TUGAS PERTEMUAN 8

Nama : Zaldy Seno Yudhanto

Kelas : 2C – Informatika

NPM : 2310631170123

1. Buatlah laporan dari source code pada link di bawah ini :
   (Mengukur Jarak Terpendek Dari Sebuah Vertex Ke Vertex Lain Pada Weighted Graph)

Jawab!

```
#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';
```

Yang pertama didefinisikan MAX yaitu 100 yang akan kita gunakan menjadi ukuran sebuah grafnya, kemduian kita membuat sebuah matriks 2x2 yang diberi nama *graph* dengan panjang MAX yang sudah kita definisikan diawal yaitu 100, kemudian kita membuat matriks dimensi satu yang digunakan untuk menyimpan bobot dari simpul. Kemudian kita mendeklarasikan variabel dengan tipe data int yang diberi nama *n* yang kita gunakan untuk menyimpan jumlah simpul, kemudian *simpul 1* dan *simpul 2*digunakan utnuk memberi label pada simpul.

```
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';
}
```

Lalu kita membuat sebuah fungsi kosong yang diberi nama *addEdge* yang berfungsi untuk memasukan bobot untuk setiap sisi antara simpul yang terhubung, kemudian berguna untuk memabngun matriks adjacency dengan menanyakan bobot antara pasangan simpul.

Disana dideklarasikan variabel dengan tipe data int dengan nama *i,j,* dan *weight.* Kemudian kita diminta untuk memberi bobot untuk edge antara kedua simpul yang terhubung. Sama seperti memasukan inputan ke dalam matriks kita menggunakan sebuah perulangan yang akan berhenti ketika i++ sudah mencapai batas yang diminta, disitu batasnya sampai kurang dari *n* yang sudah kita masukan.

Lalu ada perulangan lagi yang digunakan untuk menyimpan bobotnya kedalam matriks disana ada pengkondisian diman jika i tidak sama dengan j maka char(A+J), kemudian dimasukan ke dalam matriks, namun jika selain itu maka matriks[i][j]=0.

```
34  void addVertexWeights(int n) {
35      int weight;
36      for (int i = 0; i < n; i++) {
37          cout << "Masukkan weight untuk simpul " << char('A' + i) << ": ";
38          cin >> weight;
39          vertexWeights[i] = weight;
40      }
41  }
```

Kemudian kita membuat fungsi kosong yang diberi nama *addVertexWeights* yang digunakan untuk memasukan bobot untuk setiap simpul. Disana dideklarasikan variabel dengan tipe data int dan di beri nama *weight* kemudian ada perulangan yang akan berhenti jika jumlah i melebihi n. Kita diminta untuk memasukan bobot yang nantinya akan dimasukan ke dalam variabel *weight.* Kemudian matriks *vertexWeight* yang sudah kita definisikan diawal kita deklarasikan sebagai *weight.*

```
43  void printGraph(int n) {
44      cout << "Cetak Adjacency Matrix" << endl << endl;
45      int i, j;
46      cout << "   ";
47      for (i = 0; i < n; i++) {
48          cout << char('A' + i) << "   ";
49      }
50      cout << endl;
51      for (i = 0; i < n; i++) {
52          cout << char('A' + i) << "   ";
53          for (j = 0; j < n; j++) {
54              cout << graph[i][j] << "   ";
55          }
56          cout << endl;
57      }
58  }
```

Kemudian kita membuat fungsi kosong yang digunakan untuk mencetak matriks adjacency untuk mencetak graf. Sama seperti menampilkan matriks biasa, kita menggunakan perulangan yang akan berhenti jika sudah melebihi n tertentu.

```
60  void searchPath(char x, char y) {
61      int source = x - 'A';
62      int destination = y - 'A';
63      int dist[MAX];
64      int parent[MAX];
65      bool visited[MAX] = {false};
66
67      for (int i = 0; i < n; i++) {
68          dist[i] = INT_MAX;
69          parent[i] = -1;
70      }
71
72      dist[source] = 0;
73
74      for (int count = 0; count < n - 1; count++) {
75          int minDist = INT_MAX, u;
76
77          for (int i = 0; i < n; i++) {
78              if (!visited[i] && dist[i] <= minDist) {
79                  minDist = dist[i];
80                  u = i;
81              }
82          }
83
84          visited[u] = true;
```

```
85
86          for (int v = 0; v < n; v++) {
87              if (!visited[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
88                  dist[v] = dist[u] + graph[u][v];
89                  parent[v] = u;
90              }
91          }
92      }
93
94      if (dist[destination] == INT_MAX) {
95          cout << "Tidak ada jalur dari " << x << " ke " << y << endl;
96      } else {
97          cout << "Jarak terpendek dari " << x << " ke " << y << " adalah " << dist[destination] << endl;
98
99          cout << "Jalur terpendek adalah: ";
100         int path[MAX], count = 0;
101         for (int v = destination; v != -1; v = parent[v]) {
102             path[count++] = v;
103         }
104         for (int i = count - 1; i >= 0; i--) {
105             cout << char(path[i] + 'A');
106             if (i > 0) cout << " -> ";
107         }
108         cout << endl;
109     }
110 }
```

