LAPORAN

MODUL 6

STACK

Disusun Oleh :

NAMA :

Devi Nurliana

NIM :

103112400144

Dosen :

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. DASAR TEORI

Stack adalah salah satu jenis struktur data linear yang bekerja berdasarkan prinsip *LIFO*, yaitu data yang terakhir dimasukkan akan menjadi data pertama yang dikeluarkan. Konsep ini mirip seperti tumpukan benda, di mana penambahan dan pengambilan data hanya bisa dilakukan di bagian atas, yang disebut TOP. Karena itu, cara mengakses data dalam *stack* terbatas dan harus dikontrol dengan baik. Stack bisa dibuat menggunakan pointer atau *array*. Jika menggunakan pointer, setiap elemen disimpan dalam sebuah node yang saling terhubung. Operasi utamanya adalah *push* untuk menambahkan data ke bagian atas stack dan pop untuk mengambil data teratas. Metode ini memudahkan pengelolaan memori secara dinamis dan cocok digunakan ketika jumlah data bisa berubah-ubah.

B. GUIDED

1. Guide 1
   a. Source Code
      - stack.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;
    }

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

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

    cout << "NULL" << endl;
}

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;
    show(stack);

    cout << "menampilkan sisa stack: " << pop(stack) << endl;
    show(stack);

    return 0;
}
```

b. Screenshot Output

```
Hello world!
TOP -> 9 2 4 3
balik stack
TOP -> 3 4 2 9
```

c. Deskripsi

Program ini berfungsi untuk mengimplementasikan struktur data *Stack* menggunakan *linked list,* dengan langkah kerja sebagai berikut:

1. Pendefinisian struktur dan fungsi dasar
   Program mendefinisikan struktur Node yang berisi data bertipe integer dan pointer *next* untuk menunjuk node berikutnya. Selain itu, dibuat fungsi *isEmpty* untuk mengecek apakah stack kosong.

2. Proses *push* data
   Program menggunakan fungsi push untuk menambahkan data ke dalam *stack*. Setiap data baru dimasukkan ke bagian paling atas (*top*) *stack* dengan cara menghubungkan node baru ke node sebelumnya.

3. Proses pop dan penampilan data
   Program menggunakan fungsi pop untuk menghapus dan mengambil data teratas dari *stack*. Fungsi *show* digunakan untuk menampilkan seluruh isi *stack* dari posisi TOP hingga NULL.

4. Selesai
   Program menampilkan isi *stack*, melakukan penghapusan data, menampilkan kembali sisa *stack*, kemudian program berakhir dan mengembalikan nilai 0.

C. UNGUIDED

1. Unguided 1

a. Source Code

- stack.h

```
#ifndef STACK_H
#define STACK_H

#define MAX_STACK 20

typedef int infotype;

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

void createStack(Stack &S);
void push(Stack &S, infotype x);
```

```
    infotype pop(Stack &S);
    void printInfo(Stack S);
    void balikStack(Stack &S);

    void pushAscending(Stack &S, infotype x);
    void getInputStream(Stack &S);

#endif
```

- stack.cpp

```
    #include <iostream>
    #include "stack.h"

    using namespace std;

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

    void push(Stack &S, infotype x) {
       if (S.top < MAX_STACK - 1) {
          S.top++;
          S.info[S.top] = x;
       } else {
          cout << "Stack penuh!" << endl;
       }
    }

    infotype pop(Stack &S) {
       if (S.top >= 0) {
          infotype x = S.info[S.top];
          S.top--;
          return x;
       } else {
          cout << "Stack kosong!" << endl;
          return 0;
       }
    }

    void printInfo(Stack S) {
       if (S.top == -1) {
          cout << "Stack kosong" << endl;
       } else {
          cout << "TOP -> ";
          for (int i = S.top; i >= 0; i--) {
             cout << S.info[i] << " ";
          }
          cout << endl;
       }
    }
```

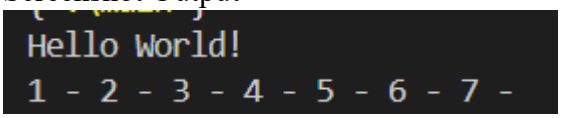
```
void balikStack(Stack &S) {
    int i = 0;
    int j = S.top;
    while (i < j) {
        infotype temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}

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

    while (S.top != -1 && S.info[S.top] > x) {
        push(temp, pop(S));
    }

    push(S, x);

    while (temp.top != -1) {
        push(S, pop(temp));
    }
}

void getInputStream(Stack &S) {
    cout << "Masukkan input (ENTER untuk selesai): ";
    char c;
    while ((c = cin.get()) != '\n') {
        push(S, c);
    }
}
```

- main.cpp

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


int main() {
cout << "Hello world!" << endl;
Stack S;
createStack(S);


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);


    return 0;
}
```

b. Screenshot Output



c. Deskripsi

Program ini bertujuan untuk mengimplementasikan struktur data *Stack* menggunakan *array* dengan kapasitas maksimum 20 elemen serta menerapkan operasi dasar dan tambahan pada *Stack*.

