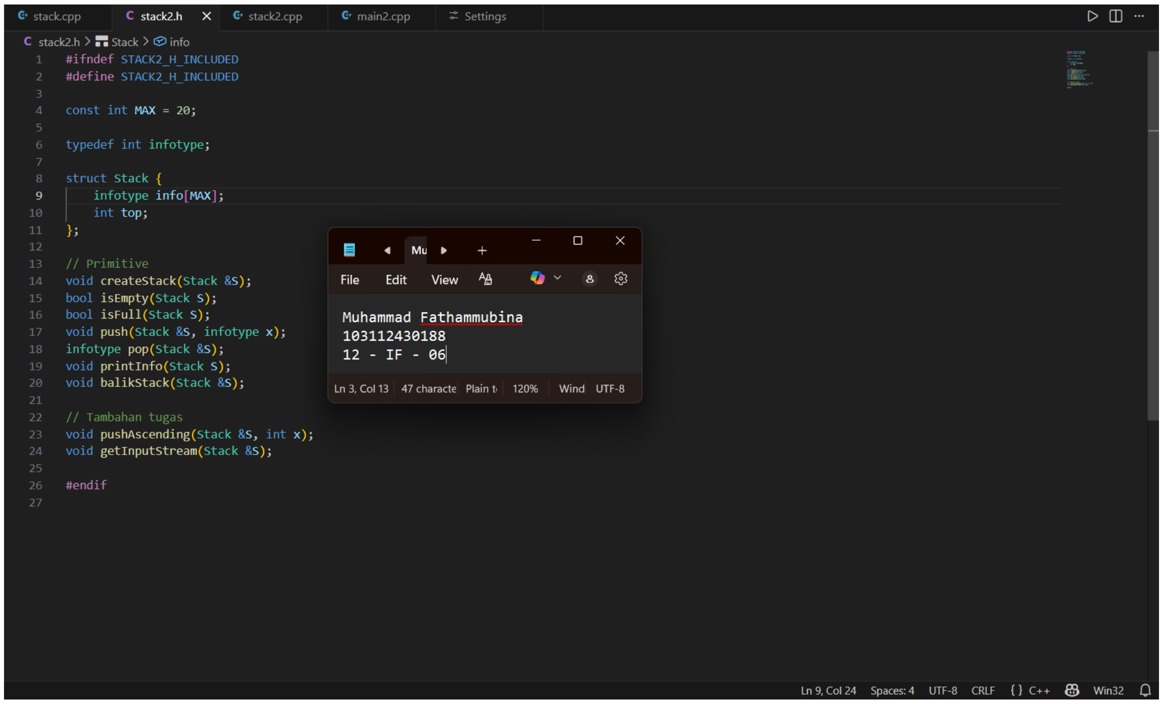
```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

Deskripsi:

Program di atas adalah implementasi struktur data stack menggunakan linked list. Setiap elemen stack disimpan dalam node yang berisi data dan pointer ke node berikutnya. Fungsi push() digunakan untuk menambah elemen di bagian atas stack, sementara pop() menghapus elemen teratas dan mengembalikan nilainya. Fungsi show() menampilkan seluruh isi stack dari elemen paling atas hingga paling bawah. Program utama membuat sebuah stack kosong, menambahkan tiga data (10, 20, 30), menampilkannya, melakukan satu operasi pop, lalu menampilkan kembali isi stack setelah penghapusan. Program ini menunjukkan cara kerja stack dengan prinsip LIFO (Last In, First Out).

Unguided 1

stack2.h

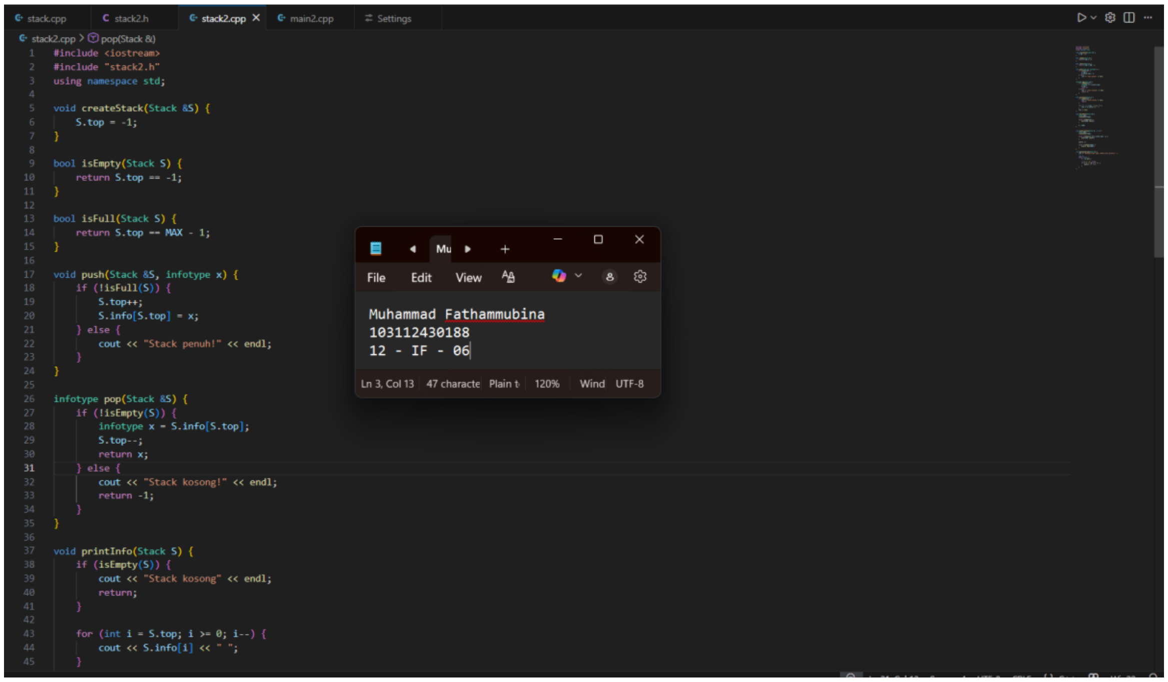