Kemudian kita membuat fungsi kosong yang digunakan untuk mencari jalur terpendek antara dua simpul menggunakan algoritma dijkstra, jika tidak ada jalur, maka kita akan diberitahu, namun jika ada maka akan dicetak jarak dan jalur terpendek.

```
112 void deleteEdge(char x, char y) {
113     int i = x - 'A';
114     int j = y - 'A';
115     graph[i][j] = 0;
116     graph[j][i] = 0;
117     cout << "Garis antara simpul " << x << " dan simpul " << y << " berhasil dihapus!" << endl;
118 }
```

Kemudian kita membuat fungsi kosong yang diberi nama deleteEdge yang digunakan untuk menghapus sisi antara dua simpul dengan mengatur nilai di matriks adjacency menjadi 0.

```
120 void deleteVertex(char z) {
121     int v = z - 'A';
122     if (v >= n) {
123         cout << "Simpul tidak ada" << endl;
124         return;
125     }
126 
127     for (int i = 0; i < n; i++) {
128         for (int j = v; j < n - 1; j++) {
129             graph[i][j] = graph[i][j + 1];
130         }
131     }
132 
133     for (int i = v; i < n - 1; i++) {
134         for (int j = 0; j < n; j++) {
135             graph[i][j] = graph[i + 1][j];
136         }
137     }
138 
139     n--;
140     cout << "Simpul " << z << " berhasil dihapus!" << endl;
141 }
```

Lalu kita membuat fungsi kosong yang digunakan untuk menghapus simpul dari graf dengan mengeser kolom dan baris yang terkait dalam matriks adjacency dan mengurangi jumlah simpul, kemudian setelah berhasil dihapus maka akan diberi pesan bahwa simpul berhasi dihapus.

```
143 int main() {
144     first:
145     system("cls");
146     char name = 'A', x, y;
147     int pil;
148 
149     cout << "=======================================" << endl;
150     cout << "            Adjacency Matrix           " << endl;
151     cout << "=======================================" << endl;
152     cout << "1. Tambah simpul dan sisi " << endl;
153     cout << "2. Print graph " << endl;
154     cout << "3. Cari jalur " << endl;
155     cout << "4. Hapus simpul " << endl;
156     cout << "5. Hapus sisi " << endl;
157     cout << "\nMasukkan pilihan : "; cin >> pil;
158 
159     if (pil == 1) {
160         system("cls");
161         cout << "Masukkan jumlah n : ";
162         cin >> n;
163         addEdge(n);
164         addVertexWeights(n);
165 
166         cout << endl;
167         cout << "Simpul berhasil dibuat." << endl;
168         cout << "Tekan apa saja untuk melanjutkan!";
169         getch();
170         goto first;
```

Lalu kita masuk kedalam fungsi utamanya dimana kita mendeklarasikan first terlebih dahulu agar bisa dipanggil saat kita menggunakan *go to,* kemudian setelah itu kita mengahpus layar agar bersih, kemudian kita deklarasikan variabel dengan tipe data char dengan *name* dan dideklarasikan menjadi ‘A’, x, y. Kemudian kita mendeklarasikan variabel dengan tipe data int dengan nama *pil* yang digunakan untuk menyimpan pilihan kita.

Lalu kita menampilkan menu seperti yang terlihat diatas, kemudian setelah menampilkan menu kita diminta untuk memasukan pilihan dan masuk kedalam pengkondisian dimana, jika kita memilih (1) maka layara akan dibersihkan kemudian kita akan diminta untuk memasukan jumlah n yang nantinya akan menjadi bobot pada fungsi yang sudah kita buat tadi, setelah itu kita panggil fungsi *addEdge,* dan *addVertexWeights* agar *n* tadi dioprasikan. Kemudian akan diberikan pesan bahwa simpul telah berhasil dibuat dan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal.

```
} else if (pil == 2) {
    system("cls");
    printGraph(n);
    cout << "\nTekan apa saja untuk melanjutkan!";
    getch();
    goto first;
} else if (pil == 3) {
    system("cls");
    cout << "Mencari jalur terpendek " << endl;
    cout << "Masukkan node asal    : "; cin >> x;
    cout << "Masukkan node tujuan : "; cin >> y;
    searchPath(x, y);
    cout << endl;
    cout << "Tekan apa saja untuk melanjutkan!";
    getch();
    goto first;
} else if (pil == 4) {
    system("cls");
    printGraph(n);
    cout << "\nMenghapus simpul = "; cin >> x;
    deleteVertex(x);
    cout << endl;
    cout << "Tekan apa saja untuk melanjutkan!";
    getch();
    goto first;
```

Jika kita memilih (2) maka layar akan dibersihkan terlebih dahulu, kemudian akan dipanggil fungsi *printGraph* untuk mencetak graf yang sudah kita buat diawal. Kemudian kita akan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal.

