LAPORAN PRAKTIKUM
Modul 7
“STACK”

Disusun Oleh:
Rifqi M Ramdani -2311104044
SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. **Tujuan**

   1.Memahami konsep stack.

   2. Mengimplementasikan stack dengan menggunakan representasi pointer dan tabel.

   3. Memahami konsep queue.

   4. Mengimplementasikan queue dengan menggunakan pointer dan tabel.

2. **Landasan Teori**

   Stack adalah salah satu struktur data yang menggunakan prinsip LIFO (Last In First Out), yang berarti elemen yang terakhir dimasukkan akan diakses pertama kali. Operasi dasar dalam stack adalah Push (penyisipan) dan Pop (pengambilan). Pada representasi tabel, stack menggunakan array berindeks dan memiliki kapasitas yang terbatas.

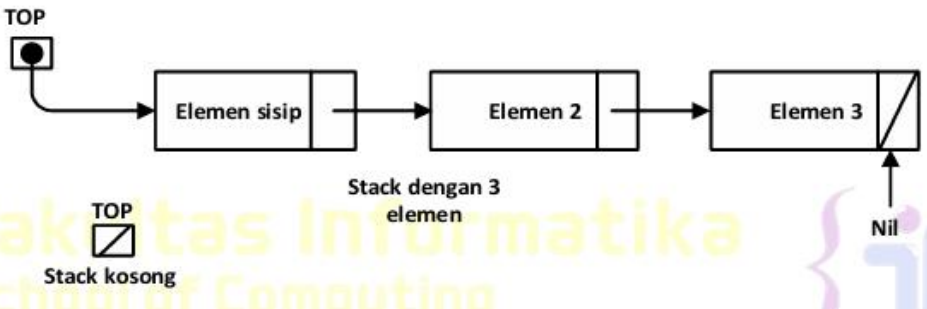
3. **Guided**

   Pengertian Stack

   Stack merupakan salah satu bentuk struktur data dimana prinsip operasi yang digunakan seperti tumpukan. Seperti halnya tumpukan, elemen yang bisa diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang pertama kali masuk, prinsip ini biasa disebut LIFO (Last In First Out)

   Komponen-Komponen dalam Stack

   Komponen – komponen dalam stack pada dasarnya sama dengan komponen pada single linked list. Hanya saja akses pada stack hanya bisa dilakukan pada awal stack saja.



   Seperti terlihat pada gambar diatas bentuk stack mirip seperti list linier, yang terdiri dari elemen – elemen yang saling terkait. Komponen utama dalam stack yang berfungsi untuk mengakses data dalam stack adalah elemen paling awal saja yang disebut “Top”. Pendeklarasian tipedata stack:

```
1   #ifndef stack_H
2   #define stack_H
3
4   #define Nil NULL
5   #define info(P) (P)->info
6   #define next(P) (P)->next
7   #define Top(S) ((S).Top)
8
9   typedef int infotype; /* tipe data dalam stack */
10  typedef struct elmStack *address;
11  /* tipe data pointe untuk elemen stack */
12  struct elmStack {
13      infotype info;
14      address next;
15  }; /*tipe data elemen stack */
16
17  /* pendeklarasian tipe data stack */
18  struct stack {
19      address Top;
20  };
21  #endif
```

Keterangan:

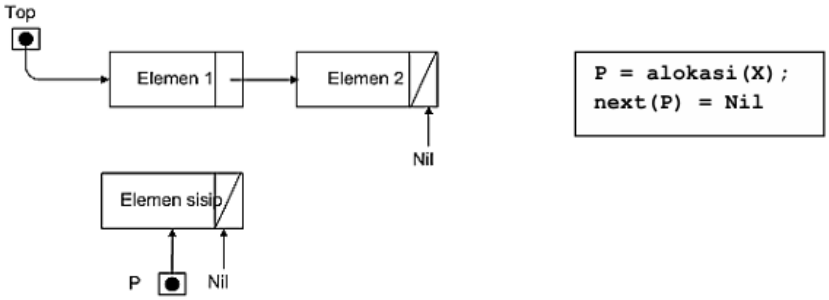
1. Dalam stack hanya terdapat TOP
2. 2. Tipe address adalah tipe elemen stack yang sama dengan elemen dalam list lainnya.

