**LAPORAN PROJECT AKHIR**

**PRAKTIKUM**

**STRUKTUR DATA KELAS D**

****

|  |
| --- |
| **Disusun Oleh:**  Nama: Juan Axl Ronaldio Zaka Putra  NIM: 220411100066  Kelas: IF 2D  **Dosen Pengampu:**  Nama: Hermawan, S.T., M.Kom  NIP: 197908282005011002  **Asisten Praktikum:**  Nama: Moh. Fadil Abdillah  NIM: 210411100142 |

**PRODI TEKNIK INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA  
FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**2023**

**MODUL 1 INSTALASI QT CREATOR**

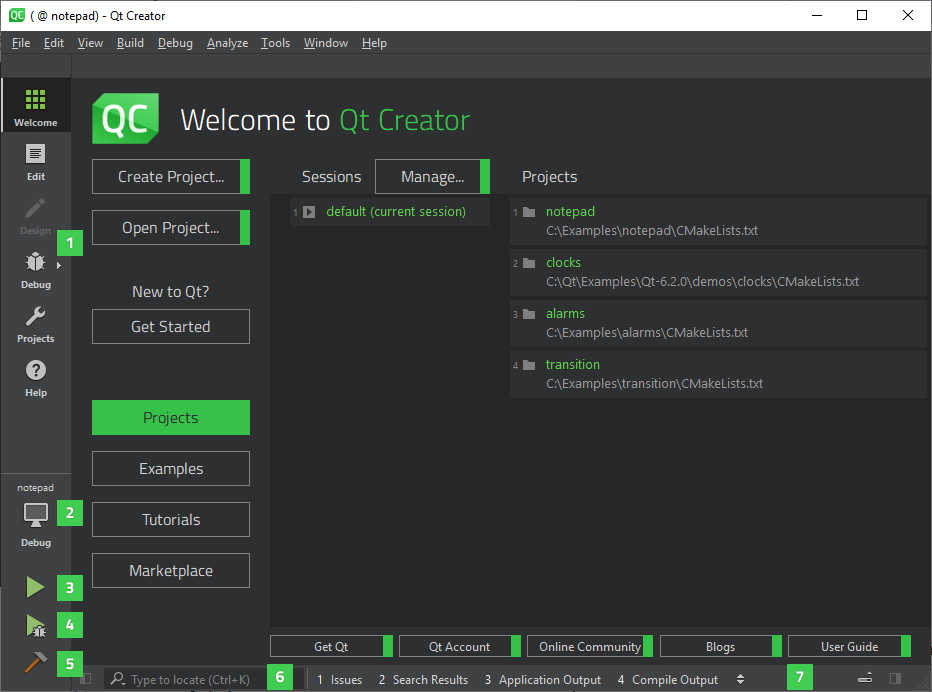
1. **Code Program**

|  |
| --- |
| Silahkan copykan code program ke dalam kolom di bawah ini (code program menggunakan font Courier New size 9,di copy bukan di screenshoot) |

1. **Penjelasan Code Program:**

|  |
| --- |
| Langkah-langkah instalasi Qt:   1. Download installer Qt terlebih dahulu melalui <https://www.qt.io/download>. 2. Setelah selesai terdownload, buka file tersebut. 3. Pada Langkah pertama melakukan install, diharuskan untuk login QT Account, jika sudah memiliki maka langsung masukkan email dan password saja, dan jika belum memiliki akun maka lakukan sign up terlebih dahulu. 4. Setelah login, berikutnya adalah langkah memilih lokasi folder dimana file aplikasi akan ditempatkan. 5. Langkah berikutnya memilih komponen Qt apa saja yang akan diinstal. 6. Langkah berikutnya menyetujui semua perstaratan dan apakah ingin menambahkan shortcut pada Start Menu. 7. Berikutnya tunggu proses download file dan instalasi sampai selesai. 8. Qt Creator sudah dapat digunakan. |

