ASESMEN STRUKTUR DATA – 18/11/25

IF-12-07

Nama : Keishin Naufa Alfaridzhi

NIM : 103112400061

I. SOURCE CODE

1. **Soal 1:**

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

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

Node* head = nullptr;

Node* createNode(string data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

void insertFront(string data) {
    Node* newNode = createNode(data);
    newNode->next = head;
    head = newNode;
    cout << "Data inserted di depan: " << data << endl;
}

void insertBehind(string data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
```

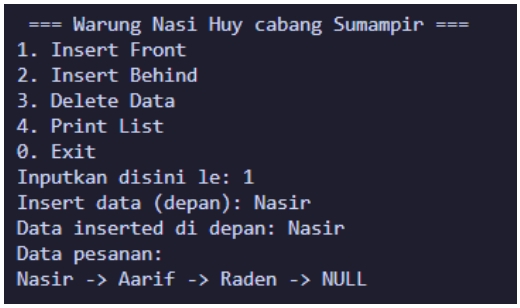
```
void deleteNode(string data) {
    if (head == nullptr) {
        cout << "List empty.\n";
        return;
    }

    Node* temp = head;
    Node* prev = nullptr;

    if (temp != nullptr && temp->data == data) {
        head = temp->next;
        delete temp;
        cout << "Data " << data << " deleted successfully\n";
        return;
    }

    while (temp != nullptr && temp->data != data) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "No data.\n";
        return;
    }

    prev->next = temp->next;
    delete temp;
    cout << "Data " << data << " deleted successfully\n";
}

void printInfo() {
    if (head == nullptr) {
        cout << "No data.\n";
        return;
    }

    Node* temp = head;
    cout << "Data pesanan:\n";
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}


int main() {
    int input;
    string data;
```

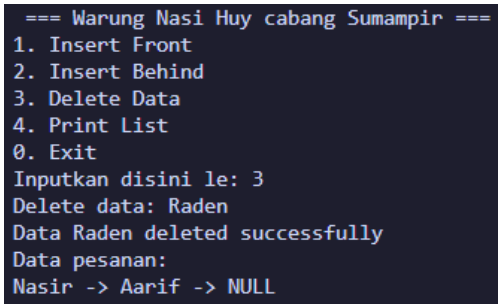
```
    do {
        cout << "\n === Warung Nasi Huy cabang Sumampir ===\n";
        cout << "1. Insert Front\n";
        cout << "2. Insert Behind\n";
        cout << "3. Delete Data\n";
        cout << "4. Print List\n";
        cout << "0. Exit\n";
        cout << "Inputkan disini le: ";
        cin >> input;