Operasi-Operasi dalam Stack

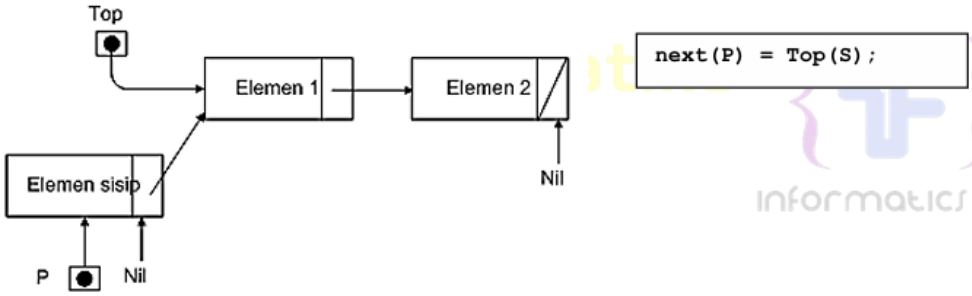
Dalam stack ada dua operasi utama, yaitu operasi penyisipan (Push) dan operasi pengambilan (Pop).

Push

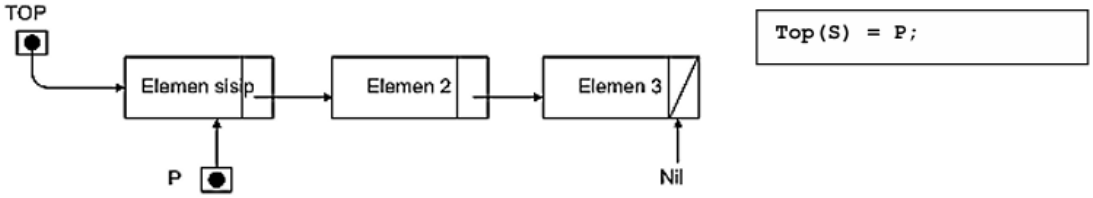
Adalah operasi menyisipkan elemen pada tumpukan data. Fungsi ini sama dengan fungsi insert first pada list biasa. Langkah – langkah dalam proses Push:



Gambar 7-2 *Stack Push* 1



Gambar 7-3 *Stack Push* 2

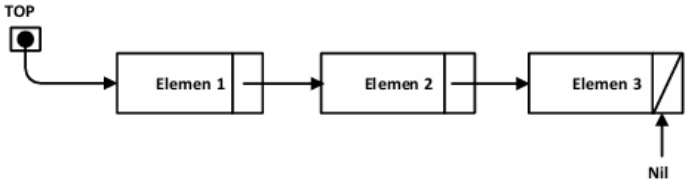


Gambar 7-4 *Stack Push* 3

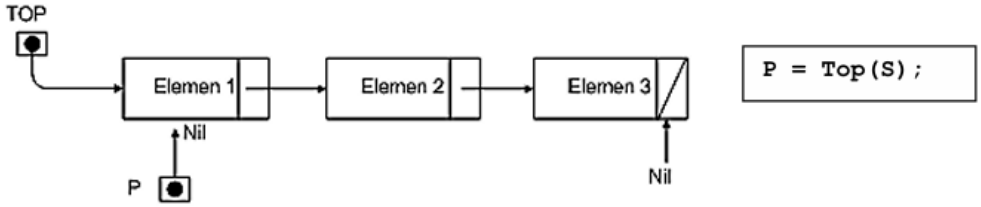
```
/* buat dahulu elemen yang akan disisipkan */
address createElm(int x){
    address p = alokasi(x);
    next(p) = null;
    return p;
}

/* contoh sintak push */
void push(address p){
    next(p) = top(s);
    top(s) = p;
}
```