1. **Hasil Running Program:**



**MODUL 2 VARIABEL, KOLEKSI ARRAY, dan KOMPOSISI**

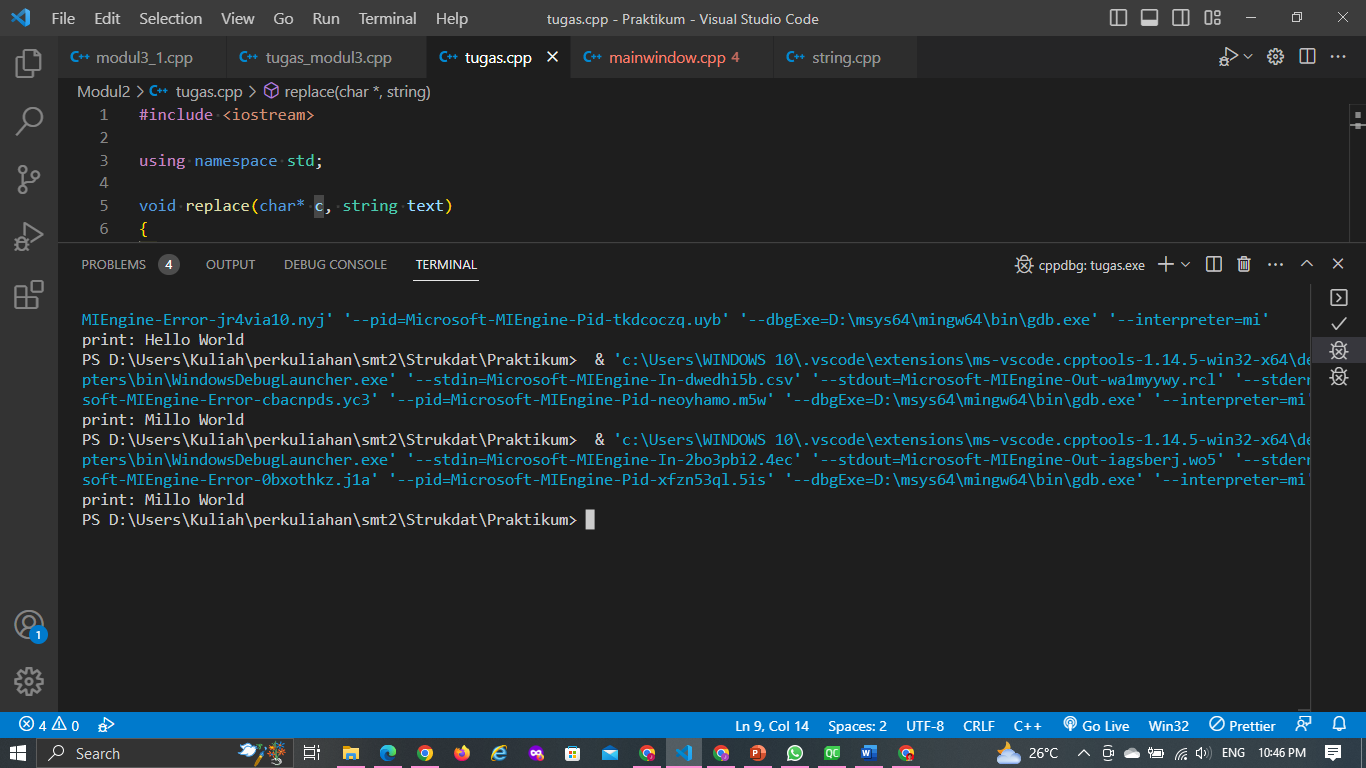
1. **Code Program**

|  |
| --- |
| void replace(char\* c, string text)  {  text = "Mi";  text.copy(c, sizeof(text));  }  int main()  {  string teks = "Hello World";  char\* c = &teks.at(0);  replace(c, teks);  cout << "print: "<< teks << endl;  return 0;  } |

1. **Penjelasan Code Program:**

|  |
| --- |
| Pertama, buat fungsi void replace dengan parameter variabel pointer c bertipe data char dan variabel text bertipe data string. Didalam fungsi tersebut variabel text diisikan string “Mi”, lalu variabel text dilakukan fungsi copy dengan parameter variabel c dan fungsi sizeof(text) atau panjang dari string text.  Berikutnya pada fungsi utama, variabel string teks yang berisi string “Hello World”, lalu variabel pointer c bertipe data char berisi alamat dari variabel teks indeks ke 0 yaitu ‘H’, panggil fungsi replace yang tadi telah dibuat dan isikan parameternya dengan variabel c dan teks, terakhir lakukan print menggunakan cout “print: “ lalu concatenate dengan variabel teks dan endline. |

1. **Hasil Running Program:**



**MODUL 3 KOMPOSISI STRUCT dan CLASS**

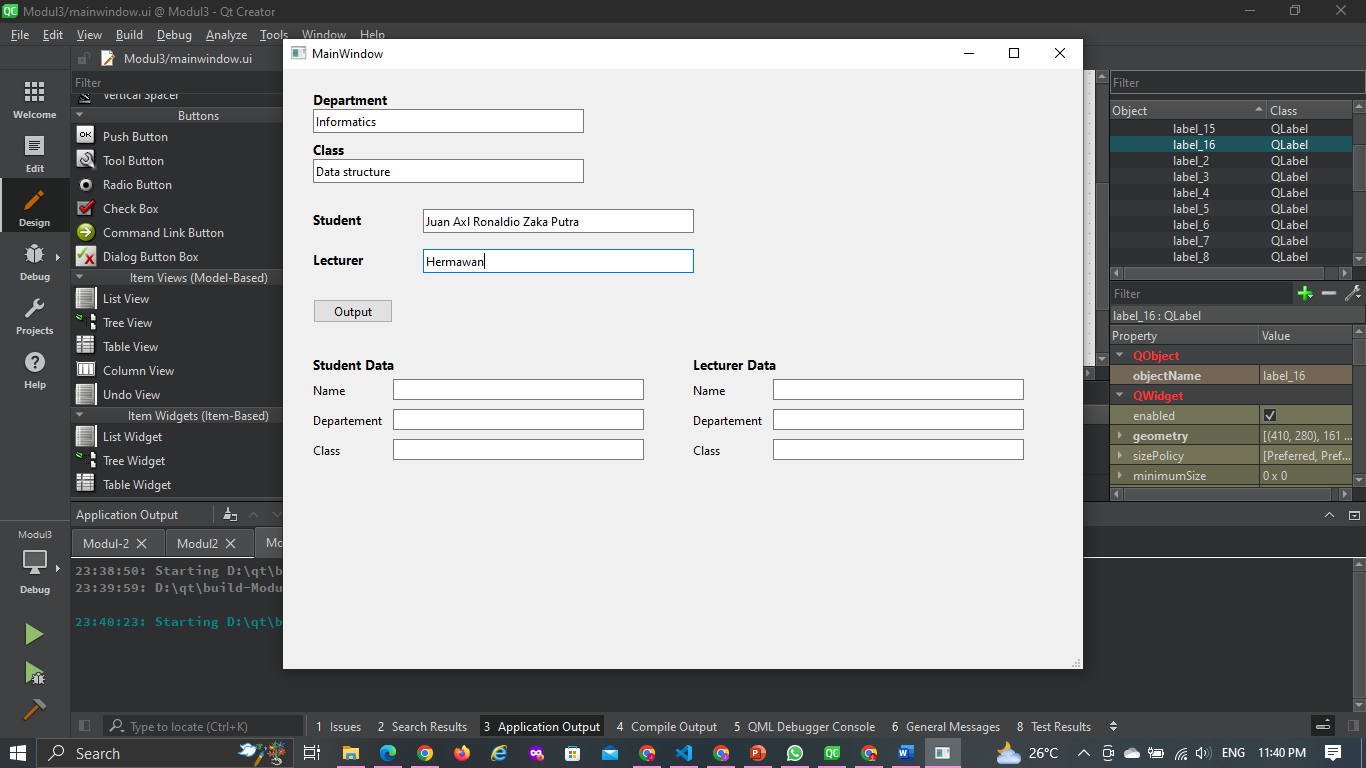
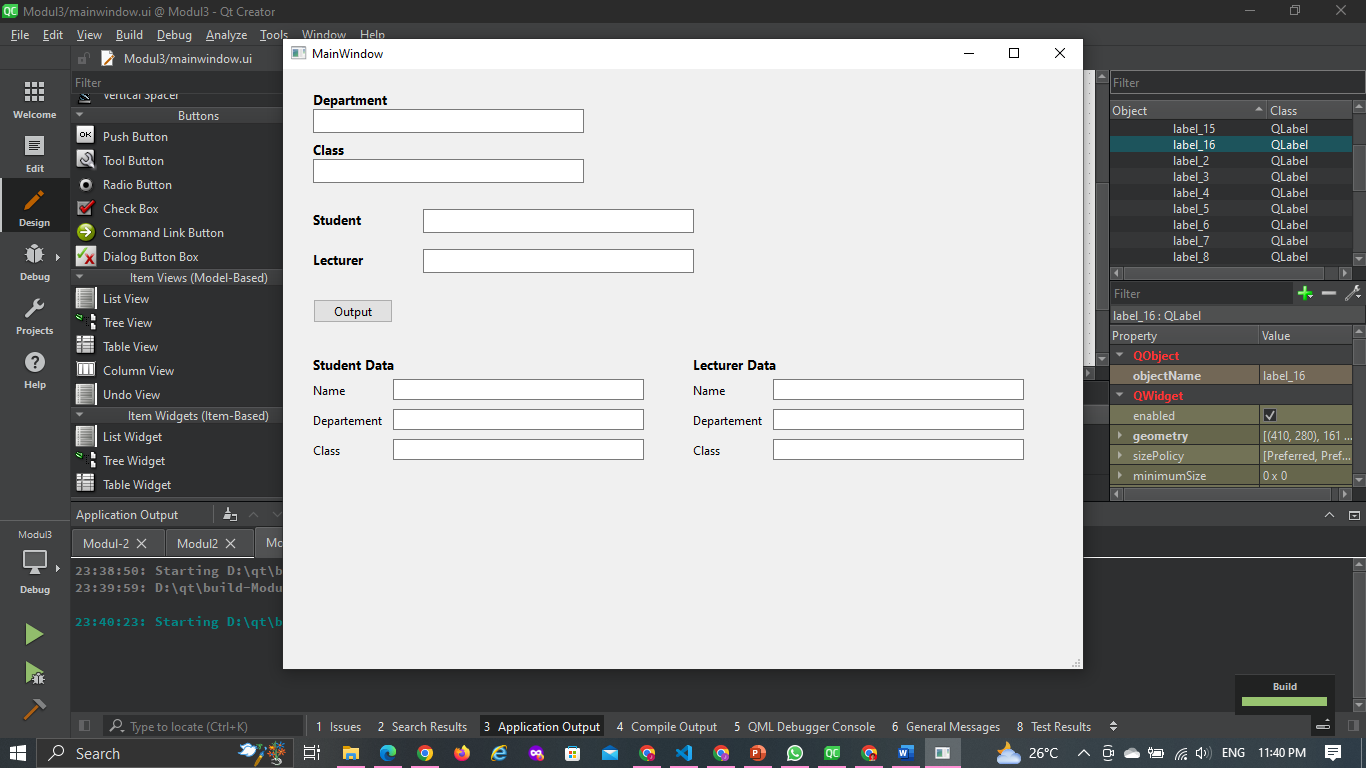
**1. Code Program**