1. Pendefinisian *Stack*
   Program mendefinisikan struktur *Stack* yang terdiri dari array info dan variabel top, serta menyediakan operasi dasar *seperti createStack, push, pop,* dan *printInfo*.

2. Proses *push* dan *pop*
   Data dimasukkan ke dalam stack menggunakan prosedur push dan dikeluarkan menggunakan fungsi *pop* sesuai dengan prinsip *Last In First Out* (LIFO).

3. Operasi tambahan *Stack*
   Program menyediakan prosedur balikStack untuk membalik urutan data dalam *stack, pushAscending* untuk memasukkan data secara terurut, serta *getInputStream* untuk menerima input dari pengguna.

4. Penampilan hasil
   Program menampilkan isi *stack* sebelum dan sesudah dilakukan pembalikan, kemudian program diakhiri.

2. Unguided 2
   a. Source Code
      - stack.h

```
#ifndef STACK_H
#define STACK_H

#define MAX_STACK 20

typedef int infotype;

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

void createStack(Stack &S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);

void pushAscending(Stack &S, infotype x);
void getInputStream(Stack &S);

#endif
```

      - stack.cpp

```
#include <iostream>
#include "stack.h"

using namespace std;

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

void push(Stack &S, infotype x) {
    if (S.top < MAX_STACK - 1) {
        S.top++;
        S.info[S.top] = x;
    } else {
        cout << "Stack penuh!" << endl;
    }
}

infotype pop(Stack &S) {
    if (S.top >= 0) {
        infotype x = S.info[S.top];
        S.top--;
        return x;
```

```
    } else {
        cout << "Stack kosong!" << endl;
        return 0;
    }
}

void printInfo(Stack S) {
    if (S.top == -1) {
        cout << "Stack kosong" << endl;
    } else {
        cout << "TOP -> ";
        for (int i = S.top; i >= 0; i--) {
            cout << S.info[i] << " ";
        }
        cout << endl;
    }
}

void balikStack(Stack &S) {
    int i = 0;
    int j = S.top;
    while (i < j) {
        infotype temp = S.info[i];
        S.info[i] = S.info[j];
        S.info[j] = temp;
        i++;
        j--;
    }
}

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

    while (S.top != -1 && S.info[S.top] > x) {
        push(temp, pop(S));
    }

    push(S, x);

    while (temp.top != -1) {
        push(S, pop(temp));
    }
}

void getInputStream(Stack &S) {
    cout << "Masukkan input (ENTER untuk selesai): ";
    char c;
    while ((c = cin.get()) != '\n') {
        push(S, c);
```

```
        }
}
```

- main.cpp

```
#include <iostream>
#include "bstree.h"
#include "bstree.cpp"

using namespace std;

int main() {
    cout << "Hello World!" << endl;

    address root = Nil;
    insertNode(root, 1);
    insertNode(root, 2);
    insertNode(root, 6);
    insertNode(root, 4);
    insertNode(root, 5);
    insertNode(root, 3);
    insertNode(root, 6);
    insertNode(root, 7);

    InOrder(root);
    cout << endl;
    cout << "kedalaman : " << hitungKedalaman(root, 0) <<
endl;
    cout << "jumlah node : " << hitungJumlahNode(root) <<
endl;
    cout << "total : " << hitungTotalInfo(root) << endl;

    return 0;
}
```

b. Screenshot Output

```
TOP -> 9 8 4 3 3 2
balik stack
TOP -> 2 3 3 4 8 9
```

c. Deskripsi

Program ini bertujuan untuk mengimplementasikan struktur data *Stack* menggunakan *array* dengan kapasitas maksimum 20 elemen serta menyediakan operasi dasar dan lanjutan pada *Stack.*

1. Pendefinisian *Stack*
   Program mendefinisikan struktur *Stack* yang terdiri dari array info dan variabel top, serta prosedur *createStack* untuk menginisialisasi *stack* dalam kondisi kosong.

2. Operasi dasar *Stack*
   Prosedur push digunakan untuk menambahkan data ke stack dan fungsi pop digunakan untuk mengambil data teratas sesuai prinsip Last In First Out (LIFO).

3. Operasi tambahan *Stack*
   Program menyediakan prosedur balikStack untuk membalik urutan elemen, *pushAscending* untuk memasukkan data secara terurut menaik, serta getInputStream untuk menerima input karakter dari pengguna hingga tombol Enter ditekan.

4. Penampilan data
   Prosedur printInfo digunakan untuk menampilkan isi *stack* dari posisi TOP ke bawah sehingga kondisi stack dapat diamati dengan jelas.