        switch (input) {
            case 1:
                cout << "Insert data (depan): ";
                cin >> data;
                insertFront(data);
                printInfo();
                break;
            case 2:
                cout << "Insert data (balakang): ";
                cin >> data;
                insertBehind(data);
                printInfo();
                break;
            case 3:
                cout << "Delete data: ";
                cin >> data;
                deleteNode(data);
                printInfo();
                break;
            case 4:
                printInfo();
                break;
            default:
                cout << "Input invalid.\n";
        }
    } while (input != 0);
}
```

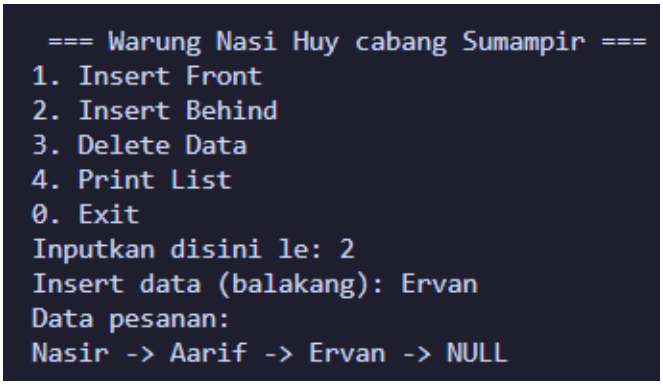
Output & Langkah:

1.
```
 === Warung Nasi Huy cabang Sumampir ===
1. Insert Front
2. Insert Behind
3. Delete Data
4. Print List
0. Exit
Inputkan disini le: 1
Insert data (depan): Aarif
Data inserted di depan: Aarif
Data pesanan:
Aarif -> NULL
```

2.
```
 === Warung Nasi Huy cabang Sumampir ===
1. Insert Front
2. Insert Behind
3. Delete Data
4. Print List
0. Exit
Inputkan disini le: 2
Insert data (balakang): Raden
Data pesanan:
Aarif -> Raden -> NULL
```

3. 

4. 

5. 

← Output akhir

Penjelasan:

Program antrian warung menggunakan singly linked-list. Dapat melakukan insert pada belakang atau depan antrian.

- InsertFront(): insert paling depan antrian, dengan cara memundurkan antrian lainnya sebanyak 1 kali, lalu masukkan antrian baru pada head.
- InsertBehind(): insert paling belakang antrian, dengan cara iterasi dari head hingga antrian akhir, lalu masukkan antrian baru pada next (setelah antrian akhir).
- deleteData(): Menghapus antrian paling depan (head) lalu menggeser antrian lainnya ke depan sebanyak 1 kali.

2. **Soal 2:**

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

struct Stack {
    string info[20];
    int top;
};

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

void push(Stack &S, string x) {
    if (S.top < 19) {
        S.top++;
        S.info[S.top] = x;
    } else {
```

```
        cout << "Stack full.\n";
    }
}

string pop(Stack &S) {
    if (S.top >= 0) {
        string x = S.info[S.top];
        S.top--;
        return x;
    } else {
        cout << "Stack empty.\n";
        return " ";
    }
}

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

int main() {
    Stack S;
    Stack meja;
    CreateStack(S);

    push(S, "K");
    push(S, "E");
    push(S, "I");
    push(S, "S");
    push(S, "H");
    push(S, "I");
    push(S, "N");
    cout << "Stack S:";
    printInfo(S);

    push(meja, pop(S));
    push(meja, pop(S));
    push(meja, pop(S));
    push(meja, pop(S));
    push(meja, pop(S));
    push(meja, pop(S));
    push(meja, pop(S));
```

```
    cout << "Stack meja:";
    printInfo(meja);
    cout << "Stack S:";
    printInfo(S);
}
```

Output:

```
d:\Tugas\SEM3\StrukturData\Praktikum\others\assesment\soal2>cd
ktikum\others\assesment\soal2\"main
Stack S:[TOP] N I H S I E K
Stack meja:[TOP] K E I S H I N
Stack S:Stack empty.
```

Penjelasan:

Program puzzle stack. Memasukkan (push) huruf secara terurut kemudian mengambilnya (pop) kembali untuk menyusun kata sebenarnya. Dilakukan dengan syntax “push(meja, pop(S))”.

- push(): Memasukkan huruf ke dalam stack, first in last out.
- pop(): Mengambil huruf paling atas (top).

3. Soal 3:

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

const int MAX = 10;

struct Queue {
    string info[MAX];
    int head, tail;
};

void CreateQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(const Queue &Q) {
    return (Q.head == -1 && Q.tail == -1);
}

bool isFullQueue(const Queue &Q) {
    return (Q.tail + 1) % MAX == Q.head;
}

void enqueue(Queue &Q, string x) {
```

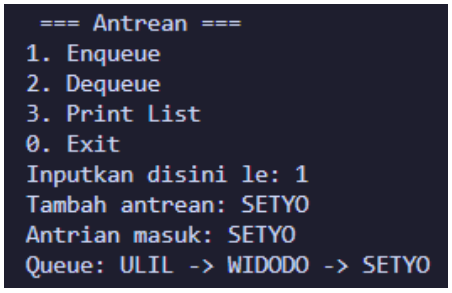
```
    if (isFullQueue(Q)) {
        cout << "Queue full.\n";
    } else {
        if (isEmptyQueue(Q)) {
            Q.head = Q.tail = 0;
        } else {
            if (Q.tail == MAX - 1) {
                Q.tail = 0;
            } else {
                Q.tail++;
            }
        }
        Q.info[Q.tail] = x;
        cout << "Antrian masuk: " << x << endl;
    }
}

string dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue empty.\n";
    } else {
        cout << "Dequeue: " << Q.info[Q.head] << endl;
        if (Q.head == Q.tail) {
            Q.head = Q.tail -1;
        } else
        if (Q.head == MAX - 1) {
            Q.head = 0;
        } else {
            Q.head++;
        }
    }
}

void printInfo(const Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue empty.\n";
    } else {
        cout << "Queue: ";
        int i = Q.head;

        while (i != Q.tail) {
            cout << Q.info[i] << " -> ";
            i = (i + 1) % MAX;
        }