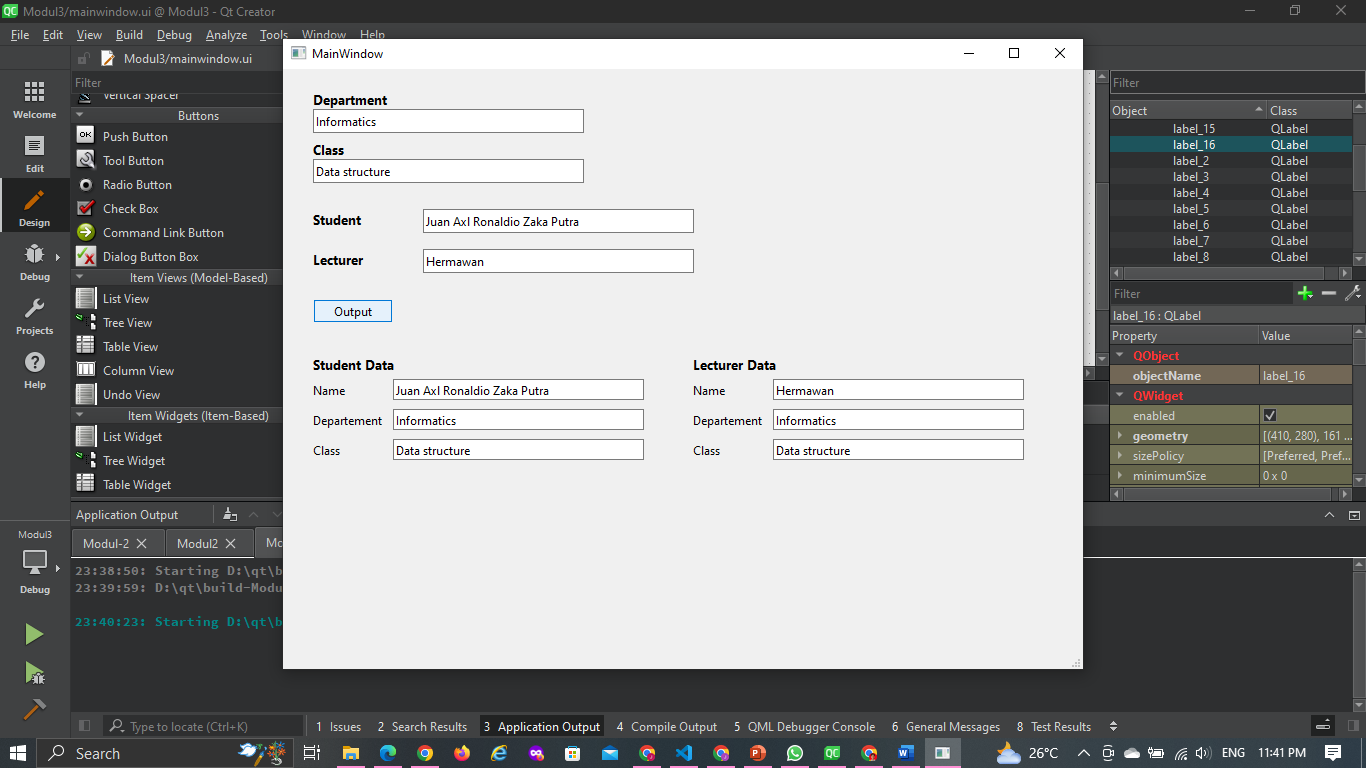
|  |
| --- |
| struct Komposisi {  string name;  string department;  };  typedef Komposisi dataDept;  class Classroom {  public:  dataDept dataLect;  dataDept dataStud;  string classR;  };  void MainWindow::on\_outputBut\_clicked()  {  string clasR = ui->classInput->text().toStdString();  string depart = ui->deptInput->text().toStdString();  Classroom\* c\_room = new Classroom();  c\_room->classR = clasR;  dataDept stud;  stud.name = ui->studentInput->text().toStdString();  stud.department = depart;  dataDept lect;  lect.name = ui->lectInput->text().toStdString();  lect.department = depart;  c\_room->dataLect = lect;  c\_room->dataStud = stud;  ui->studentNameOut->setText(QString::fromStdString(c\_room->dataStud.name));  ui->studentDeptOut->setText(QString::fromStdString(c\_room->dataStud.department));  ui->studentClassOut->setText(QString::fromStdString(c\_room->classR));  ui->lectNameOut->setText(QString::fromStdString(c\_room->dataLect.name));  ui->lectDeptOut->setText(QString::fromStdString(c\_room->dataLect.department));  ui->lectClassOut->setText(QString::fromStdString(c\_room->classR));  } |