```
#ifndef STACK2_H_INCLUDED
#define STACK2_H_INCLUDED

const int MAX = 20;

typedef int infotype;

struct Stack {
    infotype info[MAX];
    int top;
};

// Primitive
void createStack(Stack &S);
bool isEmpty(Stack S);
bool isFull(Stack S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

// Tambahan tugas
void pushAscending(Stack &S, int x);
void getInputStream(Stack &S);

#endif
```

```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

```
#ifndef STACK2_H_INCLUDED
#define STACK2_H_INCLUDED

const int MAX = 20;

typedef int infotype;

struct Stack {
    infotype info[MAX];
    int top;
};

// Primitive
void createStack(Stack &S);
bool isEmpty(Stack S);
bool isFull(Stack S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

// Tambahan tugas
void pushAscending(Stack &S, int x);
void getInputStream(Stack &S);

#endif
```

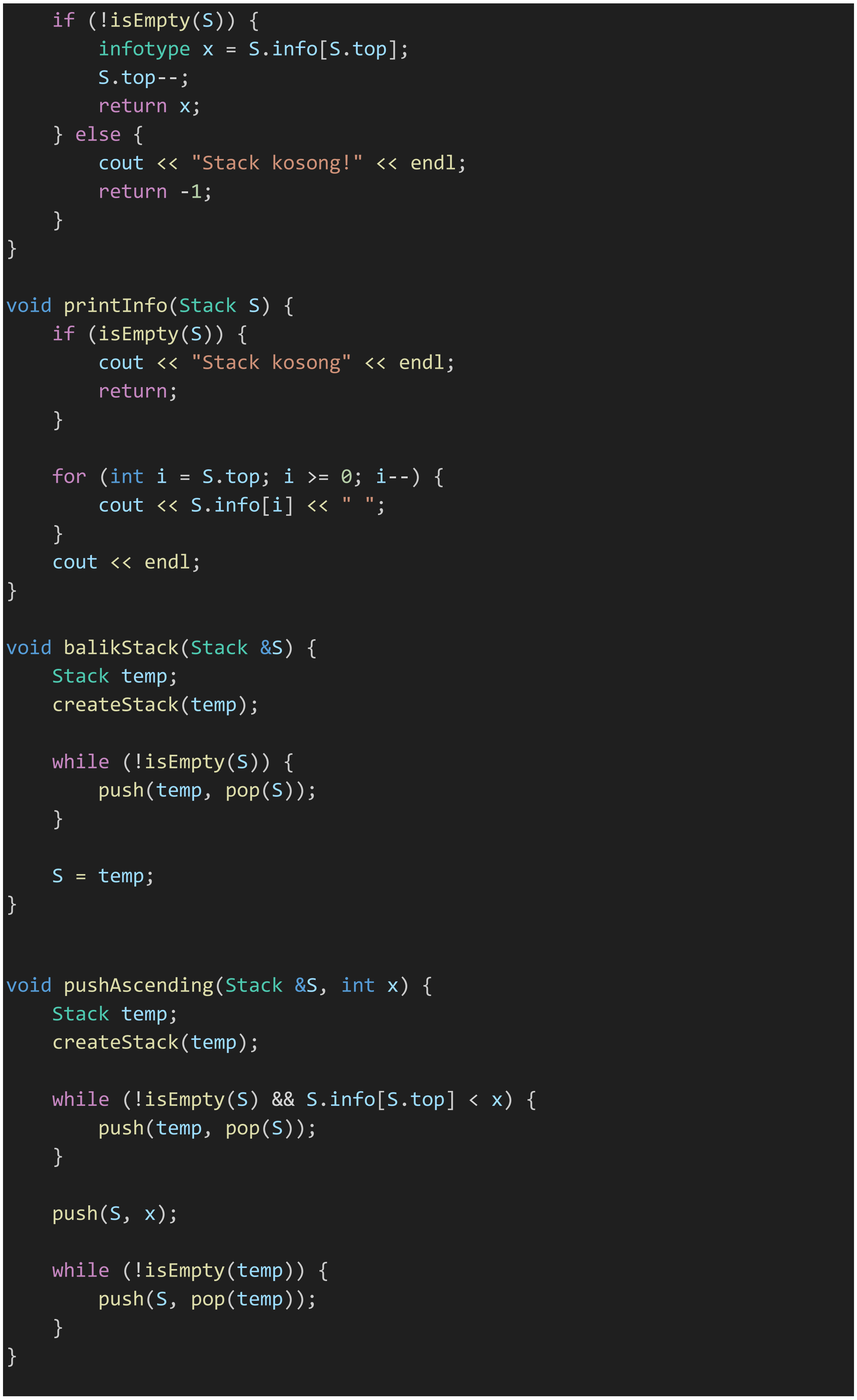
stack2.cpp



```cpp
