

LAPORAN PRATIKUM STRUKTUR DATA

JOBSHEET 7



Disusun Oleh :

NAJWA ELDIARA OWILIA TIKSA

BP/NIM : 2023/23343079

Dosen Pengampu :

Randi Proska Sandra, S.Pd, M.Sc

Kode Kelas : 202323430157

PROGRAM STUDI S1 INFORMATIKA

DEPARTEMEN ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

A. Source Code

```
//Created by 23343079_Najwa Eldiara Owilia Tiksa
#include <stdio.h>
#include <stdlib.h>

#define MAKSIMUM_SIMPUL 100

typedef struct Graf {
    int jumlahSimpul;
    int matriksKetetanggaan[MAKSIMUM_SIMPUL][MAKSIMUM_SIMPUL];
    int sudahDikunjungi[MAKSIMUM_SIMPUL];
} Graf;

typedef struct Antrian {
    int item[MAKSIMUM_SIMPUL];
    int depan;
    int belakang;
} Antrian;

Antrian* buatAntrian() {
    Antrian* antrian = (Antrian*)malloc(sizeof(Antrian));
    antrian->depan = -1;
    antrian->belakang = -1;
    return antrian;
}

int kosong(Antrian* antrian) {
    return antrian->belakang == -1;
}

void masukkan(Antrian* antrian, int nilai) {
    if (antrian->belakang == MAKSIMUM_SIMPUL - 1) {
        printf("\nAntrian penuh!!");
        return;
    }
    if (antrian->depan == -1) {
        antrian->depan = 0;
    }
    antrian->belakang++;
    antrian->item[antrian->belakang] = nilai;
}

int keluarkan(Antrian* antrian) {
    if (kosong(antrian)) {
        printf("\nAntrian kosong!!");
        return -1;
    }
    int item = antrian->item[antrian->depan];
    antrian->depan++;
    if (antrian->depan > antrian->belakang) {
        antrian->depan = antrian->belakang = -1;
    }
    return item;
}
```

```

Graf* buatGraf(int simpul) {
    Graf* graf = (Graf*)malloc(sizeof(Graf));
    graf->jumlahSimpul = simpul;
    for (int i = 0; i < simpul; i++) {
        for (int j = 0; j < simpul; j++) {
            graf->matriksKetetanggaan[i][j] = 0;
        }
        graf->sudahDikunjungi[i] = 0;
    }
    return graf;
}

void tambahSisi(Graf* graf, int src, int dest) {
    graf->matriksKetetanggaan[src][dest] = 1;
    graf->matriksKetetanggaan[dest][src] = 1;
}

void bfs(Graf* graf, int simpulAwal) {
    Antrian* antrian = buatAntrian();

    graf->sudahDikunjungi[simpulAwal] = 1;
    masukkan(antrian, simpulAwal);

    while (!kosong(antrian)) {
        int simpulSekarang = keluarkan(antrian);
        printf("%d ", simpulSekarang);

        for (int i = 0; i < graf->jumlahSimpul; i++) {
            if (graf->matriksKetetanggaan[simpulSekarang][i] == 1
&& !graf->sudahDikunjungi[i]) {
                graf->sudahDikunjungi[i] = 1;
                masukkan(antrian, i);
            }
        }
    }
}

void temukanKomponenTerhubung(Graf* graf) {
    printf("Komponen terhubung:\n");
    for (int i = 0; i < graf->jumlahSimpul; i++) {
        if (!graf->sudahDikunjungi[i]) {
            bfs(graf, i);
        }
    }
}

int main() {
    int jumlahSimpul;
    printf("Masukkan jumlah simpul: ");
    scanf("%d", &jumlahSimpul);

    Graf* graf = buatGraf(jumlahSimpul);

    int jumlahSisi;
    printf("Masukkan jumlah sisi: ");
    scanf("%d", &jumlahSisi);
}

```

```

printf("Masukkan sisi-sisi:\n");
for (int i = 0; i < jumlahSisi; i++) {
    int src, dest;
    scanf("%d %d", &src, &dest);
    tambahSisi(graf, src, dest);
}

temukanKomponenTerhubung(graf);

return 0;
}

