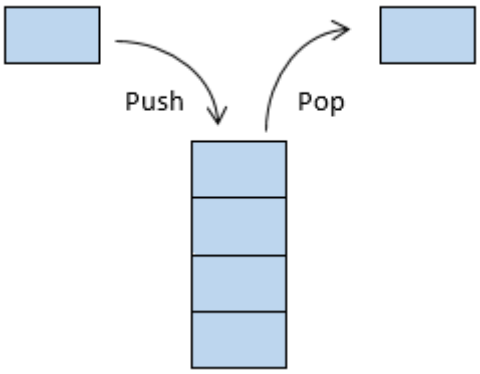
# MODUL 6

# STACK

## DASAR TEORI

### Pengertian Stack

***Stack (Tumpukan)*** merupakan sebuah kumpulan data yang diletakkan atau disusun di atas data lainnya. Stack menerapkan prinsip **LIFO** (*Last In First Out*) yakni elemen yang terakhir disimpan (masuk) dalam stack, akan menjadi elemen yang pertama diambil (keluar). Untuk meletakkan sebuah elemen pada bagian atas (**top**) dari stack, maka dilakukan operasi **Push**. Sedangkan untuk memindahkan sebuah elemen teratas dalam stack, maka dilakukan operasi **Pop**.




*Gambar 1 Ilustrasi sebuah stack*

### Operasi Dasar Pada Stack

- **Create**

  Operasi untuk membuat sebuah stack kosong.

```
struct Stack {
    int top;
    float data[n];
}tumpukan;
```

- **initStack**

  Operasi untuk menginisialisasikan nilai awal stack (stack kosong).

```
bool initStack(){
    Tumpukan.top = -1
}
```

- **isEmpty**

  Operasi untuk memeriksa apakah suatu stack masih kosong. . Stack kosong ditandai dengan nilai Top kurang dari nol (-1).

```
int isEmpty()
{
      if (Tumpukan.top == -1)
            return 1; //true
      else
            return 0; //false
}
```

- **isFull**

  Operasi untuk memeriksa apakah stack yang ada sudah penuh. Stack akan mengindikasikan penuh jika puncak stack (top) terletak tepat di bawah jumlah maksimum (MAX) yang dapat ditampung stack.

```
int isFull()
{
      if (Tumpukan.top == MAX-1)
            return 1; //true
      else
            return 0; //false
}
```

- **Push (Insert)**

  Operasi untuk menambahkan satu elemen ke dalam stack dan tidak dapat dilakukan jika stack dalam keadaan penuh.

```
void push (int data)
{
      if (isEmpty() == 1){
            Tumpukan.top++;
            Tumpukan.data[Tumpukan.top] = Data;
      }
      else if (isFull() == 0){
            Tumpukan.top++;
            Tumpukan.data[Tumpukan.top] = Data;
      }
      else{
            cout << "Stack sudah penuh!";
      }
}
```

- **Pop**

  Operasi untuk mengambil atau menghapus data teratas (top) dari stack.

```
void pop()
{
    if (isEmpty() == 0){
        myStack.top--;
        cout << " Data teratas terambil" << endl;
    }
    else{
        cout << " Stack masih kosong!" << endl;
    }
}
```

- **Clear**

  Operasi untuk menghapus atau mengosongkan seluruh data stack. Jika Top bernilai -1 maka stack dianggap kosong.

```
void clear()
{
      top = -1;
}
```

- **Display**

  Operasi untuk menampilkan seluruh data stack.

```
void display()
{
      if (isEmpty() == 0){
          for (int i=Tumpukan.top; i>=0; i--){
              cout << Tumpukan.data[i] << endl;
          }
      }
      else{
          cout << "Stack masih kosong!" << endl;
      }
}
```