**2. Penjelasan Code Program:**

|  |
| --- |
| Pertama, membuat struct dengan nama Komposisi yang didalamnya terdapat variabel string name dan department, lalu melakukan typedef Komposisi menjadi dataDept. Membuat class dengan nama Classroom yang didalamnya terdapat public yang berisi datalect dan dataStud yang bertipe data struct dataDept dan juga string classR.  Fungsi void MainWindow::on\_outputBut\_clicked() yang akan dijalankan ketika elemen pada windows diklik, yang berisikan string clasR dan depart yang berisi inputan user melalui form windows. Deklarasikan variabel pointer c\_room dengan tipe data class Classroom lalu new Classroom() yang artinya buat objek Classroom baru, classR pada variabel c\_room diisikan clasR. Lalu deklarasikan variabel stud dan lect dengan tipe data struct dataDept, variabel name pada stud dan lect diisikan inputan dari user, variabel departement pada stud dan lect diisikan variabel dept yang telah diinputkan sebelumnya.  dataLect dari c\_room diisikan variabel lect dan dataStud dari c\_room diisikan variabel stud. Tampilkan semua data yang telah diinputkan pada elemen window dengan mengakses value dari variabel c\_room->dataStud dan c\_room->datalect. |

**3. Hasil Running Program:**





**MODUL 4 MATRIKS**

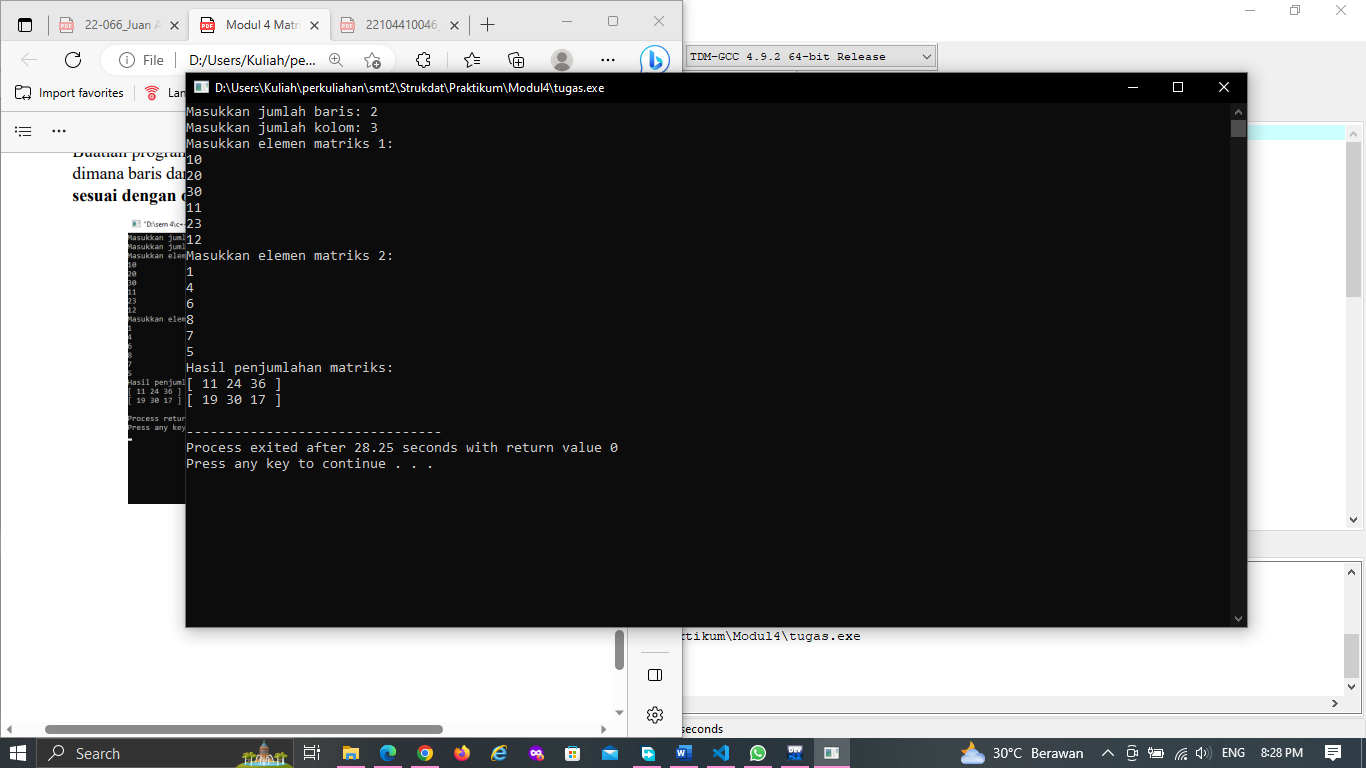
**1. Code Program**

|  |
| --- |
| // fungsi membuat matriks  void makeMatrix(int \*ptrMatrix, int baris, int kolom)  {    int index = 0;    for( int i = 0; i < baris; i++ ) {      for( int j = 0; j < kolom; j++ ) {        cin >> \*(ptrMatrix + index);        index++;      }    }  }  // fungsi menjumlahkan matriks  void tambahMatriks(int \*ptrMatrixJumlah, int \*ptrMatrix1, int \*ptrMatrix2, int baris, int kolom) {    int index = 0;    for( int i = 0; i < baris; i++ ) {      for( int j = 0; j < kolom; j++ ) {        \*(ptrMatrixJumlah + index) = \*(ptrMatrix1 + index) +  \*(ptrMatrix2 + index);        index++;      }    }  }  // fungsi menampilkan matriks  void printMatrix(int \*ptrMatrix, int baris, int kolom)  {    int index = 0;    for( int i = 0; i < baris; i++ ) {      cout << "[ ";      for( int j = 0; j < kolom; j++ ) {        cout << \*(ptrMatrix + index) << " ";        index++;      }      cout << "]" << endl;    }  }  int main()  {    int baris, kolom;    cout << "Masukkan jumlah baris: ";    cin >> baris;    cout << "Masukkan jumlah kolom: ";    cin >> kolom;    int matriks1[baris][kolom], matriks2[baris][kolom], jumlah[baris][kolom];    cout << "Masukkan elemen matriks 1: " << endl;    makeMatrix(\*matriks1, baris, kolom);    cout << "Masukkan elemen matriks 2: " << endl;    makeMatrix(\*matriks2, baris, kolom);    tambahMatriks(\*jumlah, \*matriks1, \*matriks2, baris, kolom);    cout << "Hasil penjumlahan matriks:" << endl;    printMatrix(\*jumlah, baris, kolom);    return 0;  } |

