Nama : Nabiilah Nur Fauziyyah

NPM : 2310631170105

Kelas : 2C – Informatika

Tugas Struktur Data

Link Github : https://github.com/fauziyyah22/NabiilahNF_SD_Tugas8

A. Buatlah laporan dari source code pada link di bawah ini :

Mengukur Jarak Terpendek Dari Sebuah Vertex Ke Vertex Lain Pada Weighted Graph

```cpp
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <climits>
using namespace std;

#define MAX 100

int graph[MAX][MAX];
int vertexWeights[MAX];
int n;
char simpul1 = 'A';
char simpul2 = 'A';

// Fungsi ini digunakan untuk menambahkan edge ke graf
void addEdge(int n) {
    int i, j, weight;
    cout << "Beri nilai weight untuk edge antara kedua simpul yang terhubung, dan 0 jika kedua simpul tidak terhubung" << endl << endl;
    for (i = 0; i < n; i++) {
        cout << "Simpul " << simpul1++ << " terhubung dengan: " << endl;
        for (j = 0; j < n; j++) {
            if (i != j) {
                cout << "simpul " << char('A' + j) << " (weight): ";
                cin >> weight;
                graph[i][j] = weight;
            } else {
                graph[i][j] = 0; // No self-loops
            }
        }
        cout << endl;
    }
    simpul1 = 'A';
}

// Fungsi ini digunakan untuk menambahkan weight pada setiap simpul
void addVertexWeights(int n) {
    int weight;
    for (int i = 0; i < n; i++) {
        cout << "Masukkan weight untuk simpul " << char('A' + i) << ": ";
        cin >> weight;
        vertexWeights[i] = weight;
    }
}
```

```
45  // Fungsi ini digunakan untuk mencetak adjacency matriks dari graf
46  void printGraph(int n) {
47      cout << "Cetak Adjacency Matrix" << endl << endl;
48      int i, j;
49      cout << "   ";
50      for (i = 0; i < n; i++) {
51          cout << char('A' + i) << "   ";
52      }
53      cout << endl;
54      for (i = 0; i < n; i++) {
55          cout << char('A' + i) << "   ";
56          for (j = 0; j < n; j++) {
57              cout << graph[i][j] << "   ";
58          }
59          cout << endl;
60      }
61  }
62
63  // Fungsi ini digunakan untuk mencari jalur terpendek
64  // antara 2 simpul menggunakan algoritma Dijkstra
65  void searchPath(char x, char y) {
66      int source = x - 'A';
67      int destination = y - 'A';
68      int dist[MAX];
69      int parent[MAX];
70      bool visited[MAX] = {false};
71
72      for (int i = 0; i < n; i++) {
73          dist[i] = INT_MAX;
74          parent[i] = -1;
75      }
76
77      dist[source] = 0;
78
79      for (int count = 0; count < n - 1; count++) {
80          int minDist = INT_MAX, u;
81
82          for (int i = 0; i < n; i++) {
83              if (!visited[i] && dist[i] <= minDist) {
84                  minDist = dist[i];
85                  u = i;
86              }
87          }
88
89          visited[u] = true;
90
91          for (int v = 0; v < n; v++) {
92              if (!visited[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
93                  dist[v] = dist[u] + graph[u][v];
94                  parent[v] = u;
95              }
96          }
97      }
98
```

```
    if (dist[destination] == INT_MAX) {
        cout << "Tidak ada jalur dari " << x << " ke " << y << endl;
    } else {
        cout << "Jarak terpendek dari " << x << " ke " << y << " adalah " << dist[destination] << endl;

        cout << "Jalur terpendek adalah: ";
        int path[MAX], count = 0;
        for (int v = destination; v != -1; v = parent[v]) {
            path[count++] = v;
        }
        for (int i = count - 1; i >= 0; i--) {
            cout << char(path[i] + 'A');
            if (i > 0) cout << " -> ";
        }
        cout << endl;
    }
}

// Fungsi ini digunakan untuk menghapus sisi antara 2 simpul
void deleteEdge(char x, char y) {
    int i = x - 'A';
    int j = y - 'A';
    graph[i][j] = 0;
    graph[j][i] = 0;
    cout << "Garis antara simpul " << x << " dan simpul " << y << " berhasil dihapus!" << endl;
}

