

## LAPORAN WORKSHEET 2 - STRUKTUR DATA

Kelompok: RRN

Kelas: 2 E - Teknik Informatika

Anggota:

1. Nama : Naufal Ammar Hidayatulloh  
NPM : 2010631170104
2. Nama: Rifky Al Farezal  
NPM: 2010631170029
3. Nama: Rahmat Randiansyah Siregar  
NPM: 2010631170112

### Tugas 1 Linked List

Linked list adalah suatu struktur data yg merupakan suatu rangkaian atau daftar record berjenis sama. Kemudian dihubungkan melalui bantuan pointer. Pengalokasian daftar dapat dilakukan secara dinamis sehingga isi dari daftar dapat dimanipulasi. struktur berupa rangkaian elemen saling berkait dimana setiap elemen dihubungkan elemen lain melalui pointer. Pointer adalah alamat elemen. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik walau tidak bersebelahan secara fisik di memori.

### Source Code:

```
// Program linkedList
// Nama Kelompok : RRN
// Kelas : 2E-Teknik Informatika

#include <bits/stdc++.h>

using namespace std;

// struktur node linked list
class Node
{
    public:
    int data;
    Node *next;
};
```

```

// menghapus node
void deleteNode(Node *head, Node *n)
{
    // mendefinisikan node sebelum dihapus
    Node *prev = head;
    while(prev->next != NULL && prev->next != n)
        prev = prev->next;

    // menghapus node dari Linked List
    prev->next = prev->next->next;

    // membebaskan memori
    free(n);
}

// fungsi untuk memasukkan isi node
void push(Node **head_ref, int new_data)
{
    Node *new_node = new Node();
    new_node->data = new_data;
    new_node->next = *head_ref;
    *head_ref = new_node;
}

// menampilkan linked list
void printList(Node *head)
{
    while(head!=NULL)
    {
        cout<<head->data<<" ";
        head=head->next;
    }
    cout<<endl;
}

int main()
{
    Node *head = NULL;

```

```

        // memasukkan isi node
        push(&head,5);
        push(&head,4);
        push(&head,3);
        push(&head,2);
        push(&head,1);

        cout<<"Linked List: ";
        printList(head);

        cout<<"\nMenghapus node ke-2 dari akhir list";
        deleteNode(head, head->next->next->next);

        cout<<"\nModified Linked List: ";
        printList(head);
    }
}

```

## Hasil Program:

```

linkedList.cpp - KULIAH - Visual Studio Co...
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
.cpp -o linkedList && "c:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH\Smt
2\strukturData\UAS\teori\linkedList
Linked List: 1 2 3 4 5

Menghapus node ke-2 dari akhir list
Modified Linked List: 1 2 3 5

c:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH\Smt 2\strukturData\UAS\teo
ri>

```

## Tugas 2 Stack

Stack atau tumpukan adalah kumpulan elemen yang hanya dapat ditambah dan atau dihapus dari satu ujung (gerbang) yang sama. Hal ini menunjukkan bahwa seolah-olah suatu elemen diletakkan di atas elemen yang lain. Yang memberi gambaran bahwa Stack mempunyai sifat LIFO (Last In First Out) yang berarti bahwa elemen yang terakhir masuk akan pertama keluar.

### Source Code:

```
// Program stack
// Nama Kelompok : RRN
// Kelas : 2E-Teknik Informatika

#include <bits/stdc++.h>

using namespace std;

// mendefinisikan batas stack
const int ukuranStack = 200;
int stackS = 0, stackT = 0;

// mendefinisikan stack dengan array
array<string, ukuranStack> s;
array<string, ukuranStack> t;

// cek stack penuh atau tidak
bool stackFull(int top)
{
    if (top == ukuranStack)
    {
        // jika stack mencapai batas maksimal maka akan mengembalikan nilai menjadi
        "Stack Penuh"
        return true;
    }
    else
    {
        return false;
    }
}

// menambahkan data ke stack s
void pushStkS(string data)
```

```

{
    // mengecek stack penuh atau tidak
    if (stackFull(stackS))
    {
        cout << "Stack Penuh" << endl;
    }
    else
    {
        s[stackS] = data;
        stackS++;
    }
}

