

LAPORAN PRAKTIKUM



Disusun Oleh :

Shadafi fastivan

BP/NIM: 2023/23343084

Hari/Tanggal : Selasa/ 19 Maret 2024

Sesi/Jam : 202323430164 / 07:00-08:40 WIB

Dosen Pengampu :

Randi Proska Sandra, M.Sc

Kode kelas;202323430158

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK ELEKTRONIKA

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS NEGERI PADANG

2024

Source code

```
//created by shadafi fastiyan
```

```
//nim 23343084
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct node
```

```
{  
    int data;  
    struct node *next;  
};
```

```
struct graf
```

```
{  
    int jumlah_simpul;  
    struct node **DaftarSimpul;  
    int *sudah_dikunjungi;  
};
```

```
struct node *buat_node(int x)
```

```
{  
    struct node *newnode = (struct node *)malloc(sizeof(struct node));  
    newnode->data = x;  
    newnode->next = NULL;  
    return newnode;  
}
```

```
struct graf *buat_graf(int jumlah_simpul)
```

```
{  
    struct graf *grafik = (struct graf *)malloc(sizeof(struct graf));  
    grafik->jumlah_simpul = jumlah_simpul;  
    grafik->DaftarSimpul = (struct node **)malloc(jumlah_simpul * sizeof(struct node *));
```

```

grafik->sudah_dikunjungi = (int *)malloc(jumlah_simpul * sizeof(int));
for (int i = 0; i < jumlah_simpul; i++)
{
    grafik->DaftarSimpul[i] = NULL;
    grafik->sudah_dikunjungi[i] = 0;
}
return grafik;
}

```

```

void tambah_sisi(struct graf *grafik, int sumber, int tujuan)
{
    struct node *newnode = buat_node(tujuan);
    newnode->next = grafik->DaftarSimpul[sumber];
    grafik->DaftarSimpul[sumber] = newnode;
    newnode = buat_node(sumber);
    newnode->next = grafik->DaftarSimpul[tujuan];
    grafik->DaftarSimpul[tujuan] = newnode;
}

```

// Fungsi untuk menambahkan elemen baru ke dalam antrian (queue)

```

void enqueue(int *antrian, int *belakang, int elemen)
{
    antrian[(*belakang)++] = elemen;
}

```

// Fungsi untuk menghapus elemen pertama dari antrian (queue)

```

int dequeue(int *antrian, int *depan)
{
    return antrian[(*depan)++];
}

```

```
}
```

```
void BFS(struct graf *grafik, int simpul_awal)
```

```
{
```

```
    int *antrian = (int *)malloc(grafik->jumlah_simpul * sizeof(int));
```

```
    int depan = 0;
```

```
    int belakang = 0;
```

```
    grafik->sudah_dikunjungi[simpul_awal] = 1;
```

```
    enqueue(antrian, &belakang, simpul_awal);
```

```
    while (depan < belakang)
```

```
    {
```

```
        int simpul_sekarang = deque(antrian, &depan);
```

```
        printf("%d ", simpul_sekarang);
```

```
        struct node *temp = grafik->DaftarSimpul[simpul_sekarang];
```

```
        while (temp)
```

```
        {
```

```
            int simpul_tetangga = temp->data;
```

```
            if (grafik->sudah_dikunjungi[simpul_tetangga] == 0)
```

```
            {
```

```
                grafik->sudah_dikunjungi[simpul_tetangga] = 1;
```

```
                enqueue(antrian, &belakang, simpul_tetangga);
```

```
            }
```

```
            temp = temp->next;
```

```
        }
```

```
    }
```

```

    free(antrian);
}

int main()
{
    struct graf *graf = buat_graf(4);
    tambah_sisi(graf, 0, 1);
    tambah_sisi(graf, 0, 2);
    tambah_sisi(graf, 1, 2);
    tambah_sisi(graf, 2, 0);
    tambah_sisi(graf, 2, 3);
    tambah_sisi(graf, 3, 3);
    tambah_sisi(graf, 4, 3);
    tambah_sisi(graf,4,2);

    printf("Hasil Breadth First Traversal dimulai dari simpul 2:\n");
    BFS(graf, 2);

    return 0;
}

```

Penjelasan program

Program ini mendemonstrasikan penggunaan struktur data graf dan algoritma Breadth First Search (BFS) untuk melakukan traversal pada graf. Berikut penjelasannya:

Tujuan:

- Mempelajari struktur data graf dan cara implementasinya dalam bahasa C.
- Memahami algoritma BFS dan cara kerjanya dalam melakukan traversal pada graf.
- Mencetak semua simpul dalam graf dengan urutan BFS, dimulai dari simpul tertentu.

Struktur Data:

- **Node:** Sebuah struct yang mewakili simpul dalam graf. Memiliki dua anggota:
 - data: Menyimpan data yang terkait dengan simpul.
 - next: Pointer ke simpul berikutnya dalam daftar simpul yang terhubung.
- **Graf:** Sebuah struct yang mewakili graf itu sendiri. Memiliki tiga anggota:
 - jumlah_simpul: Jumlah simpul dalam graf.
 - DaftarSimpul: Array pointer ke simpul.
 - sudah_dikunjungi: Array integer yang menandakan apakah simpul telah dikunjungi selama traversal.