#include <iostream>
#include "stack2.h"
using namespace std;

void createStack(Stack &S) {
    S.top = -1;
}

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

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

void push(Stack &S, infotype x) {
    if (!isFull(S)) {
        S.top++;
        S.info[S.top] = x;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

infotype pop(Stack &S) {
    if (!isEmpty(S)) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    } else {
        cout << "Stack kosong!" << endl;
        return -1;
    }
}

void printInfo(Stack S) {
    if (isEmpty(S)) {
        cout << "Stack kosong" << endl;
        return;
    }

    for (int i = S.top; i >= 0; i--) {
        cout << S.info[i] << " ";
    }
```

```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

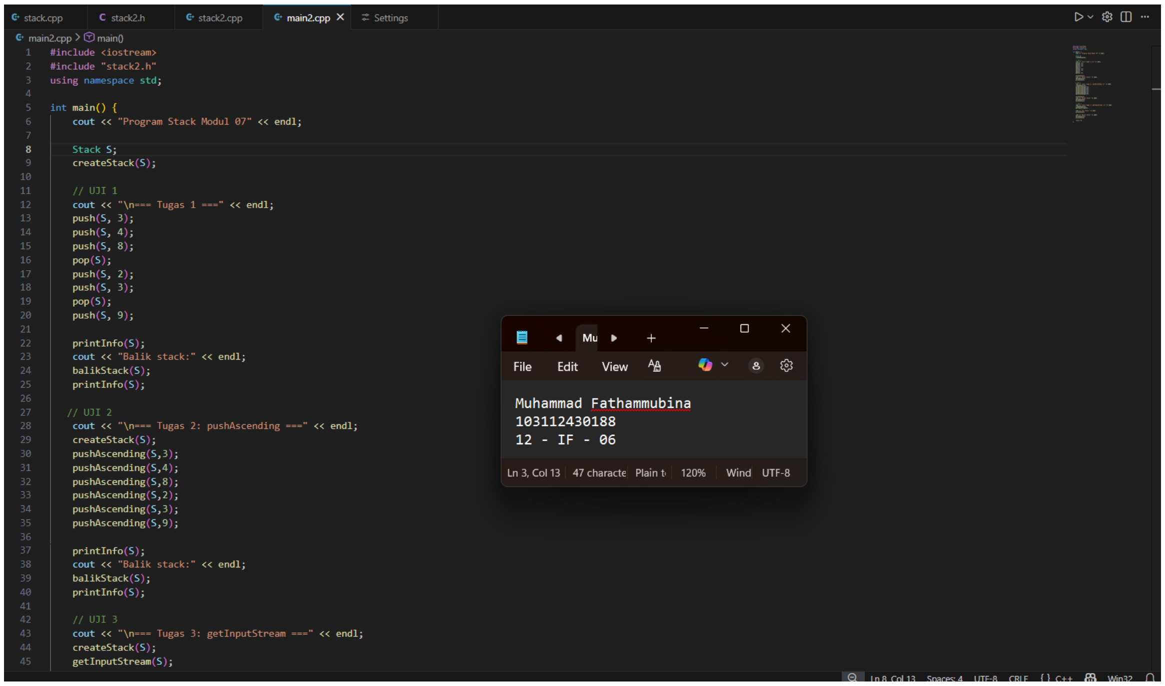
```cpp
#include <iostream>
#include "stack2.h"
using namespace std;