```

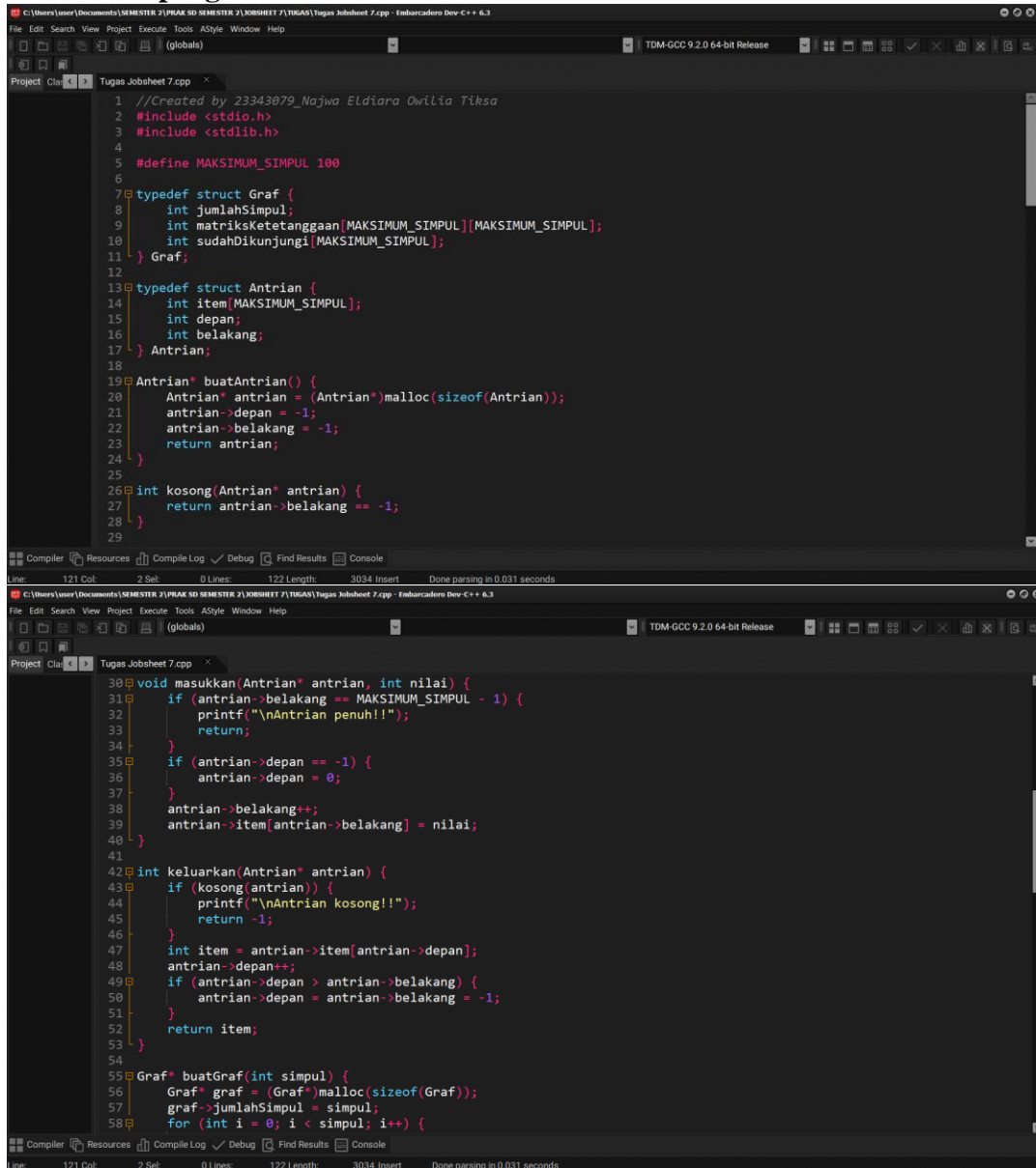
B. Penjelasan Program

Program ini adalah implementasi dari algoritma Breath-First Search (BFS) untuk mencari komponen terhubung dalam sebuah graf. Ini adalah algoritma pencarian yang digunakan untuk menjelajahi atau mencari semua simpul dari graf, dimulai dari simpul awal tertentu dan mengunjungi semua simpul yang terhubung dengan simpul tersebut secara terurut.

1. Dua struktur data digunakan: Graf untuk merepresentasikan graf menggunakan matriks ketetanggaan dan Antrian untuk mengimplementasikan antrian yang diperlukan oleh algoritma BFS.
2. Fungsi `buatAntrian()` digunakan untuk membuat antrian baru.
3. Fungsi `kosong()` digunakan untuk memeriksa apakah antrian kosong.
4. Fungsi `masukkan()` digunakan untuk memasukkan elemen ke dalam antrian.
5. Fungsi `keluarkan()` digunakan untuk mengeluarkan elemen dari antrian.
6. Fungsi `buatGraf()` digunakan untuk membuat graf baru dengan jumlah simpul tertentu.
7. Fungsi `tambahSisi()` digunakan untuk menambahkan sisi antara dua simpul dalam graf.
8. Fungsi `bfs()` merupakan implementasi algoritma BFS untuk menjelajahi graf secara terurut.
9. Fungsi `temukanKomponenTerhubung()` digunakan untuk menemukan dan mencetak komponen terhubung dalam graf.
10. Fungsi `main()` merupakan titik masuk utama program yang meminta pengguna untuk memasukkan jumlah simpul, jumlah sisi, dan sisi-sisi dari graf, lalu menjalankan algoritma BFS untuk mencari komponen terhubung dalam graf tersebut.