// Fungsi ini digunakan untuk menghapus simpul dari graf
void deleteVertex(char z) {
    int v = z - 'A';
    if (v >= n) {
        cout << "Simpul tidak ada" << endl;
        return;
    }

    for (int i = 0; i < n; i++) {
        for (int j = v; j < n - 1; j++) {
            graph[i][j] = graph[i][j + 1];
        }
    }

    for (int i = v; i < n - 1; i++) {
       for (int j = 0; j < n; j++) {
           graph[i][j] = graph[i + 1][j];
       }
    }

    n--;
    cout << "Simpul " << z << " berhasil dihapus!" << endl;
}

// Fungsi untuk menampilkan identitas
void identitas() {
    cout << "\nProgram Mengukur Jarak Terpendek Dari Sebuah Vertex Ke Vertex Lain Pada Weighted Graph";
    cout << "\nNama      : Nabiilah Nur Fauziyyah";
    cout << "\nNPM       : 2310631170105";
    cout << "\nKelas     : 2C - Informatika" << "\n";
}
```

```
158  // Fungsi utama yang menjalankan program
159  int main() {
160      first:
161      system("cls");
162      char name = 'A', x, y;
163      int pil;
164
165      cout << "=======================================" << endl;
166      cout << "             Adjacency Matrix          " << endl;
167      cout << "=======================================" << endl;
168      cout << "1. Tambah simpul dan sisi " << endl;
169      cout << "2. Print graph " << endl;
170      cout << "3. Cari jalur " << endl;
171      cout << "4. Hapus simpul " << endl;
172      cout << "5. Hapus sisi " << endl;
173      cout << "6. Keluar " << endl;
174      cout << "\nMasukkan pilihan : "; cin >> pil;
175
176      if (pil == 1) {
177          system("cls");
178          cout << "Masukkan jumlah n : ";
179          cin >> n;
180          addEdge(n);
181          addVertexWeights(n);
182
183          cout << endl;
184          cout << "Simpul berhasil dibuat." << endl;
185          cout << "Tekan apa saja untuk melanjutkan!";
186          getch();
187          goto first;
188      } else if (pil == 2) {
189          system("cls");
190          printGraph(n);
191          cout << "\nTekan apa saja untuk melanjutkan!";
192          getch();
193          goto first;
194      } else if (pil == 3) {
195          system("cls");
196          cout << "Mencari jalur terpendek " << endl;
197          cout << "Masukkan node asal    : "; cin >> x;
198          cout << "Masukkan node tujuan : "; cin >> y;
199          searchPath(x, y);
200          cout << endl;
201          cout << "Tekan apa saja untuk melanjutkan!";
202          getch();
203          goto first;
```

```
204         } else if (pil == 4) {
205             system("cls");
206             printGraph(n);
207             cout << "\nMenghapus simpul = "; cin >> x;
208             deleteVertex(x);
209             cout << endl;
210             cout << "Tekan apa saja untuk melanjutkan!";
211             getch();
212             goto first;
213         } else if (pil == 5) {
214             system("cls");
215             printGraph(n);
216             cout << "\nMenghapus garis antara simpul "; cin >> x;
217             cout << " dengan simpul "; cin >> y;
218             deleteEdge(x, y);
219             cout << endl;
220             cout << "Tekan apa saja untuk melanjutkan!";
221             getch();
222             goto first;
223         } else if (pil == 6) {
224             system("cls");
225             cout << "Terimakasih sudah menggunakan program ini!" << endl;
226             identitas();
227             return 0;
228         } else {
229             cout << "Input yang anda masukkan salah" << endl;
230             char rep;
231             cout << "Apakah anda ingin melanjutkan?"; cin >> rep;
232             if (rep == 'y' || rep == 'Y') {
233                 cout << endl;
234                 cout << "Tekan apa saja untuk melanjutkan!";
235                 getch();
236                 goto first;
237             } else {
238                 return 0;
239             }
240         }
241     }
242
```

Penjelasan :