Jika kita memilih (3) maka layar akan dibersihkan terlebih dahulu, kemudian kita akan diminta untuk memasuakan node asal dan node tujuan kemudian dipanggil fungsi *searchPath* yang digunakan untuk mencari jalur tercepat jika ada. Kemudian kita akan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal.

Jika kita memilih (4) maka layar akan dibersihkan terlebih dahulu, kemudian akan dipanggil fungsi *printGraph* untuk mencetak graf yang sudah kita buat diawal, kemudian kita diminta memasukan simpul apa yang akan kita hapus, lalu dipanggil fungsi *deleteVertex* yang digunakan untuk menghapus sebuah simpul yang sudah kita buat di awal. Kemudian kita akan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal.

```
196     } else if (pil == 5) {
197         system("cls");
198         printGraph(n);
199         cout << "\nMenghapus garis antara simpul "; cin >> x;
200         cout << " dengan simpul "; cin >> y;
201         deleteEdge(x, y);
202         cout << endl;
203         cout << "Tekan apa saja untuk melanjutkan!";
204         getch();
205         goto first;
206     } else {
207         cout << "Input yang anda masukkan salah" << endl;
208         char rep;
209         cout << "Apakah anda ingin melanjutkan?"; cin >> rep;
210         if (rep == 'y' || rep == 'Y') {
211             cout << endl;
212             cout << "Tekan apa saja untuk melanjutkan!";
213             getch();
214             goto first;
215         } else {
216             cout << "Program dibuat oleh : Zaldy Seno Yudhanto (2310631170123)\n";
217             return 0;
218         }
219     }
220 }
```

Jika kita memilih (5) maka layar akan dibersihkan terlebih dahulu, kemudian akan dipanggil fungsi *printGraph* untuk mencetak graf yang sudah kita buat diawal, kemudian kita diminta untuk memasukan dua simpul yang nantinya akan dihapus garis antara dua simpul tersebut. Kemudian akan dipanggil fungsi *deleteEdge* yang digunakan untuk menghapus sisi antara dua simpul yang sudah kita buat di awal. Kemudian kita akan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal.

Jika kita memilih selain 1-5 maka akan diberi pesan input yang anda masukan salah kemudian dideklarasikan sebuah variabel baru dengan tipe data char yang diberi nama *rep*y yang digunakan untuk menyimpan jawaban kita. Kemudian akan ditanya apakah ingin melanjutkan program, lalu masuk ke dalam sebuah pengkondisian dimana jika kita menginput y/Y maka kita akan diminta untuk menekan apa saja untuk melanjutkan dengan fungsi *getch()*, lalu akan kembali ke *first* yang sudah kita deklarasikna di awal. Namun jika kita menginput selain y/Y maka program akan selesai.

Ini adalah output dari programnya:

```
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): 3
simpul C (weight): 2

Simpul C terhubung dengan:
simpul A (weight): 1
simpul B (weight): 0

Masukkan weight untuk simpul A: 3
Masukkan weight untuk simpul B: 4
Masukkan weight untuk simpul C: 2

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

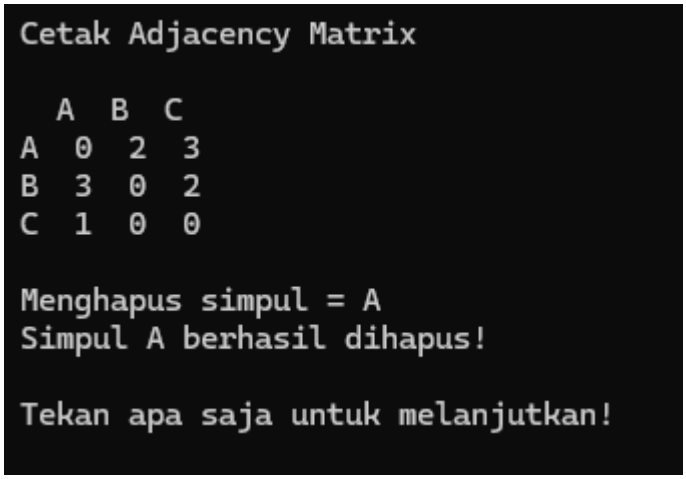
Ini adalah tampilan dari menu 1.