```

```

// menambahkan data ke stack t
void pushStkT(string data)
{
    // mengecek stack penuh atau tidak
    if (stackFull(stackT))
    {
        cout << "Stack Penuh" << endl;
    }
    else
    {
        t[stackT] = data;
        stackT++;
    }
}

```

```

// menghapus tanda "#" dan karakter sebelumnya di stack s
void deleteHashtagS()
{
    for (int i = 0; i < s.size(); i++)
    {
        if (s[i] == "#")
        {
            s[i] = "";
            stackS--;
        }
        else if (s[i] == "#" && s[i + 1] == "#")
        {

```

```

        s[1 + 1] = "";
        s[i] = "";
        s[i - 1] = "";
        s[i - 2] = "";
    }
    else if (s[i] == "#")
    {
        s[i] = "";
        s[i - 1] = "";
        stackS -= 2;
    }
}
}

```

// menghapus tanda "#" dan karakter sebelumnya di stack s

```

void deleteHashtagT()
{
    for (int i = 0; i < t.size(); i++)
    {
        if (t[0] == "#")
        {
            t[0] = "";
            stackT--;
        }
        else if (s[i] == "#" && s[i + 1] == "#")
        {
            s[1 + 1] = "";
            s[i] = "";
            s[i - 1] = "";
            s[i - 2] = "";
        }
        else if (t[i] == "#")
        {
            t[i] = "";
            t[i - 1] = "";
            stackT -= 2;
        }
    }
}
}

```

```

// mengecek konten stack s dan t sama atau tidak
bool stackSama(array<string, ukuranStack> stackA, array<string, ukuranStack> stackB)
{
    for (int i = 0; i < stackA.size(); i++)
    {
        if (stackA[i] != stackB[i])
        {
            return false;
        }
    }
    return true;
}

```

```

// menampilkan data di stack
void tampilkanStack(array<string, ukuranStack> stack)
{
    for (int i = 0; i < stack.size(); i++)
    {
        if (stack[i] != "")
        {
            cout << stack[i];
        }
    }
}

```

```

// mengecek input data lebih dari 1 atau tidak
bool cekJumlahChar(string data)
{
    if (data.size() > 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

```

int main()
{

```

```

string data;