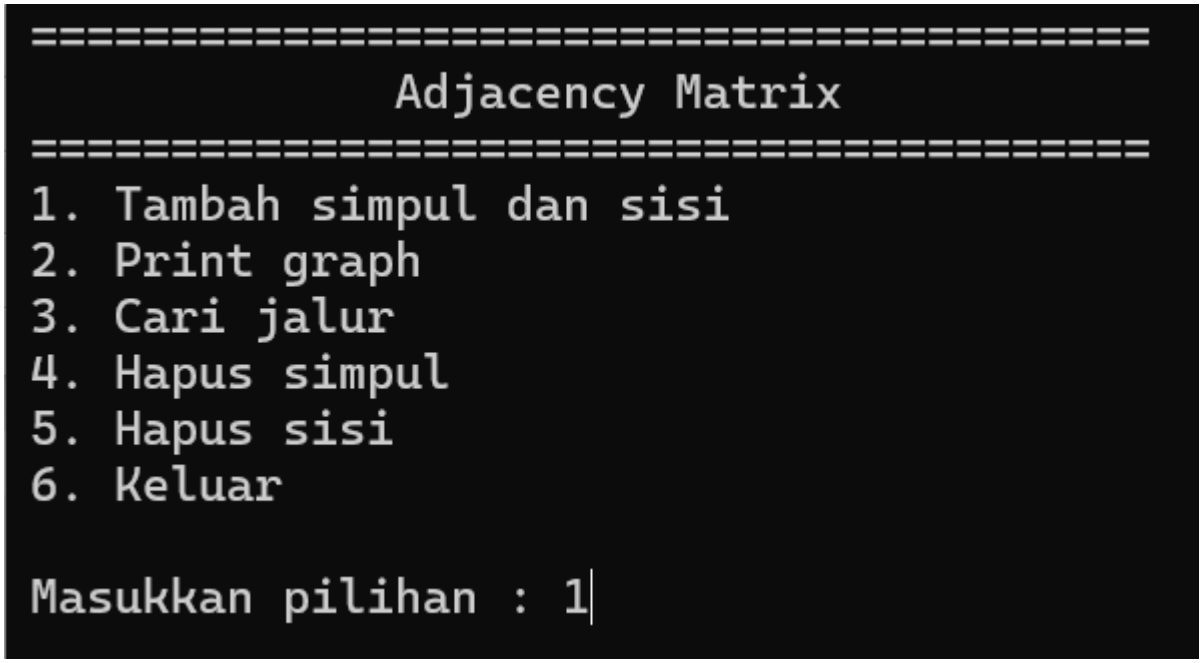
Program ini adalah sebuah program yang mengimplementasikan representasi graf menggunakan adjacency matrix. Berikut adalah alur atau cara kerja program ini dari main():

1. Variabel-variabel yang digunakan dalam program ini diinisialisasi, seperti simpul1 dan simpul2 yang merepresentasikan simpul-simpul dalam graf.
2. Program menampilkan menu interaktif kepada pengguna yang berisi beberapa pilihan operasi yang dapat dilakukan pada graf.
3. Pilihan tersebut mencakup menambah simpul dan sisi, mencetak graf, mencari jalur terpendek, menghapus simpul, menghapus sisi, dan keluar dari program.
4. Setelah menampilkan menu, program membaca input dari pengguna untuk memilih operasi yang diinginkan.

5. Bergantung pada pilihan yang dimasukkan pengguna, program akan melakukan operasi yang sesuai.
6. Jika pengguna memilih untuk menambah simpul dan sisi (pil == 1), program akan meminta jumlah simpul, mengisi adjacency matrix dengan weight yang sesuai, dan menambahkan weight untuk setiap simpul.
7. Jika pengguna memilih untuk mencetak graf (pil == 2), program akan mencetak adjacency matrix dari graf.
8. Jika pengguna memilih untuk mencari jalur terpendek (pil == 3), program akan meminta input simpul asal dan simpul tujuan, lalu mencari jalur terpendek antara keduanya menggunakan algoritma Dijkstra.
9. Jika pengguna memilih untuk menghapus simpul (pil == 4), program akan meminta input simpul yang akan dihapus, lalu menghapus simpul tersebut beserta sisi-sisinya dari graf.
10. Jika pengguna memilih untuk menghapus sisi (pil == 5), program akan meminta input dua simpul yang berhubungan, lalu menghapus sisi yang menghubungkan keduanya dari graf.
11. Jika pengguna memilih untuk keluar dari program (pil == 6), program akan menampilkan pesan terima kasih dan keluar dari loop while.
12. Setelah mengeksekusi operasi yang diminta pengguna, program kembali ke awal loop while untuk menampilkan kembali menu interaktif.
13. Pengguna dapat terus berinteraksi dengan program dan melakukan operasi yang diinginkan hingga memilih untuk keluar.