```
Cetak Adjacency Matrix

  A  B  C
A  0  2  3
B  3  0  2
C  1  0  0

Tekan apa saja untuk melanjutkan!
```

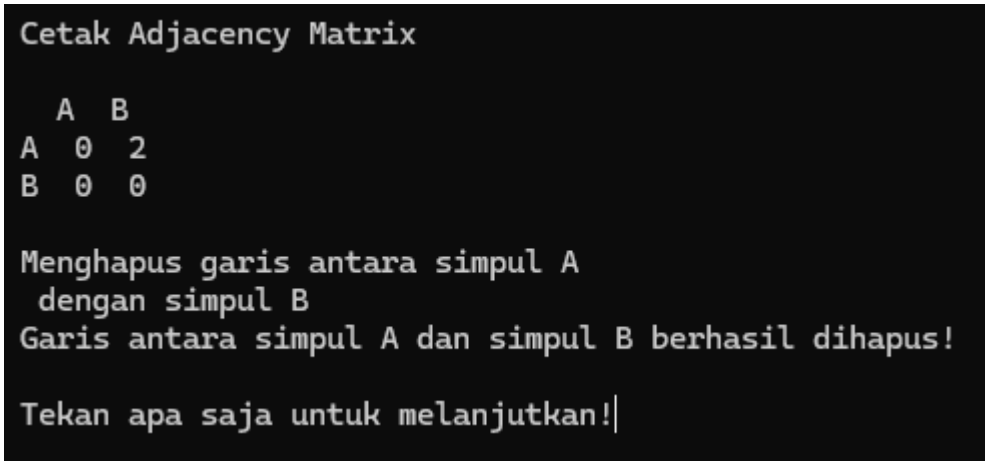
Ini adalah tampilan dari menu 2.

```
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!
```

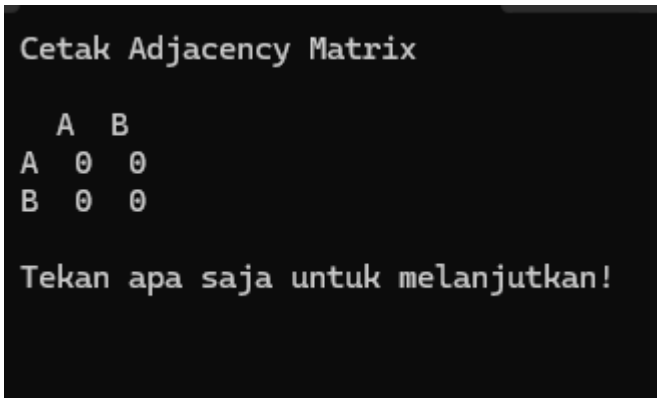
Ini adalah tampilan dari menu 3.



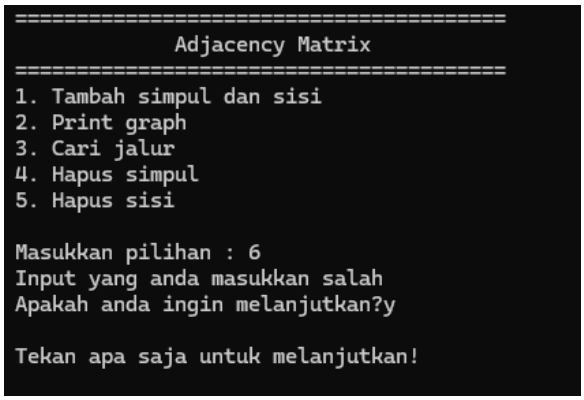
Ini adalah tampilan dari menu 4.

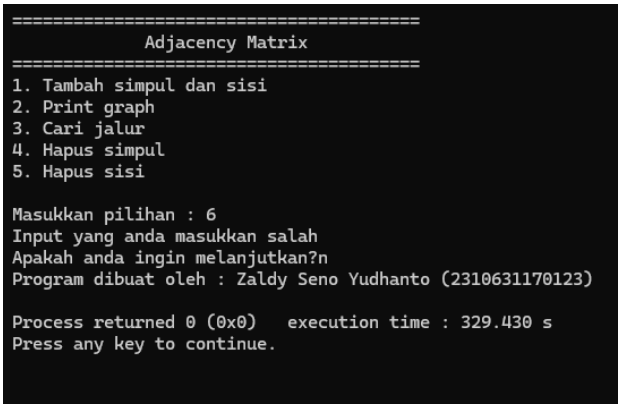


Ini adalah tampilan dari menu 5.



Kemudian ini adalah tampilanya setelah berhasil dihapus.





Kemudian ini adalah tampilan saat saya memasukan inputan selain 1-5.

Ini adalah link github tugas saya:

https://github.com/Dokii06/Semester-2/tree/main