Pop

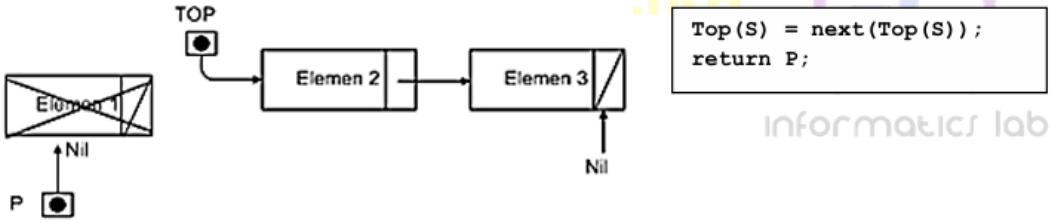
Adalah operasi pengambilan data dalam list. Operasi ini mirip dengan operasi delete first dalam list linear, karena elemen yang paling pertama kali diakses adalah elemen paling atas atau elemen paling awal saja. Langkah-langkah dalam proses Pop:



Gambar 7-5 *Stack Pop* 1



Gambar 7-6 *Stack Pop* 2



Gambar 7-7 *Stack Pop* 3

```
/* contoh sintak pop */
address pop(address p){
    p = top(s);
    top(s) = next(top(s));
    return p;
}
```

Primitif-Primitif dalam Stack

Primitif-primitif dalam stack pada dasarnya sama dengan primitif-primitif pada list lainnya. Malahan primitif dalam stack lebih sedikit, karena dalam stack hanya melakukan operasi-operasi terhadap elemen paling atas.

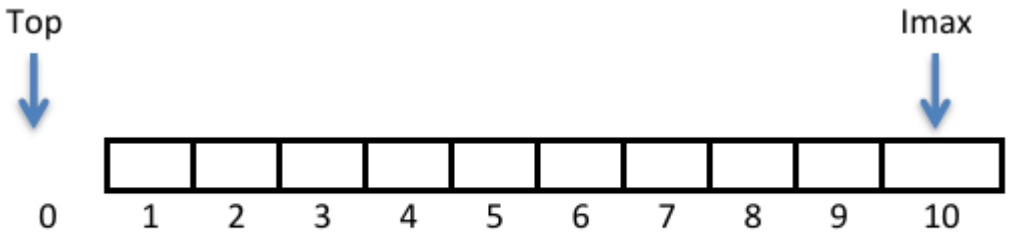
Primitif -primitif dalam stack :

1. createStack().
2. isEmpty().
3. alokasi().
4. dealokasi().
5. Fungsi – fungsi pencarian.
6. Dan fungsi – fungsi primitif lainnya.

Seperti halnya pada model list yang lain, primitif-primitifnya tersimpan pada file *.c dan file *.h.

Stack (Representasi Tabel)

Pada prinsipnya representasi menggunakan tabel sama halnya dengan menggunakan pointer. Perbedaannya terletak pada pendeklarasian struktur datanya, menggunakan array berindeks dan jumlah tumpukan yang terbatas.



Gambar 7-8 *Stack* Kosong dengan Representasi Table

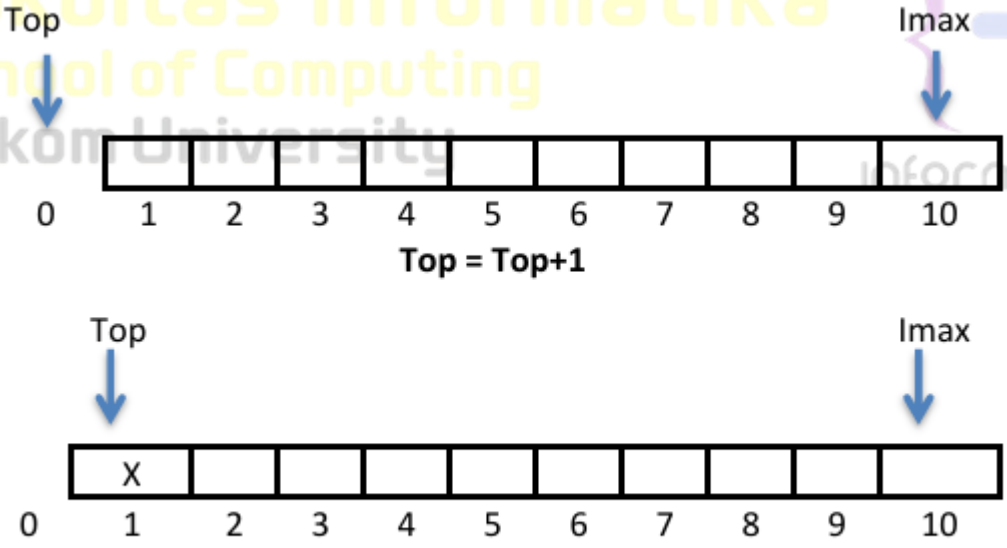
Gambar di atas menunjukan stack maksimum terdapat pada indeks Imax=10, sedangkan Stack masih kosong karena Top = 0.