Demikianlah alur kerja dari program ini, di mana pengguna dapat berinteraksi dengan graf dan melakukan berbagai operasi yang relevan sesuai dengan kebutuhan mereka.

Hasil dari program tersebut sebagai berikut :



```
Masukkan jumlah n : 3
Beri nilai weight untuk edge antara kedua simpul yang terhubung,
dan 0 jika kedua simpul tidak terhubung

Simpul A terhubung dengan:
simpul B (weight): 2
simpul C (weight): 3

Simpul B terhubung dengan:
simpul A (weight): 2
simpul C (weight): 4

Simpul C terhubung dengan:
simpul A (weight): 3
simpul B (weight): 4

Masukkan weight untuk simpul A: 1
Masukkan weight untuk simpul B: 2
Masukkan weight untuk simpul C: 3

Simpul berhasil dibuat.
Tekan apa saja untuk melanjutkan!
```

```
==========================================
            Adjacency Matrix
==========================================
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi
6. Keluar

Masukkan pilihan : 2
```

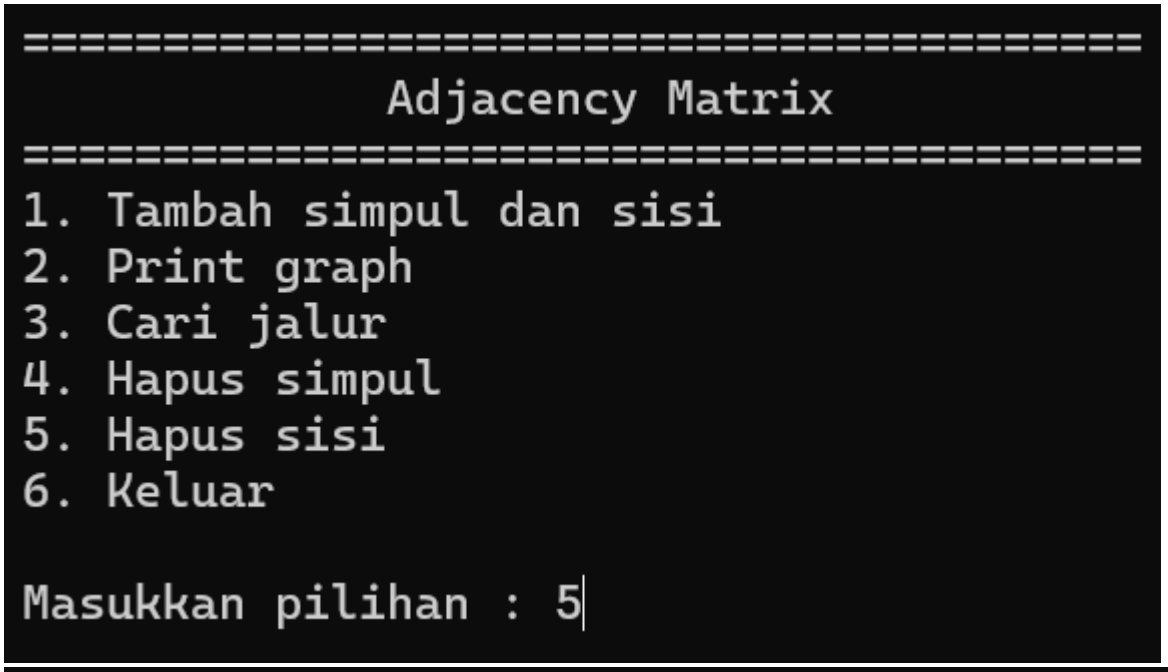
```
Cetak Adjacency Matrix

  A  B  C
A  0  2  3
B  2  0  4
C  3  4  0

Tekan apa saja untuk melanjutkan!
```