**2. Penjelasan Code Program:**

|  |
| --- |
| Pertama-tama, membuat fungsi void makeMatrix(int \*ptrMatrix, int baris, int kolom) untuk menginputkan nilai matriks dengan menggunakan parameter variabel pointer dari matriks, lalu baris dan juga kolom. Lalu didalam fungsi tersebut membuat variabel index yang berisi nilai 0. Melakukan nested loop menggunakan for dimana looping pertama dilakukan selama variabel i kurang dari variabel baris dan looping kedua dilakukan selama variabel j kurang dari variabel kolom. Didalam looping tersebut melakukan input menggunakan cin yang diinputkan pada \*(ptrMatrix + index) yaitu pointer matriks yang telah dimasukkan dalam parameter pada indeks ke- nilai dari variabel index. Terakhir melakukan increment pada variabel index.  Berikutnya, membuat fungsi void tambahMatriks(int \*ptrMatrixJumlah, int \*ptrMatrix1, int \*ptrMatrix2, int baris, int kolom) untuk menjumlahkan matriks dengan menggunakan parameter variabel pointer matriks penampung hasil penjumlahan lalu matriks pertama dan matriks kedua, dan juga baris dan kolom. Lalu didalam fungsi tersebut membuat variabel index yang berisi nilai 0. Melakukan nested loop menggunakan for dimana looping pertama dilakukan selama variabel i kurang dari variabel baris dan looping kedua dilakukan selama variabel j kurang dari variabel kolom. Didalam looping tersebut melakukan assignment pada \*(ptrMatrixJumlah + index) yaitu pointer matriks jumlah diisikan dengan nilai dari \*(ptrMatrix1 + index) + \*(ptrMatrix2 + index) yaitu penjumlahan dari matriks 1 indeks ke- nilai dari variabel index dan matriks 2 indeks ke- nilai dari variabel index. Terakhir melakukan increment pada variabel index.  Berikutnya, membuat fungsi void printMatrix(int \*ptrMatrix, int baris, int kolom) untuk menampilkan matriks dengan menggunakan parameter variabel pointer dari matriks, lalu baris dan juga kolom. Lalu didalam fungsi tersebut membuat variabel index yang berisi nilai 0. Melakukan nested loop menggunakan for dimana looping pertama dilakukan selama variabel i kurang dari variabel baris dan looping kedua dilakukan selama variabel j kurang dari variabel kolom. Di dalam looping pertama menampilkan “[ “untuk awalan baris, lalu pada looping kedua menampilkan \*(ptrMatrix + index) yaitu matriks indeks ke- nilai dari variabel index. Lalu melakukan increment pada variabel index, dan terakhir menampilkan “]” sebagai penutup baris.  Pada fungsi utama atau fungsi main(), mendeklarasikan variabel baris dan kolom menggunakan tipe data integer, menampilkan “Masukkan jumlah baris: “ lalu menginputkan nilai pada variabel baris, menampilkan “Masukkan jumlah kolom: “ lalu menginputkan nilai pada variabel kolom. Selanjutnya, mendeklarasikan variabel matriks1, matriks2, jumlah dengan tipe data integer array dengan banyak index baris dan kolom. Variabel matriks1 digunakan untuk menyimpan nilai array matriks pertama, variabel matriks2 digunakan untuk menyimpan nilai array matriks kedua, dan variabel jumlah digunakan untuk menyimpan nilai array hasil penjumlahan matriks pertama dan matriks kedua. Selanjutnya, menampilkan “Masukkan elemen matriks 1: “ lalu memanggil fungsi makeMatrix(\*matriks1, baris, kolom) dan mengisikan parameternya dengan pointer variabel matriks1, baris, dan kolom untuk menginputkan nilai pada variabel matriks1. Melakukan hal serupa pada matriks2. Berikutnya, Memanggil fungsi tambahMatriks(\*jumlah, \*matriks1, \*matriks2, baris, kolom) dan mengisikan parameternya dengan pointer variabel jumlah, pointer variabel matriks1, pointer variabel matriks2, baris, dan kolom untuk menjumlahkan kedua buah matriks dan dimasukkan kedalam variabel jumlah. Terakhir, menampilkan “Hasil penjumlahan matriks:“ lalu memanggil fungsi printMatrix(\*jumlah, baris, kolom) dan mengisikan parameternya dengan pointer variabel jumlah, baris, dan kolom untuk menampilkan setiap anggota dari variabel matriks jumlah. |