Operasi-operasi Dalam Stack

Operasi-operasi dalam stack representasi tabel pada dasarnya sama dengan representasi pointer, yaitu PUSH dan POP.

A. Push

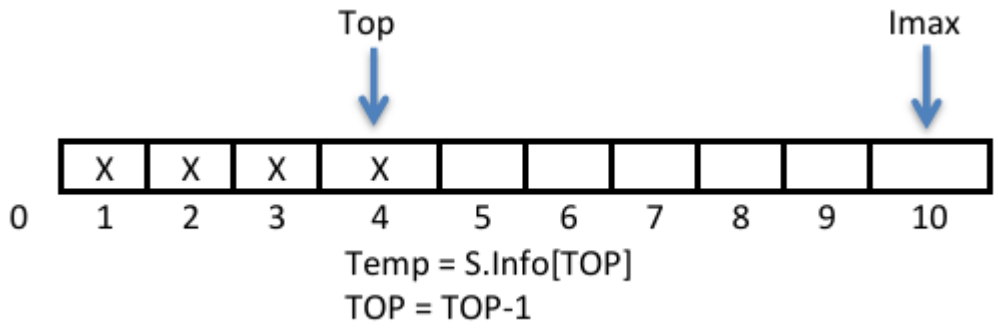
Push merupakan operasi penyisipan data ke dalam stack, penyisipan dilakukan dengan menggeser indeks dari TOP ke indeks berikutnya. Perhatikan contoh dibawah ini:

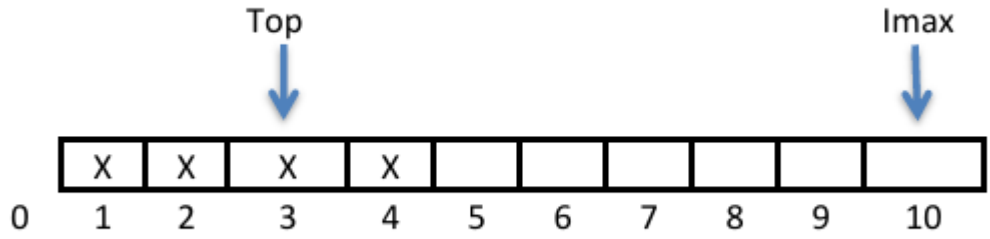


Gambar 7-9 *Push* Elemen dengan Representasi Tabel

B.Pop

Pop merupakan operasi pengambilan data di posisi indeks TOP berada dalam sebuah stack. Setelah data diambil, indeks TOP akan bergeser ke indeks sebelum TOP tanpa menghilangkan info dari indeks TOP sebelumnya. Perhatikan contoh dibawah ini:





Gambar 7-10 *Pop* Elemen dengan Representasi Tabel

Primitif-primitif Dalam Stack Primitif-primitif pada stack representasi tabel pada dasarnya sama dengan representasi pointer. Perbedaanya hanya pada manajemen memori, pada representasi tabel tidak diperlukan manajemen memori. antara lain sebagai berikut,

1. createStack().
2. isEmpty().
3. Fungsi – fungsi pencarian.
4. Dan fungsi – fungsi primitif lainnya.

Seperti halnya pada model list yang lain, primitif-primitifnya tersimpan pada file *.c dan file *.h.