void createStack(Stack &S) {
    S.top = -1;
}

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

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

void push(Stack &S, infotype x) {
    if (!isFull(S)) {
        S.top++;
        S.info[S.top] = x;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

infotype pop(Stack &S) {
```



```cpp
    if (!isEmpty(S)) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
    } else {
        cout << "Stack kosong!" << endl;
        return -1;
    }
}

void printInfo(Stack S) {
    if (isEmpty(S)) {
        cout << "Stack kosong" << endl;
        return;
    }

    for (int i = S.top; i >= 0; i--) {
        cout << S.info[i] << " ";
    }
    cout << endl;
}

void balikStack(Stack &S) {
    Stack temp;
    createStack(temp);

    while (!isEmpty(S)) {
        push(temp, pop(S));
    }

    S = temp;
}


void pushAscending(Stack &S, int x) {
    Stack temp;
    createStack(temp);

    while (!isEmpty(S) && S.info[S.top] < x) {
        push(temp, pop(S));
    }

    push(S, x);

    while (!isEmpty(temp)) {
        push(S, pop(temp));
    }
}
```

```cpp
void getInputStream(Stack &S) {
    cout << "Masukkan digit angka (ENTER untuk berhenti): ";

    char c;
    while (true) {
        c = cin.get();

        if (c == '\n') break;
        if (c >= '0' && c <= '9') {
            push(S, c - '0');
        }
    }
}
```

main2.cpp

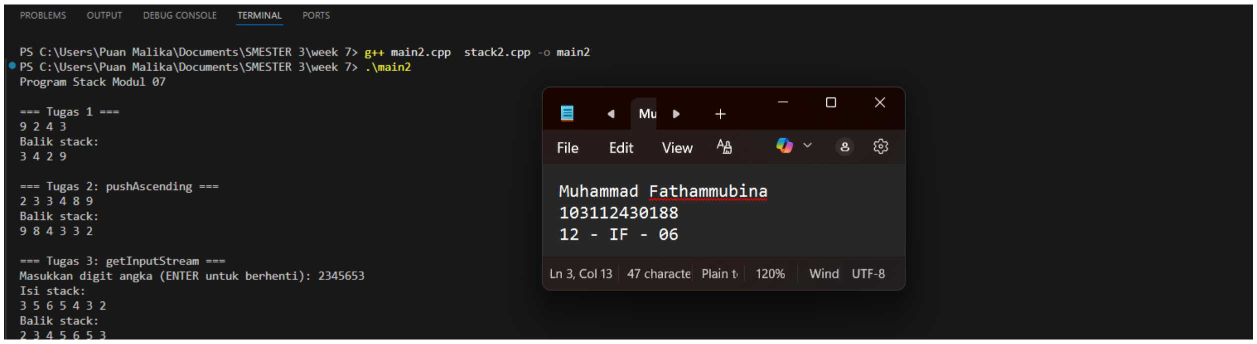