**3. Hasil Running Program:**



**MODUL 5 LINKED LIST**

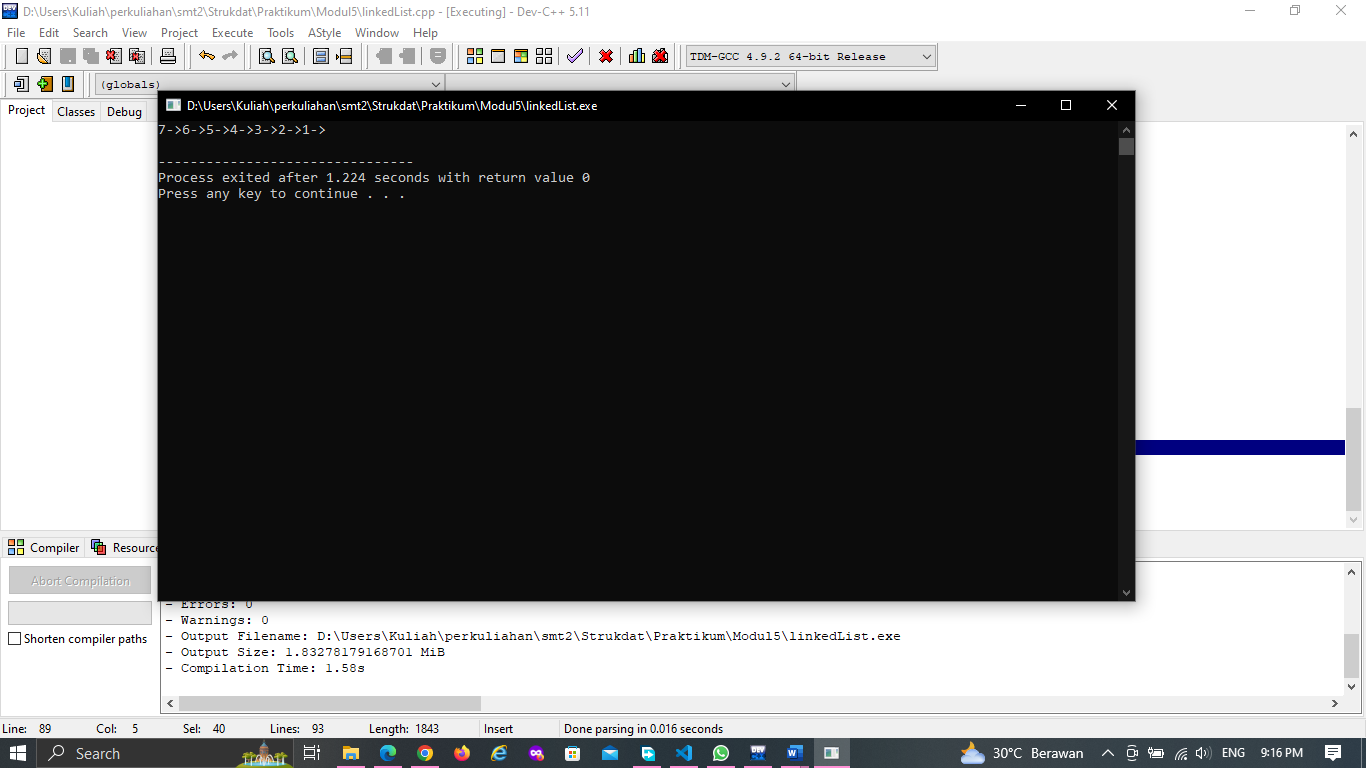
**1. Code Program**

|  |
| --- |
| struct Node {  int data;  Node\* next;  };  void tambahNodeAwal(Node\*\* head, int dataBaru)  {  // Buat node baru  Node\* newNode = new Node;  newNode->data = dataBaru;  // Jika linked list masih kosong, node baru menjadi head  if (\*head == NULL) {  \*head = newNode;  newNode->next = NULL;  return;  }    // tambahkan node baru ke elemen pertama  newNode->next = \*head;  \*head = newNode;  }  void cetakList(Node\* head)  {  while (head != NULL) {  cout << head->data << "->";  head = head->next;  }  cout << endl;  }  int main()  {  Node\* head = NULL;  tambahNodeAwal(&head, 1);  tambahNodeAwal(&head, 2);  tambahNodeAwal(&head, 3);  tambahNodeAwal(&head, 4);  tambahNodeAwal(&head, 5);  tambahNodeAwal(&head, 6);  tambahNodeAwal(&head, 7);  cetakList(head);  return 0;  } |

**2. Penjelasan Code Program:**

|  |
| --- |
| Membuat fungsi void tambahNodeAwal(Node\*\* head, int dataBaru) untuk menambahkan node pada awal linked list dengan menggunakan parameter variabel pointer head dan integer dataBaru yaitu data yang akan dimasukkan. Lalu didalam fungsi tersebut membuat node baru pada variabel newNode dengan membuat objek Node baru. Melakukan pengkondisian jika variabel pointer head masih belum terisi atau NULL, maka head adalah newNode dan next dari newNode adalah NULL, lalu return agar fungsi langsung berhenti. Jika tidak memenuhi pengkondisian tersebut maka akan langsung menjalankan program yang diluar pengkondisian yaitu, next dari newNode adalah head dan head dipindahkan ke newNode.  Menjalankan program pada utama atau fungsi main(), membuat variabel head dengan tipe data struct Node dan diisikan NULL karena linked list masih kosong. Memanggil fungsi tambahNodeAwal(&head, int nilai) dan mengisikan parameternya dengan alamat dari head dan nilai yang akan dimasukkan untuk melakukan insert pada linked list, dilakukan sebanyak 7 kali dengan memasukkan nilai berurutan dari angka 1-7. Terakhir memanggil fungsi cetakList(head) yang telah dibuat sebelumnya dan mengisikan parameternya dengan variabel head untuk menampilkan setiap data pada linked list. |