Untuk lebih memahami struktur data dari stack representasi tabel, berikut ini contoh ADT stack representasi tabel dalam file *.h:

```
1   /* file : stack.h
2      contoh ADT stack dengan representasi tabel */
3   #ifndef STACK_H_INCLUDED
4   #define STACK_H_INCLUDED
5
6   #include <stdio.h>
7   #include <conio.h>
8   struct infotype {
9       char nim[20];
10      char nama[20];
11  };
12  struct Stack {
13      infotype info[10];
14      int Top;
15  };
16
17  /* prototype */
18  /******** pengecekan apakah Stack kosong *********/
19  int isEmpty(Stack S);
20  /* mengembalikan nilai 0 jika stack kosong */
21  /******** pembuatan Stack ********/
22  void createStack(Stack &S);
23  /* I.S. sembarang
24     F.S. terbentuk stack dengan Top=0 */
25  /******** penambahan elemen pada Stack ********/
26  void push(Stack &S, infotype X);
27  /* I.S. Stack mungkin kosong
28     F.S. menambahkan elemen pada stack dengan nilai X, TOP=TOP+1 */
29  /******** penghapusan elemen pada list ********/
30  void pop(Stack &S, infotype &X);
31  /* I.S. stack tidak kosong
32     F.S. nilai info pada indeks TOP disimpan pada X, kemudian TOP=TOP-1 */
33  /******** proses semua elemen list *********/
34  void viewStack(Stack S);
35  /* I.S. stack mungkin kosong
36     F.S. jika stack tidak kosong menampilkan semua info yang ada pada stack
    */
37  #endif // STACK_H_INCLUDED
```

Guided

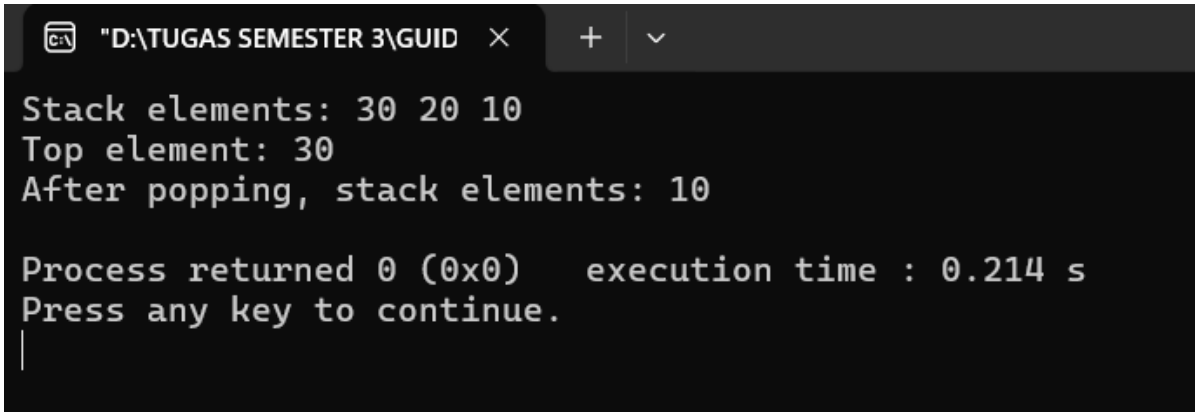
Kode Program

1.

```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\n";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[top];
        }
        cout << "Stack is empty\n";
        return -1; // Return a sentinel value
    }

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