while (true)
{
    system("cls");
    // menampilkan data stack
    cout << "Data Di Stack " << endl
        << "Stack S : ";
    tampilkanStack(s);
    cout << endl
        << "Stack T : ";
    tampilkanStack(t);
    // menu
    cout << endl
        << "-----" << endl
        << "1. Input stack s" << endl
        << "2. Input stack t" << endl
        << "3. Tampilkan hasil" << endl
        << "4. keluar" << endl
        << "Input [1-4] : ";
    cin >> data;
    if (data == "1")
    {
        pushS:
        system("cls");
        cout << "Masukan data ke stack S : ";
        cin >> data;
        if (cekJumlahChar(data))
        {
            cout << "Maaf kamu hanya dapat memasukan 1 character saja\n\n";
            system("pause");
            goto pushS;
        }
        pushStkS(data);
    }
    else if (data == "2")
    {
        pushT:
        system("cls");
        cout << "Masukan data ke stack T : ";
    }
}

```



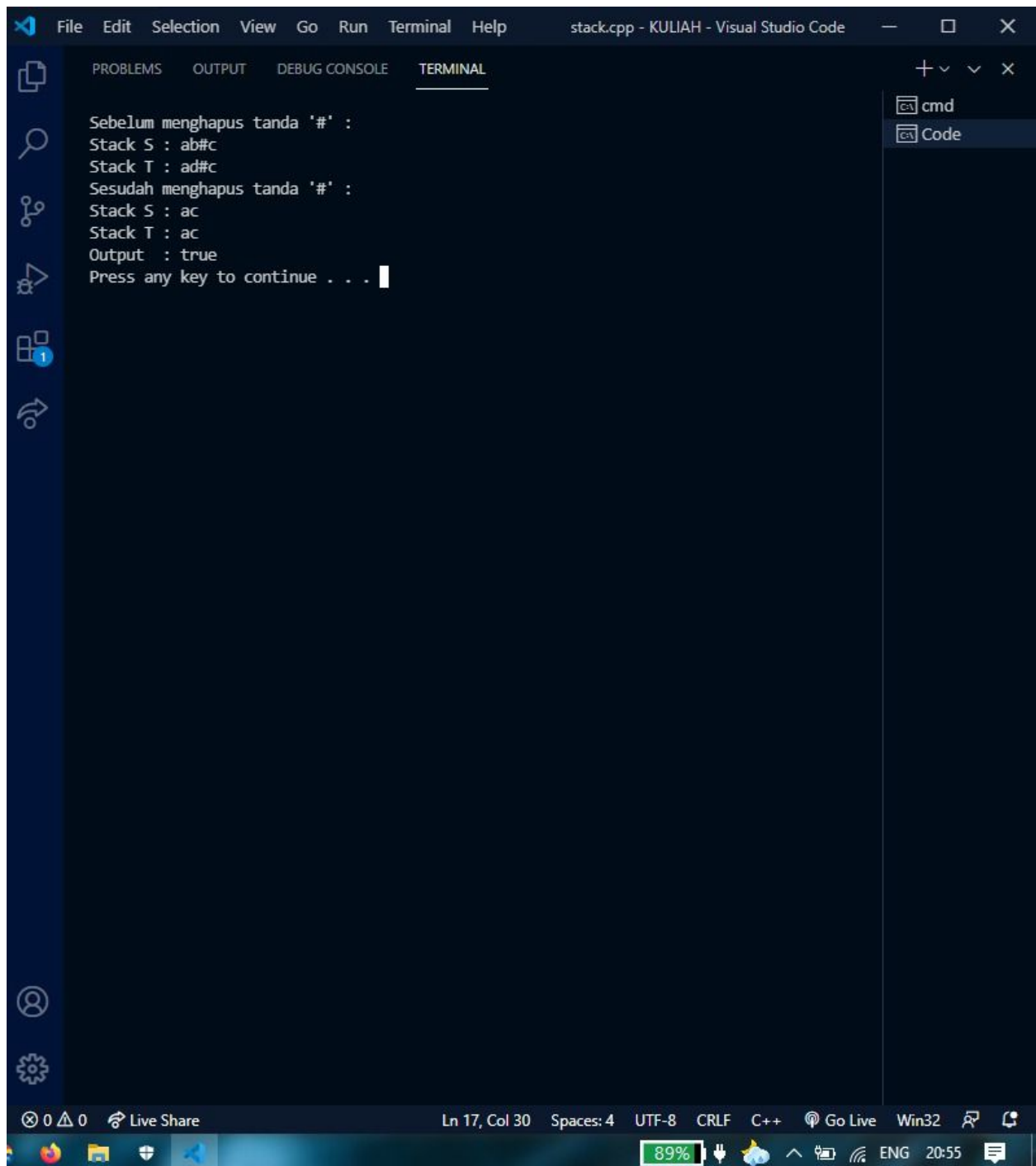
```

cin >> data;
if (cekJumlahChar(data))
{
    cout << "Maaf kamu hanya dapat memasukan 1 character saja\n\n";
    system("pause");
    goto pushT;
}
pushStkT(data);
}
else if (data == "3")
{
    system("cls");
    cout << "Sebelum menghapus tanda '#' : " << endl;
    cout << "Stack S : ";
    tampilkanStack(s);
    cout << endl;
    cout << "Stack T : ";
    tampilkanStack(t);
    cout << endl;
    // menghapus tanda #
    deleteHashtagS();
    deleteHashtagT();
    cout << "Sesudah menghapus tanda '#' : " << endl;
    cout << "Stack S : ";
    tampilkanStack(s);
    cout << endl;
    cout << "Stack T : ";
    tampilkanStack(t);
    cout << endl;
    cout << "Output : " << boolalpha << stackSama(s, t) << endl;
    system("pause");
}
else if (data == "4")
{
    break;
}
else
{
    cout << "Input angka 1 - 4 " << endl;
    system("pause");
}

```

```
}  
}  
}
```

## Hasil Program



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output displays the state of two stacks, S and T, before and after removing a '#' character. The output is as follows:

```
Sebelum menghapus tanda '#' :  
Stack S : ab#c  
Stack T : ad#c  
Sesudah menghapus tanda '#' :  
Stack S : ac  
Stack T : ac  
Output : true  
Press any key to continue . . .
```

The status bar at the bottom indicates the current line and column as 'Ln 17, Col 30', the file encoding as 'UTF-8', and the line endings as 'CRLF'. The system tray shows a battery level of 89% and the time as 20:55.

### Tugas 3 Queue

Queue atau Antrian merupakan kumpulan elemen dengan penyisipan dan penghapusan elemen yang dilakukan dari sisi/gerbang yang berbeda. Penyisipan dilakukan dari gerbang belakang dan penghapusan dilakukan dari gerbang depan. Hal ini menunjukkan bahwa untuk Queue mempunyai dua gerbang yaitu gerbang depan dan gerbang belakang. Dengan demikian dapat dilihat bahwa Queue mempunyai sifat FIFO (First In First Out), yaitu elemen yang pertama masuk akan keluar pertama juga.

#### Source Code:

```
/*
Program Queue dengan Array
Nama Kelompok: RRN
Kelas: 2 E
*/
#include <iostream>
#include <stdio.h>
#include <iomanip>
using namespace std;
//Deklarasi struct antrian
struct Queue
{
int size;
int front;
int rear;
int *Q;
int n=5;
};
void create(struct Queue *q,int size)
{
q->size=size;
q->front=q->rear=-1;
q->Q=(int *)malloc(q->size*sizeof(int));
}
//Menambahkan data ke antrian
void enqueue(struct Queue *q,int x)
{
if(q->rear==q->size-1)
cout<<"Queue is Full";
else
```

```

{
q->rear++;
q->
Q[q->rear]=x;
}
}
//Mengambil data antrian
int dequeue(struct Queue *q)
{
int x=-1;
if(q->front==q->rear)
cout<<"Queue is Empty\n";
else
{

q->front++;
x=q->Q[q->front];
}
return x;
}
//Menampilkan antrian
void Display(struct Queue q)
{
int i;
for(i=q.front+1;i<=q.rear;i++)
cout<<q.Q[i];
cout<<endl;
}
int main()
{
int k=5;
int arr[5]={2,4,6,8,1};
cout<<"Data N:"<<k<<endl;
cout<<"\tArr[] ";
for(int n=0; n<5; n++){
cout<<setw(4)<<arr[n];
}

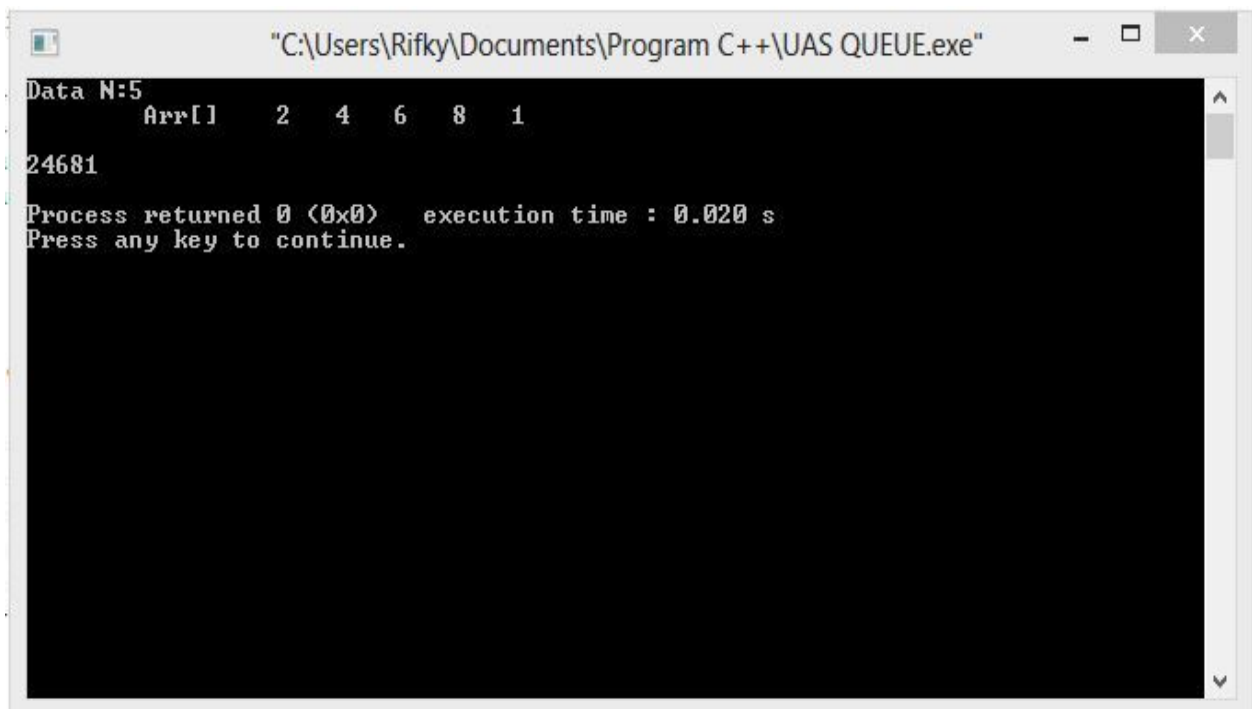
cout<<endl<<endl;
struct Queue q;
create(&q,5);

```