**3. Hasil Running Program:**



**MODUL 6 TREE**

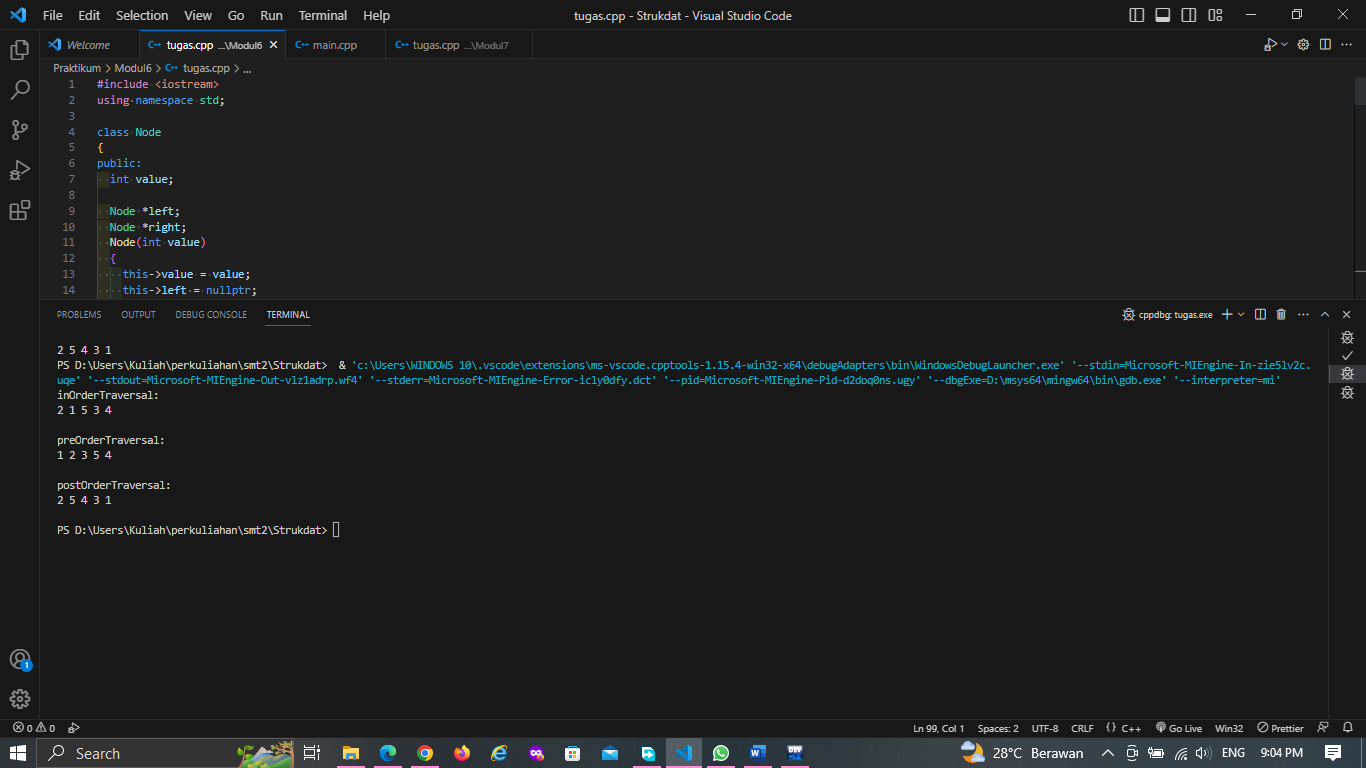
**1. Code Program**

|  |
| --- |
| #include <iostream>  using namespace std;  class Node  {  public:  int value;  Node \*left;  Node \*right;  Node(int value)  {  this->value = value;  this->left = nullptr;  this->right = nullptr;  }  };  void inOrderTraversal(Node \*root)  {  if (root == nullptr)  return;  inOrderTraversal(root->left);  cout << root->value << " ";  inOrderTraversal(root->right);  }  // preOrder Traversal  void preOrderTraversal(Node \*root)  {  if (root == nullptr)  return;  cout << root->value << " ";  preOrderTraversal(root->left);  preOrderTraversal(root->right);  }  // postOrder Traversal  void postOrderTraversal(Node \*root)  {  if (root == nullptr)  return;  postOrderTraversal(root->left);  postOrderTraversal(root->right);  cout << root->value << " ";  }  Node \*createBinaryTree()  {  Node \*root = new Node(1);  return root;  }  void addChildren(Node \*root)  {  Node \*leftChild = new Node(2);  Node \*rightChild = new Node(3);  Node \*nodebaru1 = new Node(4);  Node \*nodebaru2 = new Node(5);  root->left = leftChild;  root->right = rightChild;  rightChild->right = nodebaru1;  rightChild->left = nodebaru2;  }  int main()  {  Node \*root = createBinaryTree();  addChildren(root);  cout << "inOrderTraversal:" << endl;  inOrderTraversal(root);  cout << "\n" << endl;    cout << "preOrderTraversal:" << endl;  preOrderTraversal(root);  cout << "\n" << endl;  cout << "postOrderTraversal:" << endl;  postOrderTraversal(root);  cout << "\n" << endl;  delete root->left;  delete root->right;  delete root;  return 0;  } |

**2. Penjelasan Code Program:**

|  |
| --- |
| Membuat fungsi void preOrderTraversal(Node \*root) , dengan parameter root dari binary tree. Fungsi ini digunakan untuk melakukan traversal pada binary tree secara pre order yaitu mengunjungi root terlebih dahulu lalu menelusuri left child dan terakhir menelusuri right child. Pada fungsi tersebut, pertama lakukan pengkondisian menggunakan if, jika root bernilai null maka fungsi akan langsung return atau berhenti tanpa mengembalikan apa-apa. Diluar pengkondisian tersebut, mencetak value dari root lalu spasi, berikutnya melakukan rekursif dengan parameter left child dari root, dan terakhir rekursif dengan parameter right child dari root.  Membuat fungsi void postOrderTraversal(Node \*root) , dengan parameter root dari binary tree. Fungsi ini digunakan untuk melakukan traversal pada binary tree secara post order yaitu menelusuri left child terlebih dahulu lalu menelusuri right child dan terakhir baru mengunjungi root. Pada fungsi tersebut, pertama lakukan pengkondisian menggunakan if, jika root bernilai null maka fungsi akan langsung return atau berhenti tanpa mengembalikan apa-apa. Diluar pengkondisian tersebut, melakukan rekursif dengan parameter left child dari root, berikutnya rekursif dengan parameter right child dari root, dan terakhir mencetak value dari root lalu spasi. |