```
47              cout << arr[i] << " ";
48          }
49          cout << "\n";
50      }
51  };
52
53  int main() {
54      Stack s;
55      s.push(10);
56      s.push(20);
57      s.push(30);
58
59      cout << "Stack elements: ";
60      s.display();
61
62      cout << "Top element: " << s.peek() << "\n";
63
64      s.pop();
65      s.pop();
66      cout << "After popping, stack elements: ";
67      s.display();
68
69      return 0;
70  }
71
```

Maka Akan menghasilkan output



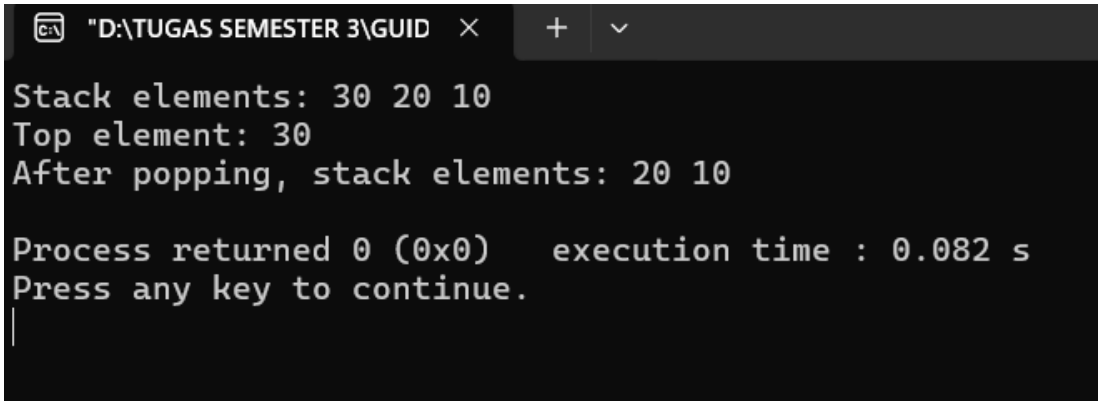
2.

```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;
```

```
37          delete temp;
38      }
39
40      int peek() {
41          if (!isEmpty()) {
42              return top->data;
43          }
44          cout << "Stack is empty\n";
45          return -1; // Return a sentinel value
46      }
47
48      void display() {
49          if (isEmpty()) {
50              cout << "Stack is empty\n";
51              return;
52          }
53          Node* current = top;
54          while (current) {
55              cout << current->data << " ";
56              current = current->next;
57          }
58          cout << "\n";
59      }
60  };
61
62  int main() {
63      Stack s;
64      s.push(10);
65      s.push(20);
66      s.push(30);
67
68      cout << "Stack elements: ";
69      s.display();
70
71      cout << "Top element: " << s.peek() << "\n";
72
73      s.pop();
```

```
73      s.pop();
74      cout << "After popping, stack elements: ";
75      s.display();
76
77      return 0;
78  }
79
```

Maka akan menghasilkan output



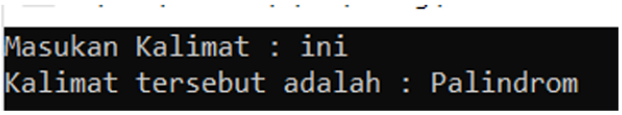
4. Unguided

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

   Contoh:

   Kalimat : ini
   Kalimat tersebut adalah polindrom

   Kalimat : telkom
   Kalimat tersebut adalah bukan polindrom



   JAWABAN
   Kode Program:

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

bool isPalindrome(string kalimat) {
    stack<char> s;

    kalimat.erase(remove(kalimat.begin(), kalimat.end(), ' '), kalimat.end());
    transform(kalimat.begin(), kalimat.end(), kalimat.begin(), ::tolower);

    for (char ch : kalimat) {
        s.push(ch);
    }

    string kalimatTerbalik;
    while (!s.empty()) {
        kalimatTerbalik += s.top();
        s.pop();
    }

    return kalimat == kalimatTerbalik;
}

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

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

    return 0;
}
```

Preproses Kalimat: Menghapus spasi dari kalimat dan mengubah semua huruf menjadi huruf kecil menggunakan erase dan transform.
Memasukkan Karakter ke Stack: Setiap karakter dalam kalimat dimasukkan ke dalam stack.
Membalik Kalimat: Karakter diambil dari stack untuk membentuk versi terbalik dari kalimat.
Perbandingan: Kalimat asli dan kalimat terbalik dibandingkan. Jika sama, maka kalimat