        cout << Q.info[Q.tail] << " \n";
    }
}

int main() {
```

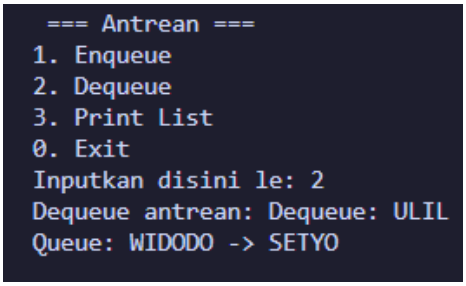
```
    Queue Q;
    CreateQueue(Q);
    int input;
    string data;

    do {
        cout << "\n === Antrean ===\n";
        cout << "1. Enqueue\n";
        cout << "2. Dequeue\n";
        cout << "3. Print List\n";
        cout << "0. Exit\n";
        cout << "Inputkan disini le: ";
        cin >> input;

        switch (input) {
            case 1:
                cout << "Tambah antrean: ";
                cin >> data;
                enqueue(Q, data);
                printInfo(Q);
                break;
            case 2:
                cout << "Dequeue antrean: ";
                dequeue(Q);
                printInfo(Q);
                break;
            case 3:
                printInfo(Q);
                break;
            default:
                cout << "Input invalid.\n";
        }
    } while (input != 0);
}
```

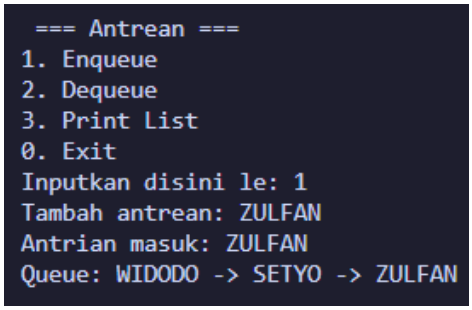
Output & Langkah:

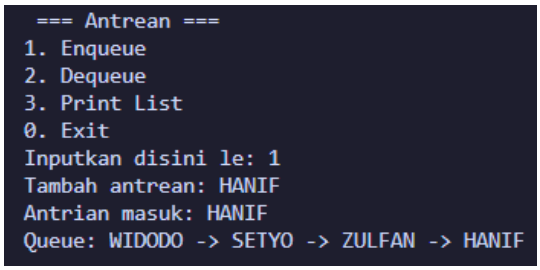
1.
```
 === Antrean ===
1. Enqueue
2. Dequeue
3. Print List
0. Exit
Inputkan disini le: 1
Tambah antrean: ULIL
Antrian masuk: ULIL
Queue: ULIL
```

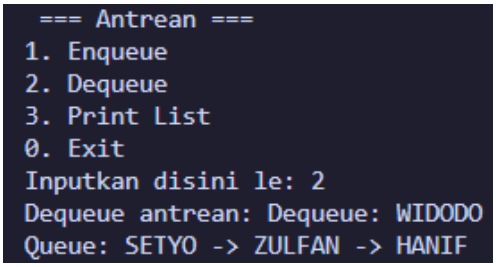
2.
```
 === Antrean ===
1. Enqueue
2. Dequeue
3. Print List
0. Exit
Inputkan disini le: 1
Tambah antrean: WIDODO
Antrian masuk: WIDODO
Queue: ULIL -> WIDODO
```

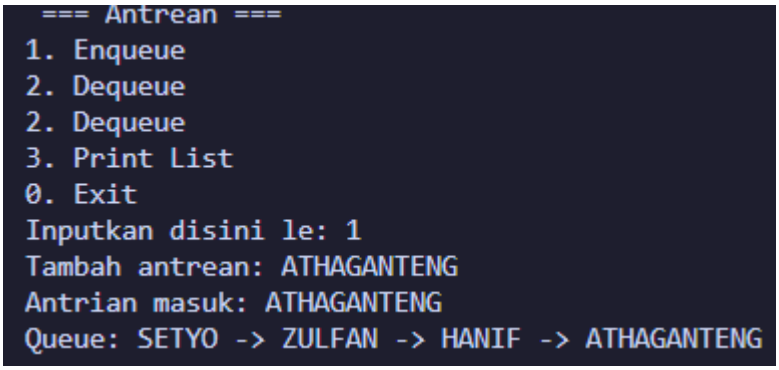
3. 

4. 

5. 

6. 

7. 

8. 

← Output akhir

Penjelasan:

Program antrean apotek dengan circular queue. Antrean dilakukan secara terurut.

- enqueue(): menambahkan antrian pada paling belakang (tail).
- dequeue(): mengambil antrian pada paling depan (head).