**3. Hasil Running Program:**



**MODUL 7 GRAPH ADJACENCY LIST**

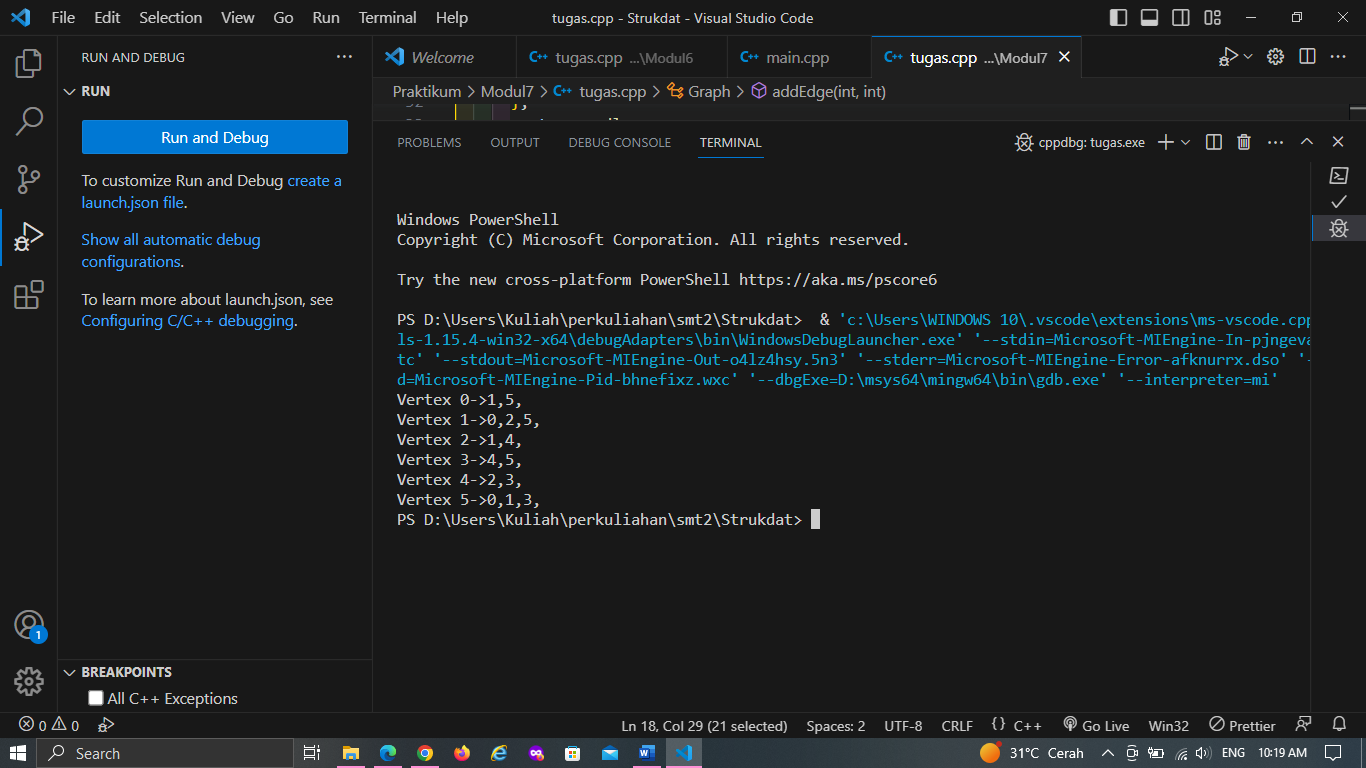
**1. Code Program**

|  |
| --- |
| #include <iostream>  #include <list>  using namespace std;  class Graph  {  int V;  // array of list  list<int> \*l;  public:  Graph(int V)  {  this->V = V;  l = new list<int>[V];  }  void addEdge(int x, int y)  {  l[x].push\_back(y);  l[y].push\_back(x);  }    void printGraph()  {  for (int i = 0; i < V; i++)  {  cout << "Vertex " << i << "->";  for (int num : l[i])  {  cout << num << ",";  };  cout << endl;  }  }  };  int main()  {  Graph g(6);  g.addEdge(0, 1);  g.addEdge(0, 5);  g.addEdge(1, 2);  g.addEdge(1, 5);  g.addEdge(2, 4);  g.addEdge(3, 4);  g.addEdge(3, 5);  g.printGraph();  return 0;  } |

**2. Penjelasan Code Program:**

|  |
| --- |
| Setelah melakukan include standard library iostream, menambahkan juga include standard library list agar dapat menggunakan array of list. Membuat class dengan nama Graph, lalu didalamnya terdapat variabel V bertipe data integer untuk menyimpan banyaknya vertex, dan juga variabel pointer l menggunakan array of list integer. Pada bagian public class Graph tersebut, membuat fungsi Graph(int V) sebagai konstruktor, addEdge(int x, int y) untuk menambahkan edge, dan printGraph() untuk menampilkan data graph.  Fungsi Graph(int V) menggunakan parameter integer V, pada fungsi tersebut this-> V diisikan oleh argumen yang telah dimasukkan pada parameter V, artinya V pada objek graph diisikan oleh parameter, lalu pada variabel l membuat array of list baru dengan jumlah sebanyak V. Berikutnya pada fungsi addEdge(int x, int y) dengan parameter integer x dan y, pada fungsi tersebut l[x] atau list index ke x dilakukan push\_back(y) atau menambahkan nilai y pada list tersebut, dan l[y] atau list index ke y dilakukan push\_back(x) atau menambahkan nilai x pada list tersebut.  Fungsi printGraph(), menggunakan for i selama i kurang dari V, lalu menampilkan vertex i -> dan menggunakan for each atau perulangan setiap nilai dari l[i] (l index ke-i) untuk menampilkan semua nilai pada list tersebut. Pada fungsi main, membuat Graph pada g dan mengisikan parameternya dengan jumlah vertex yang akan dibuat. Lalu memanggil fungsi addEdge(x,y) pada g dan mengisikan parameternya dengan vertex yang akan diberi edge, lakukan pada semua vertex yang akan diberi edge. Terakhir tampilkan graph yang telah dibuat dengan memanggil fungsi printGraph() pada g. |

**3. Hasil Running Program:**

****

**MODUL 8 GRAPH ADJACENCY MATRIKS**

**1. Code Program**

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  void printAdjacencyMatrix(vector<vector<int>> &adjMatrix)  {  int n = adjMatrix.size();  for (int i = 0; i < n; i++)  {  cout << '[';  for (int j = 0; j < n; j++)  {  cout << adjMatrix[i][j] << " ";  }  cout << ']';  cout << endl;  }  }  int main()  {  int n = 7; // Jumlah simpul  vector<vector<int>> adjMatrix(n, vector<int>(n, 0));  // Mengatur sisi-sisi pada adjacency matrix  adjMatrix[0][1] = 1;  adjMatrix[0][2] = 1;  adjMatrix[0][3] = 1;  adjMatrix[1][0] = 1;  adjMatrix[1][2] = 1;  adjMatrix[1][4] = 1;  adjMatrix[2][0] = 1;  adjMatrix[2][1] = 1;  adjMatrix[2][3] = 1;  adjMatrix[2][4] = 1;  adjMatrix[2][5] = 1;  adjMatrix[2][6] = 1;  adjMatrix[3][0] = 1;  adjMatrix[3][2] = 1;  adjMatrix[3][5] = 1;  adjMatrix[3][6] = 1;  adjMatrix[4][1] = 1;  adjMatrix[4][2] = 1;  adjMatrix[5][2] = 1;  adjMatrix[5][3] = 1;  adjMatrix[6][2] = 1;  adjMatrix[6][3] = 1;  cout << "Adjacency matrix:" << endl;  printAdjacencyMatrix(adjMatrix);  return 0;  } |

**2. Penjelasan Code Program:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Berdasarkan gambar data graph pada soal, maka dapat diubah kedalam tabel matriks sebagai berikut:  **Permisalan:**  **0 = A**  **1 = B**  **2 = C**  **3 = D**  **4 = E**  **5 = F**  **6 = G**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **0** | **1** | **1** | **1** | **0** | **0** | **0** | | **1** | **0** | **1** | **0** | **1** | **0** | **0** | | **1** | **1** | **0** | **1** | **1** | **1** | **1** | | **1** | **0** | **1** | **0** | **0** | **1** | **1** | | **0** | **1** | **1** | **0** | **0** | **0** | **0** | | **0** | **0** | **1** | **1** | **0** | **0** | **0** | | **0** | **0** | **1** | **1** | **0** | **0** | **0** |   Setelah didapatkan tabel tersebut, maka dapat mengatur sisi-sisi pada adjacency matriks dengan menginputkan nilai matriks pada index sesuai dengan posisi baris dan kolom pada tabel yang bernilai 1 dan pada index matriks tersebut juga diberi nilai 1. |

**3. Hasil Running Program:**