3. Unguided 3
   a. Source Code
      - stack.h

```
#ifndef STACK_H
#define STACK_H

#define MAX_STACK 20

typedef int infotype;

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

void createStack(Stack &S);
void push(Stack &S, infotype x);
infotype pop(Stack &S);
void printInfo(Stack S);
void balikStack(Stack &S);
void getInputStream(Stack &S);

#endif
```

      - stack.cpp

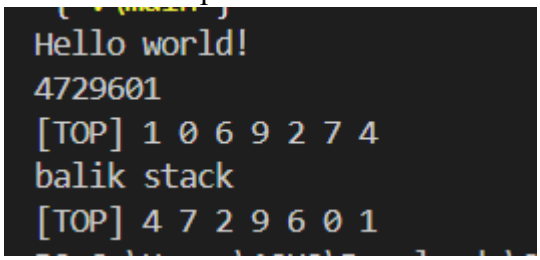
```
#include <iostream>
#include "stack.h"

using namespace std;

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

```
    }

    void push(Stack &S, infotype x) {
        if (S.top < MAX_STACK - 1) {
            S.top++;
            S.info[S.top] = x;
        }
    }

    infotype pop(Stack &S) {
        if (S.top >= 0) {
            infotype x = S.info[S.top];
            S.top--;
            return x;
        }
        return 0;
    }

    void printInfo(Stack S) {
        if (S.top == -1) {
            cout << "Stack kosong" << endl;
        } else {
            cout << "[TOP] ";
            for (int i = S.top; i >= 0; i--) {
                cout << S.info[i] << " ";
            }
            cout << endl;
        }
    }

    void balikStack(Stack &S) {
        int i = 0;
        int j = S.top;
        while (i < j) {
            infotype temp = S.info[i];
            S.info[i] = S.info[j];
            S.info[j] = temp;
            i++;
            j--;
        }
    }

    void getInputStream(Stack &S) {
        char c;
        while ((c = cin.get()) != '\n') {
            if (c >= '0' && c <= '9') {
                push(S, c - '0'); // ubah char ke integer
            }
        }
}
```

- main.cpp

```
#include <iostream>
#include "stack.h"
#include "stack.cpp"

using namespace std;

int main() {
   cout << "Hello world!" << endl;

   Stack S;
   createStack(S);

   getInputStream(S);
   printInfo(S);

   cout << "balik stack" << endl;
   balikStack(S);
   printInfo(S);

   return 0;
}
```

b. Screenshot Output



c. Deskripsi

Program ini bertujuan untuk mengimplementasikan struktur data Stack menggunakan array dengan kemampuan menerima input langsung dari pengguna dan membalik urutan data dalam *stack.*

1. Pendefinisian Stack
   Program mendefinisikan struktur Stack yang terdiri dari array info sebagai tempat penyimpanan data dan variabel top sebagai penanda elemen teratas, serta prosedur *createStack* untuk inisialisasi *stack* kosong.

2. Operasi dasar *Stack*
   Operasi push digunakan untuk menambahkan data ke stack, sedangkan fungsi pop digunakan untuk mengambil data teratas sesuai prinsip *Last In First Out* (LIFO).

3. Input data dan pembalikan *Stack*
   Prosedur getInputStream membaca input angka dari pengguna hingga tombol Enter ditekan dan memasukkannya ke dalam *stack*. Prosedur balikStack digunakan untuk membalik urutan elemen dalam *stack.*

4. Penampilan data
   Prosedur printInfo menampilkan isi *stack* dari posisi TOP ke bawah sebelum dan sesudah proses pembalikan untuk melihat perubahan susunan data.

D. KESIMPULAN

Berdasarkan hasil penerapan pada ketiga program, struktur data *Stack* yang menggunakan *array* berhasil diterapkan dengan baik dengan kapasitas maksimum 20 elemen. Program mampu menjalankan operasi dasar Stack seperti *createStack, push, pop*, dan *printInfo* sesuai dengan prinsip *Last In First Out* (LIFO). Selain itu, juga ditambahkan beberapa operasi tambahan seperti balikStack untuk membalik urutan elemen, *pushAscending* untuk menambahkan data secara terurut, serta *getInputStream* untuk menerima masukan langsung dari pengguna. Dengan adanya operasi-operasi tersebut, program menjadi lebih fleksibel dan membantu dalam memahami konsep *Stack* secara lebih mendalam, baik dalam hal pengelolaan data, proses input, maupun manipulasi urutan elemen.

E. REFERNSI

- Modul Praktikum Struktur Data. *Modul 7: Stack*. Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Telkom, Tahun 2025.
- Kadir, Abdul. *Dasar Pemrograman C++*. Andi Offset, Yogyakarta.
- Weiss, Mark Allen. *Data Structures and Algorithm Analysis in C++*. Pearson Education.