**GUIDED**

**Program Stack 1**

```
#include <iostream>
#define MAX 5   //menetapkan nilai konstanta MAX = 5 (ukuran stack)

using namespace std;

///PROGRAM STACK DENGAN STRUCTURE

//Deklarasi Stack
struct Stack{
    int top;
    int data[MAX];  //array dengan ukuran MAX (5)
}myStack;
```

```
void initStack()    //inisialisasi stack
{
    myStack.top = -1;
}

int isEmpty()   //mengecek apakah tumpukan kosong
{
    if (myStack.top == -1)
        return 1;
    else
        return 0;
}

int isFull()    //mengecek apakah tumpukan penuh
{
    if (myStack.top == MAX - 1)
        return 1;
    else
        return 0;
}

//Insert Data (Push)
void push(int data)
{
    if (isEmpty() == 1){
        myStack.top++;
        myStack.data[myStack.top] = data;
        cout << " Data " << data << " telah dimasukkan" << endl;
    }
    else if (isFull() == 0){
        myStack.top++;
        myStack.data[myStack.top] = data;
        cout << " Data " << data << " telah dimasukkan" << endl;
    }
    else{
        cout << " Stack penuh!" << endl;
    }
}

//Hapus Data (Pop)
void pop()
{
    if (isEmpty() == 0){
        myStack.top--;
        cout << " Data teratas terambil" << endl;
    }
```

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

//Tampilkan Top
void top()
{
    if (isEmpty() == 0){
        cout << " " << myStack.data[myStack.top] << endl;
    }
      else{
        cout << " Stack masih kosong!" << endl;
      }
}

//Tampilkan Stack
void display()
{
    if (isEmpty() == 0){
        for (int i = myStack.top; i>=0; i--){
            cout << " " << myStack.data[i] << endl;
        }
    }
    else{
        cout << " Stack masih kosong!" << endl;
    }
}

//Hapus Stack
void clear()
{
      myStack.top = -1;
      cout << " Stack berhasil dihapus" << endl;
}

int main()
{
    int pilihan, nilai;

    initStack();

    do{
        system("cls");
        cout << "=== Menu ===" << endl;
        cout << " 1. Memasukkan data (Push)" << endl;
        cout << " 2. Menghapus data (Pop)" << endl;
```

```
        cout << " 3. Menampilkan data teratas (Top)" << endl;
        cout << " 4. Menampilkan data" << endl;
        cout << " 5. Hapus Stack" << endl;
        cout << " 6. Keluar" << endl;
        cout << " Masukkan pilihan: ";
        cin >> pilihan;
        cout << endl;

        if (pilihan==1){
            cout << " Masukkan data: ";
            cin >> nilai;
            push(nilai);
        }
        else if (pilihan==2){
            pop();
        }
        else if (pilihan==3){
            top();
        }
        else if (pilihan==4){
            display();
        }
        else if (pilihan==5){
            clear();
        }
        else{
            return 0;
        }
        cout << endl;
        system("pause");
    }
    while (pilihan != 6);

    return 0;
}
```

**Program Stack Dengan Array**

```
#include <iostream>

using namespace std;

///PROGRAM STACK DENGAN ARRAY
```

```
//Deklarasi Global
const int n = 5; //konstanta ukuran array
int stack[n];
int top = -1;

int isEmpty()   //mengecek apakah tumpukan kosong
{
    if (top == -1)
        return 1;
    else
        return 0;
}

int isFull()    //mengecek apakah tumpukan penuh
{
    if (top == n-1)
        return 1;
    else
        return 0;
}

//Insert Data (Push)
void push(int data)
{
    if (isEmpty() == 1){
        top++;
        stack[top] = data;
        cout << " Data " << data << " telah dimasukkan" << endl;
    }
    else if (isFull() == 0){
        top++;
        stack[top] = data;
        cout << " Data " << data << " telah dimasukkan" << endl;
    }
    else{
        cout << " Stack penuh!" << endl;
    }
}