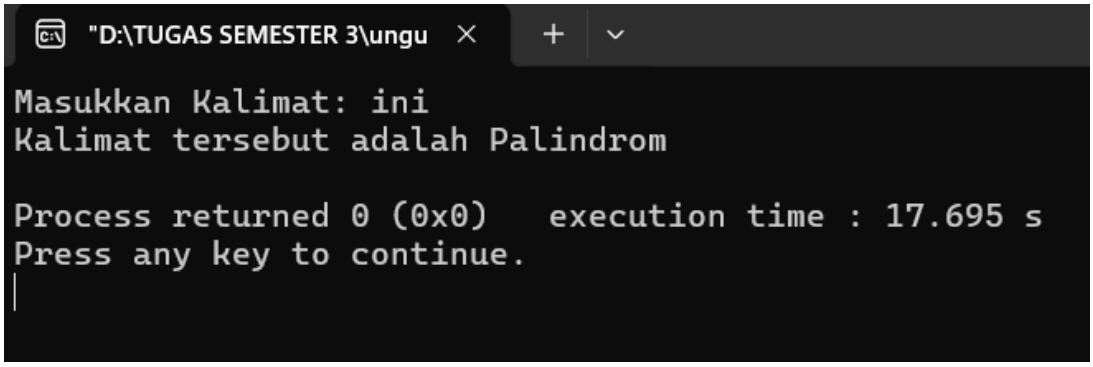
Tersebut adalah palindrom

Maka akan menghasilkan Output

Jika ketik "ini", output akan menjadi: Kalimat tersebut adalah Palindrom

Jika kwtik "telkom", output akan menjadi: Kalimat tersebut adalah bukan

palindrom



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?

   Contoh
   Kalimat : Telkom Purwokerto
   Hasil : otrekowruP mokleT

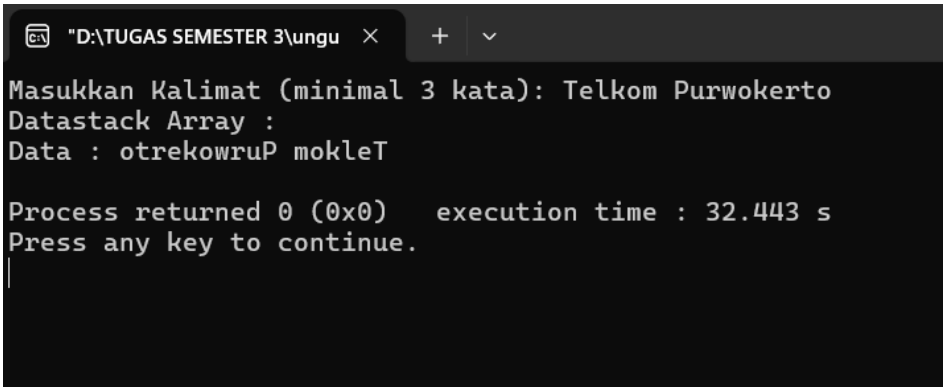
   JAWABAN
   Kode Program

main.cpp | main.cpp | main.cpp | main.cpp

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

void balikKalimat(string kalimat) {
    stack<char> s;

    cout << "Datastack Array :" << endl;

    for (char ch : kalimat) {
        s.push(ch);
    }

    cout << "Data : ";

    while (!s.empty()) {
        cout << s.top();
        s.pop();
    }
    cout << endl;
}

int main() {
    string kalimat;
    cout << "Masukkan Kalimat (minimal 3 kata): ";
    getline(cin, kalimat);

    balikKalimat(kalimat);

    return 0;
}
```

Memasukkan Karakter ke Stack: Setiap karakter dari kalimat yang diinput oleh pengguna akan dimasukkan ke dalam stack satu per satu.
Membentuk Kalimat Terbalik: Karakter dikeluarkan dari stack satu per satu dan dicetak untuk membentuk kalimat terbalik.

Maka akan menghasilkan output
Jika ketik “Telkom Purwokerto” output akan muncul seperti dibawah



5. **Kesimpulan**

   Stack adalah struktur data yang mengikuti prinsip LIFO (Last In First Out), di mana elemen terakhir yang masuk akan menjadi elemen pertama yang keluar. Dalam praktikum ini, stack telah digunakan untuk dua aplikasi sederhana: memeriksa apakah sebuah kalimat adalah palindrom dan membalikkan urutan kalimat. Keduanya menunjukkan bagaimana stack dapat mempermudah manipulasi data dengan cara yang efisien dan sederhana.