Algoritma BFS sendiri adalah algoritma pencarian yang sistematis yang dimulai dari simpul awal dan menelusuri semua simpul yang terhubung dengannya, sebelum melanjutkan ke simpul yang lain. Algoritma ini menggunakan antrian untuk melacak simpul yang akan dieksplorasi selanjutnya.

C. Screenshot program



```
1 //Created by 23343079_Najwa Eldiara Owilia Tiksa
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 #define MAKSIMUM_SIMPUL 100
6
7 typedef struct Graf {
8     int jumlahSimpul;
9     int matriksKetetanggaan[MAKSIMUM_SIMPUL][MAKSIMUM_SIMPUL];
10    int sudahDikunjungi[MAKSIMUM_SIMPUL];
11 } Graf;
12
13 typedef struct Antrian {
14     int item[MAKSIMUM_SIMPUL];
15     int depan;
16     int belakang;
17 } Antrian;
18
19 Antrian* buatAntrian() {
20     Antrian* antrian = (Antrian*)malloc(sizeof(Antrian));
21     antrian->depan = -1;
22     antrian->belakang = -1;
23     return antrian;
24 }
25
26 int kosong(Antrian* antrian) {
27     return antrian->belakang == -1;
28 }
29
30 void masukkan(Antrian* antrian, int nilai) {
31     if (antrian->belakang == MAKSIMUM_SIMPUL - 1) {
32         printf("\nAntrian penuh!!");
33         return;
34     }
35     if (antrian->depan == -1) {
36         antrian->depan = 0;
37     }
38     antrian->belakang++;
39     antrian->item[antrian->belakang] = nilai;
40 }
41
42 int keluarkan(Antrian* antrian) {
43     if (kosong(antrian)) {
44         printf("\nAntrian kosong!!");
45         return -1;
46     }
47     int item = antrian->item[antrian->depan];
48     antrian->depan++;
49     if (antrian->depan > antrian->belakang) {
50         antrian->depan = antrian->belakang = -1;
51     }
52     return item;
53 }
54
55 Graf* buatGraf(int simpul) {
56     Graf* graf = (Graf*)malloc(sizeof(Graf));
57     graf->jumlahSimpul = simpul;
58     for (int i = 0; i < simpul; i++) {
```

```
C:\Users\user\Documents\SEMESTER 2\PEAK SD SEMESTER 2\JOBSHEET 7\TUGAS\Tugas Jobsheet 7.cpp - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals) TDM-GCC 9.2.0 64-bit Release

Project: Tugas Jobsheet 7.cpp
59     for (int j = 0; j < simpul; j++) {
60         graf->matriksKetetanggaan[i][j] = 0;
61     }
62     graf->sudahDikunjungi[i] = 0;
63 }
64 return graf;
65 }
66
67 void tambahSisi(Graf* graf, int src, int dest) {
68     graf->matriksKetetanggaan[src][dest] = 1;
69     graf->matriksKetetanggaan[dest][src] = 1;
70 }
71
72 void bfs(Graf* graf, int simpulAwal) {
73     Antrian* antrian = buatAntrian();
74
75     graf->sudahDikunjungi[simpulAwal] = 1;
76     masukkan(antrian, simpulAwal);
77
78     while (!kosong(antrian)) {
79         int simpulSekarang = keluarkan(antrian);
80         printf("%d ", simpulSekarang);
81
82         for (int i = 0; i < graf->jumlahSimpul; i++) {
83             if (graf->matriksKetetanggaan[simpulSekarang][i] == 1 && !graf->sudahDikunjungi[i]) {
84                 graf->sudahDikunjungi[i] = 1;
85                 masukkan(antrian, i);
86             }
87         }
88     }
89 }
90
91 void printKomponenTerhubung(Graf* graf) {
92     printf("Komponen terhubung:\n");
93     for (int i = 0; i < graf->jumlahSimpul; i++) {
94         if (!graf->sudahDikunjungi[i]) {
95             bfs(graf, i);
96         }
97     }
98 }
99
100 int main() {
101     int jumlahSimpul;
102     printf("Masukkan jumlah simpul: ");
103     scanf("%d", &jumlahSimpul);
104
105     Graf* graf = buatGraf(jumlahSimpul);
106
107     int jumlahSisi;
108     printf("Masukkan jumlah sisi: ");
109     scanf("%d", &jumlahSisi);
110
111     printf("Masukkan sisi-sisi:\n");
112     for (int i = 0; i < jumlahSisi; i++) {
113         int src, dest;
114         scanf("%d %d", &src, &dest);
115         tambahSisi(graf, src, dest);
116     }
117
118     temukanKomponenTerhubung(graf);
119
120     return 0;
121 }
```

D. Output

```
C:\Users\User\Documents\SE7 < + - >
Masukkan jumlah simpul: 4
Masukkan jumlah sisi: 4
Masukkan sisi-sisi:
a b c d
Komponen terhubung:
0 1 2 3
-----
Process exited after 28.3 seconds with return value 0
Press any key to continue . . .
```