```cpp
#include <iostream>
#include "stack2.h"
using namespace std;

int main() {
    cout << "Program Stack Modul 07" << endl;

    Stack S;
    createStack(S);

    // UJI 1
    cout << "\n=== Tugas 1 ===" << endl;
    push(S, 3);
    push(S, 4);
    push(S, 8);
    pop(S);
    push(S, 2);
    push(S, 3);
    pop(S);
    push(S, 9);

    printInfo(S);
    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

   // UJI 2
    cout << "\n=== Tugas 2: pushAscending ===" << endl;
    createStack(S);
    pushAscending(S,3);
    pushAscending(S,4);
    pushAscending(S,8);
    pushAscending(S,2);
    pushAscending(S,3);
    pushAscending(S,9);

    printInfo(S);
    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

    // UJI 3
    cout << "\n=== Tugas 3: getInputStream ===" << endl;
    createStack(S);
    getInputStream(S);
```

```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

```cpp
#include <iostream>
#include "stack2.h"
using namespace std;

int main() {
    cout << "Program Stack Modul 07" << endl;

    Stack S;
    createStack(S);

    // UJI 1
    cout << "\n=== Tugas 1 ===" << endl;
    push(S, 3);
    push(S, 4);
    push(S, 8);
    pop(S);
    push(S, 2);
    push(S, 3);
    pop(S);
    push(S, 9);

    printInfo(S);
    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

   // UJI 2
    cout << "\n=== Tugas 2: pushAscending ===" << endl;
    createStack(S);
    pushAscending(S,3);
    pushAscending(S,4);
    pushAscending(S,8);
    pushAscending(S,2);
    pushAscending(S,3);
    pushAscending(S,9);

    printInfo(S);
    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

    // UJI 3
    cout << "\n=== Tugas 3: getInputStream ===" << endl;
    createStack(S);
    getInputStream(S);

    cout << "Isi stack:" << endl;
    printInfo(S);

    cout << "Balik stack:" << endl;
    balikStack(S);
    printInfo(S);

    return 0;
}
```

Screenshots Output



```
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 7> g++ main2.cpp  stack2.cpp -o main2
PS C:\Users\Puan Malika\Documents\SMESTER 3\week 7> .\main2
Program Stack Modul 07

=== Tugas 1 ===
9 2 4 3
Balik stack:
3 4 2 9

=== Tugas 2: pushAscending ===
2 3 3 4 8 9
Balik stack:
9 8 4 3 3 2

=== Tugas 3: getInputStream ===
Masukkan digit angka (ENTER untuk berhenti): 2345653
Isi stack:
3 5 6 5 4 3 2
Balik stack:
2 3 4 5 6 5 3
```

```
Muhammad Fathammubina
103112430188
12 - IF - 06
```

Deskripsi:

Program stack diatas menggunakan array berukuran 20 elemen untuk menyimpan data dengan prinsip LIFO (Last In First Out). Program menyediakan operasi dasar seperti push untuk menambah data, pop untuk mengambil data teratas, serta printInfo untuk menampilkan isi stack. Selain itu, terdapat fitur tambahan seperti pushAscending yang memasukkan data secara berurutan naik dan getInputStream yang membaca input angka dari pengguna hingga ENTER ditekan. Program ini menunjukkan bagaimana stack bekerja dan bagaimana elemen dapat dimanipulasi melalui berbagai operasi.

C. Kesimpulan

D. Referensi

Muliono, R. (2017). Abstract Data Type (ADT). Universitas Multimedia Nusantara.

Trivusi. (2022, September 16). *Struktur Data Stack: Pengertian, Karakteristik, dan Kegunaannya*. Trivusi.