```
        enqueue(&q,2);
        enqueue(&q,4);
        enqueue(&q,6);
        enqueue(&q,8);
        enqueue(&q,1);
        Display(q);

    return 0;
}
```

## Hasil Program



```
"C:\Users\Rifky\Documents\Program C++\UAS QUEUE.exe"
Data N:5
Arr[] 2 4 6 8 1
24681
Process returned 0 (0x0) execution time : 0.020 s
Press any key to continue.
```

## Tugas 4 Tree

Kumpulan node yang saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layaknya struktur sebuah pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (one-to-many) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node-node dari atas ke bawah. Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (one-to-many) dan tidak linier antara elemen-elmennya.

### Source Code:

```
// Program tree
// Nama Kelompok : RRN
// Kelas : 2E-Teknik Informatika

#include <bits/stdc++.h>

using namespace std;

// struktur node tree
struct Node
{
    int data;
    struct Node *left, *right;
    Node(int data)
    {
        this->data = data;
        left = right = NULL;
    }
};

// menentukan postorder
// aturan postorder adalah left, right, root
void printPostorder(struct Node *node)
{
    if (node == NULL)
        return;

    printPostorder(node->left);

    printPostorder(node->right);
```

```

    cout << node->data << " ";
}

// menentukan inorder
// aturan inorder adalah left, root, right
void printInorder(struct Node *node)
{
    if (node == NULL)
        return;

    printInorder(node->left);

    cout << node->data << " ";

    printInorder(node->right);
}

// menentukan preorder
// aturan postorder adalah root, left, right
void printPreorder(struct Node *node)
{
    if (node == NULL)
        return;

    cout << node->data << " ";

    printPreorder(node->left);

    printPreorder(node->right);
}

int main()
{
    // input angka dari node tree
    struct Node *root = new Node(27);
    root->left = new Node(14);
    root->right = new Node(35);
    root->left->left = new Node(10);
    root->left->right = new Node(19);

```

```

root->right->left = new Node(31);
root->right->right = new Node(42);

cout << "\nPreorder traversal\n";
printPreorder(root);

cout << "\nInorder traversal\n";
printInorder(root);

cout << "\nPostorder traversal\n";
printPostorder(root);
}

```

### Hasil Program:

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal displays the output of a C++ program that performs preorder, inorder, and postorder traversals on a binary tree. The command prompt shows the directory path and the compilation command using g++. The output lists the traversal types and their corresponding node values.

```

Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH>cd "c:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH\Smt 2\strukturData\UAS\teori" && g++ tree.cpp -o tree && "c:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH\Smt 2\strukturData\UAS\teori\"tree

Preorder traversal
27 14 10 19 35 31 42
Inorder traversal
10 14 19 27 31 35 42
Postorder traversal
10 19 14 31 42 35 27
c:\Users\Ammar\Documents\Naufal Ammar\Naufal\Ammar\KULIAH\Smt 2\strukturData\UAS\teori>

```