Fungsi:

- `buat_node(int x)`: Membuat dan mengalokasikan memori untuk simpul baru dengan data x.
- `buat_graf(int jumlah_simpul)`: Membuat dan mengalokasikan memori untuk graf baru dengan jumlah_simpul. Menginisialisasi DaftarSimpul dan sudah_dikunjungi.
- `tambah_sisi(struct graf *grafik, int sumber, int tujuan)`: Menambahkan sisi (edge) baru ke dalam graf. Menghubungkan simpul sumber dan tujuan.
- `enqueue(int *antrian, int *belakang, int elemen)`: Menambahkan elemen baru ke dalam antrian (queue).
- `deque(int *antrian, int *depan)`: Menghapus elemen pertama dari antrian (queue).
- `BFS(struct graf *grafik, int simpul_awal)`: Melakukan traversal pada graf dengan algoritma BFS dimulai dari simpul simpul_awal.

Algoritma BFS:

1. Menandai simpul awal sebagai dikunjungi dan menambahkannya ke dalam antrian.
2. Mengambil simpul pertama dari antrian dan mencetak datanya.
3. Menjelajahi semua simpul yang terhubung dengan simpul yang diambil dari antrian:
 - Jika simpul belum dikunjungi, tandai sebagai dikunjungi dan tambahkan ke antrian.
4. Mengulangi langkah 2 dan 3 hingga antrian kosong.

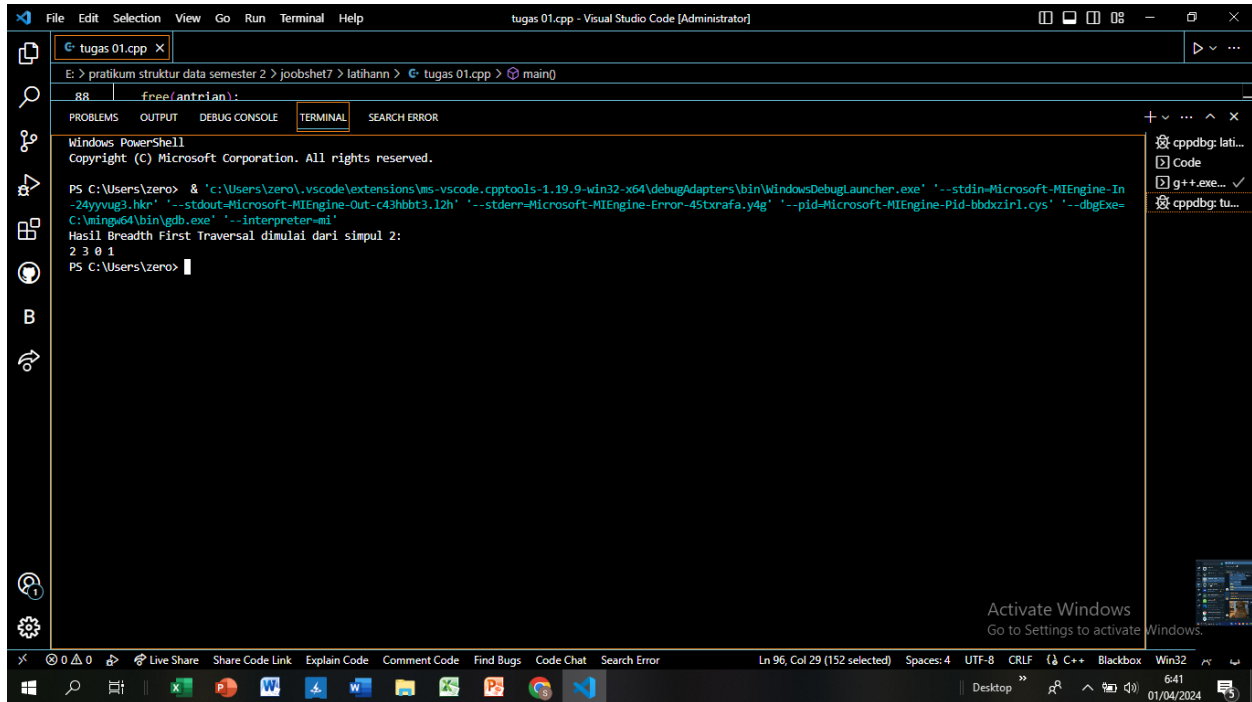
Main Function:

- Membuat graf dengan 4 simpul.
- Menambahkan sisi ke dalam graf untuk membentuk struktur yang diinginkan.
- Memanggil fungsi BFS untuk melakukan traversal pada graf dimulai dari simpul 2.
- Mencetak data simpul dalam urutan BFS.

Kesimpulan:

Program ini menunjukkan cara menggunakan struktur data graf dan algoritma BFS untuk melakukan traversal pada graf. Algoritma BFS membantu dalam menjelajahi semua simpul dalam graf secara sistematis dan efisien.

output



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the output of a C++ program that performs a Breadth First Traversal (BFS) on a graph. The output is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\zero> & 'c:\Users\zero\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-24yyvug3.hkr' '--stdout=Microsoft-MIEngine-Out-c43hbt3.12h' '--stderr=Microsoft-MIEngine-Error-45txrafa.y4g' '--pid=Microsoft-MIEngine-Pid-bbdxzir1.cys' '--dbgExe=C:\mingw64\bin\gdb.exe' '--interpreter=mi'
Hasil Breadth First Traversal dimulai dari simpul 2:
2 3 0 1
PS C:\Users\zero>
```

The terminal window is titled "tugas 01.cpp - Visual Studio Code [Administrator]". The status bar at the bottom indicates the current line and column as "Ln 96, Col 29 (152 selected)".