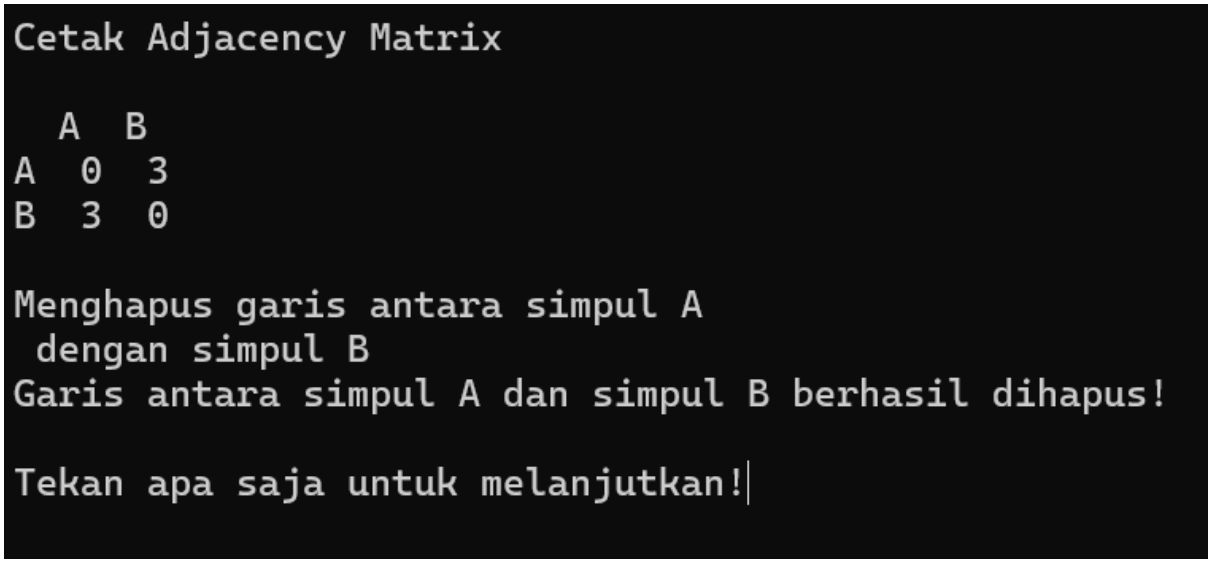
```
==========================================
              Adjacency Matrix
==========================================
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi
6. Keluar

Masukkan pilihan : 3
```

```
Mencari jalur terpendek
Masukkan node asal   : A
Masukkan node tujuan : C
Jarak terpendek dari A ke C adalah 3
Jalur terpendek adalah: A -> C

Tekan apa saja untuk melanjutkan!
```

```
========================================
            Adjacency Matrix
========================================
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi
6. Keluar

Masukkan pilihan : 4
```

```
Cetak Adjacency Matrix

  A  B  C
A  0  2  3
B  2  0  4
C  3  4  0

Menghapus simpul = B
Simpul B berhasil dihapus!

Tekan apa saja untuk melanjutkan!
```


==========
Adjacency Matrix
==========
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi
6. Keluar
Masukkan pilihan : 5


Cetak Adjacency Matrix
A B
A 0 3
B 3 0
Menghapus garis antara simpul A
dengan simpul B
Garis antara simpul A dan simpul B berhasil dihapus!
Tekan apa saja untuk melanjutkan!

```
==========================================
              Adjacency Matrix
==========================================
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi

Masukkan pilihan : 6
Input yang anda masukkan salah
Apakah anda ingin melanjutkan?
```

```
==========================================
              Adjacency Matrix
==========================================
1. Tambah simpul dan sisi
2. Print graph
3. Cari jalur
4. Hapus simpul
5. Hapus sisi
6. Keluar

Masukkan pilihan : 6
```

```
Terimakasih sudah menggunakan program ini!

Program Mengukur Jarak Terpendek Dari Sebuah Vertex Ke Vertex Lain Pada Weighted Graph
Nama     : Nabiilah Nur Fauziyyah
NPM      : 2310631170105
Kelas    : 2C - Informatika
```