//Hapus Data (Pop)
void pop()
{
    if (isEmpty() == 0){
        top--;
        cout << " Data teratas terambil" << endl;
    }
```

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

//Tampilkan Stack
void display()
{
    if (isEmpty() == 0){
        for (int i = top; i>=0; i--){
            cout << " " << stack[i] << endl;
        }
    }
    else{
        cout << " Stack masih kosong!" << endl;
    }
}

//Hapus Stack
void clear()
{
      top = -1;
      cout << " Stack berhasil dihapus" << endl;
}

int main()
{
    int pilihan, nilai;

    do{
        system("cls");
        cout << "=== Menu ===" << endl;
        cout << " 1. Memasukkan data (Push)" << endl;
        cout << " 2. Menghapus data (Pop)" << endl;
        cout << " 3. Menampilkan data" << endl;
        cout << " 4. Hapus Stack" << endl;
        cout << " 5. Keluar" << endl;
        cout << " Masukkan pilihan: ";
        cin >> pilihan;
        cout << endl;

        if (pilihan==1){
            cout << " Masukkan data: ";
            cin >> nilai;
            push(nilai);
        }
        else if (pilihan==2){
```

```
            pop();
        }
        else if (pilihan==3){
            display();
        }
        else if (pilihan==4){
            clear();
        }
        else{
            return 0;
        }
        cout << endl;
        system("pause");
    }
    while (pilihan != 5);

    return 0;
}
```

**Program Stack Dengan Single Linked List**

```
#include <iostream>

using namespace std;

///PROGRAM STACK DENGAN SINGLE LINKED LIST

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

Node *top = NULL;

int isEmpty()
{
    if(top == NULL)
        return 1;
    else
        return 0;
}

//Insert Data (Push)
```

```
void push (int nilai)
{
    Node *ptr = new Node();
    ptr->data = nilai;
    ptr->link = top;
    top = ptr;

    cout << " Data " << nilai << " telah dimasukkan" << endl;
}

//Hapus Data (Pop)
void pop ()
{
    if (isEmpty() == 1)
        cout << " Stack masih kosong!" << endl;
    else
    {
        Node *ptr = top;
        top = top->link;
        cout << " Data " << ptr->data << " terambil" << endl;
        delete(ptr);
    }
}

//Tampilkan data teratas (Top)
void showTop()
{
    if (isEmpty() == 1)
    cout << " Stack masih kosong!" << endl;
    else
    cout << " Data teratas (top) : " << top->data << endl;
}

//Hapus Stack
void clearStack()
{
    Node *bantu, *hapus;
    bantu = top;

    while (bantu != NULL){
        hapus = bantu;
        bantu = bantu->link;
        delete hapus;
    }
    top = NULL;
    cout << "Stack berhasil terhapus!" << endl;
}
```

```
//Tampilkan Stack
void displayStack()
{
     if (isEmpty() == 1)
        cout << " Stack masih kosong!" << endl;
     else
     {
        Node *temp = top;
        while(temp != NULL){
            cout << " " << temp->data << " ";
            temp = temp->link;
        }
        cout << endl;
     }
 }


int main()
{
    int pilihan, nilai;

    do{
        system("cls");
        cout << "=== Menu ===" << endl;
        cout << " 1. Memasukkan data (Push)" << endl;
        cout << " 2. Menghapus data (Pop)" << endl;
        cout << " 3. Menampilkan data teratas (Top)" << endl;
        cout << " 4. Menampilkan data" << endl;
        cout << " 5. Hapus Stack" << endl;
        cout << " 6. Keluar" << endl;
        cout << " Masukkan pilihan: ";
        cin >> pilihan;
        cout << endl;

        if (pilihan==1){
            cout << " Masukkan data: ";
            cin >> nilai;
            push(nilai);
        }
        else if (pilihan==2){
            pop();
        }
        else if (pilihan==3){
            showTop();
        }
        else if (pilihan==4){
```

```
            displayStack();
        }
        else if (pilihan==5){
            clearStack();
        }
        else{
            return 0;
        }
        cout << endl;
        system("pause");
    }
    while (pilihan != 6);

    return 0;
}
```

**TUGAS**

1. Buatlah program untuk melakukan pembalikan (*reversing*) terhadap kalimat dengan menggunakan stack!

   Contoh:

   Kalimat: **Struktur Data**

   Hasil setelah dibalik: **ataD rutkurtS**

2. Buatlah program untuk melakukan konversi notasi infix ke notasi postfix dengan menggunakan stack!

   Misalnya:

| **Notasi Infix** | **Notasi Postfix** |
|---|---|
| A + B – C | A B + C - |
| (A + B) * (C – D) | A B + C D - * |

~ SELAMAT MENGERJAKAN 😊 ~