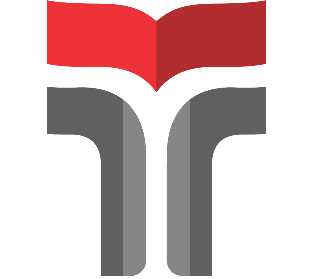
**LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA**

**MODUL III**

**SINGLE AND DOUBLE LINKED LIST**



# Disusun Oleh :

NAMA : Aulia Radix Putra Winarko  
NIM : 2311102056

# Dosen

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA   
FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

1. **Dasar Teori**

* **Single Linked List**

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail. Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List. Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.

* **Double Linked List**

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya. Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

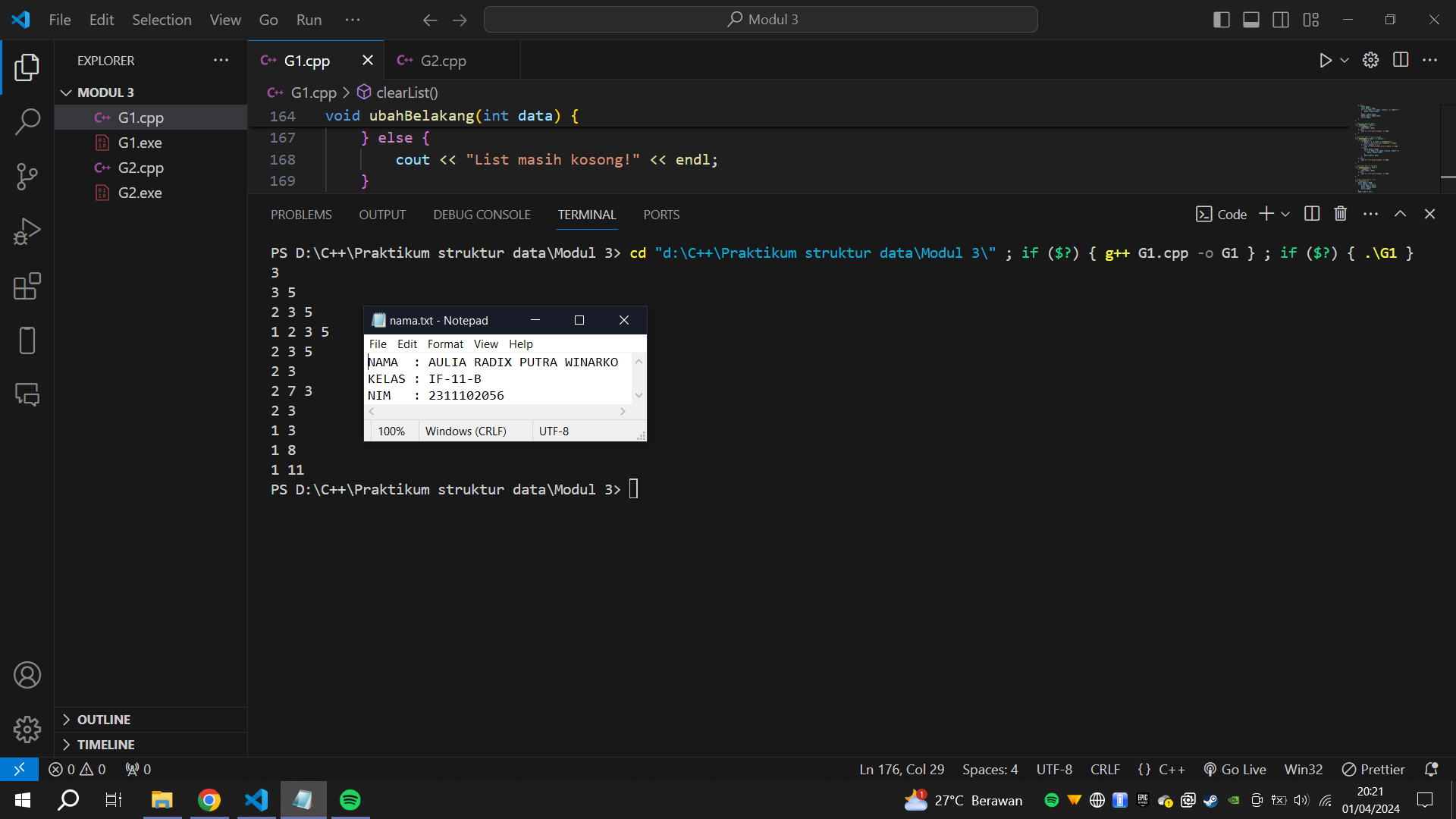
1. **Guided** [

Guided 1

Latihan Single Linked List

|  |
| --- |
| #include <iostream>  using namespace std;  // Deklarasi Struct Node  struct Node {      int data;      Node\* next;  };  Node\* head;  Node\* tail;  // Inisialisasi Node  void init() {      head = NULL;      tail = NULL;  }  // Pengecekan apakah list kosong  bool isEmpty() {      return head == NULL;  }  // Tambah Node di depan  void insertDepan(int nilai) {      Node\* baru = new Node;      baru->data = nilai;      baru->next = NULL;      if (isEmpty()) {          head = tail = baru;      } else {          baru->next = head;          head = baru;      }  }  // Tambah Node di belakang  void insertBelakang(int nilai) {      Node\* baru = new Node;      baru->data = nilai;      baru->next = NULL;      if (isEmpty()) {          head = tail = baru;      } else {          tail->next = baru;          tail = baru;      }  }  // Hitung jumlah Node di list  int hitungList() {      Node\* hitung = head;      int jumlah = 0;      while (hitung != NULL) {          jumlah++;          hitung = hitung->next;      }      return jumlah;  }  // Tambah Node di posisi tengah  void insertTengah(int data, int posisi) {      if (posisi < 1 || posisi > hitungList()) {          cout << "Posisi diluar jangkauan" << endl;      } else if (posisi == 1) {          cout << "Posisi bukan posisi tengah" << endl;      } else {          Node\* baru = new Node();          baru->data = data;          Node\* bantu = head;          int nomor = 1;          while (nomor < posisi - 1) {              bantu = bantu->next;              nomor++;          }          baru->next = bantu->next;          bantu->next = baru;      }  }  // Hapus Node di depan  void hapusDepan() {      if (!isEmpty()) {          Node\* hapus = head;          if (head->next != NULL) {              head = head->next;              delete hapus;          } else {              head = tail = NULL;              delete hapus;          }      } else {          cout << "List kosong!" << endl;      }  }  // Hapus Node di belakang  void hapusBelakang() {      if (!isEmpty()) {          if (head != tail) {              Node\* hapus = tail;              Node\* bantu = head;              while (bantu->next != tail) {                  bantu = bantu->next;              }              tail = bantu;              tail->next = NULL;              delete hapus;          } else {              head = tail = NULL;          }      } else {          cout << "List kosong!" << endl;      }  }  // Hapus Node di posisi tengah  void hapusTengah(int posisi) {      if (posisi < 1 || posisi > hitungList()) {          cout << "Posisi diluar jangkauan" << endl;      } else if (posisi == 1) {          cout << "Posisi bukan posisi tengah" << endl;      } else {          Node\* hapus;          Node\* bantu = head;          for (int nomor = 1; nomor < posisi - 1; nomor++) {              bantu = bantu->next;          }          hapus = bantu->next;          bantu->next = hapus->next;          delete hapus;      }  }  // Ubah data Node di depan  void ubahDepan(int data) {      if (!isEmpty()) {          head->data = data;      } else {          cout << "List masih kosong!" << endl;      }  }  // Ubah data Node di posisi tengah  void ubahTengah(int data, int posisi) {      if (!isEmpty()) {          if (posisi < 1 || posisi > hitungList()) {              cout << "Posisi di luar jangkauan" << endl;          } else if (posisi == 1) {              cout << "Posisi bukan posisi tengah" << endl;          } else {              Node\* bantu = head;              for (int nomor = 1; nomor < posisi; nomor++) {                  bantu = bantu->next;              }              bantu->data = data;          }      } else {          cout << "List masih kosong!" << endl;      }  }  // Ubah data Node di belakang  void ubahBelakang(int data) {      if (!isEmpty()) {          tail->data = data;      } else {          cout << "List masih kosong!" << endl;      }  }  // Hapus semua Node di list  void clearList() {      Node\* bantu = head;      while (bantu != NULL) {          Node\* hapus = bantu;          bantu = bantu->next;          delete hapus;      }      head = tail = NULL;      cout << "List berhasil terhapus!" << endl;  }  // Tampilkan semua data Node di list  void tampil() {      if (!isEmpty()) {          Node\* bantu = head;          while (bantu != NULL) {              cout << bantu->data << " ";              bantu = bantu->next;          }          cout << endl;      } else {          cout << "List masih kosong!" << endl;      }  }  int main() {      init();      insertDepan(3); tampil();      insertBelakang(5); tampil();      insertDepan(2); tampil();      insertDepan(1); tampil();      hapusDepan(); tampil();      hapusBelakang(); tampil();      insertTengah(7, 2); tampil();      hapusTengah(2); tampil();      ubahDepan(1); tampil();      ubahBelakang(8); tampil();      ubahTengah(11, 2); tampil();      return 0;  } |

Screenshots Output



Deskripsi:

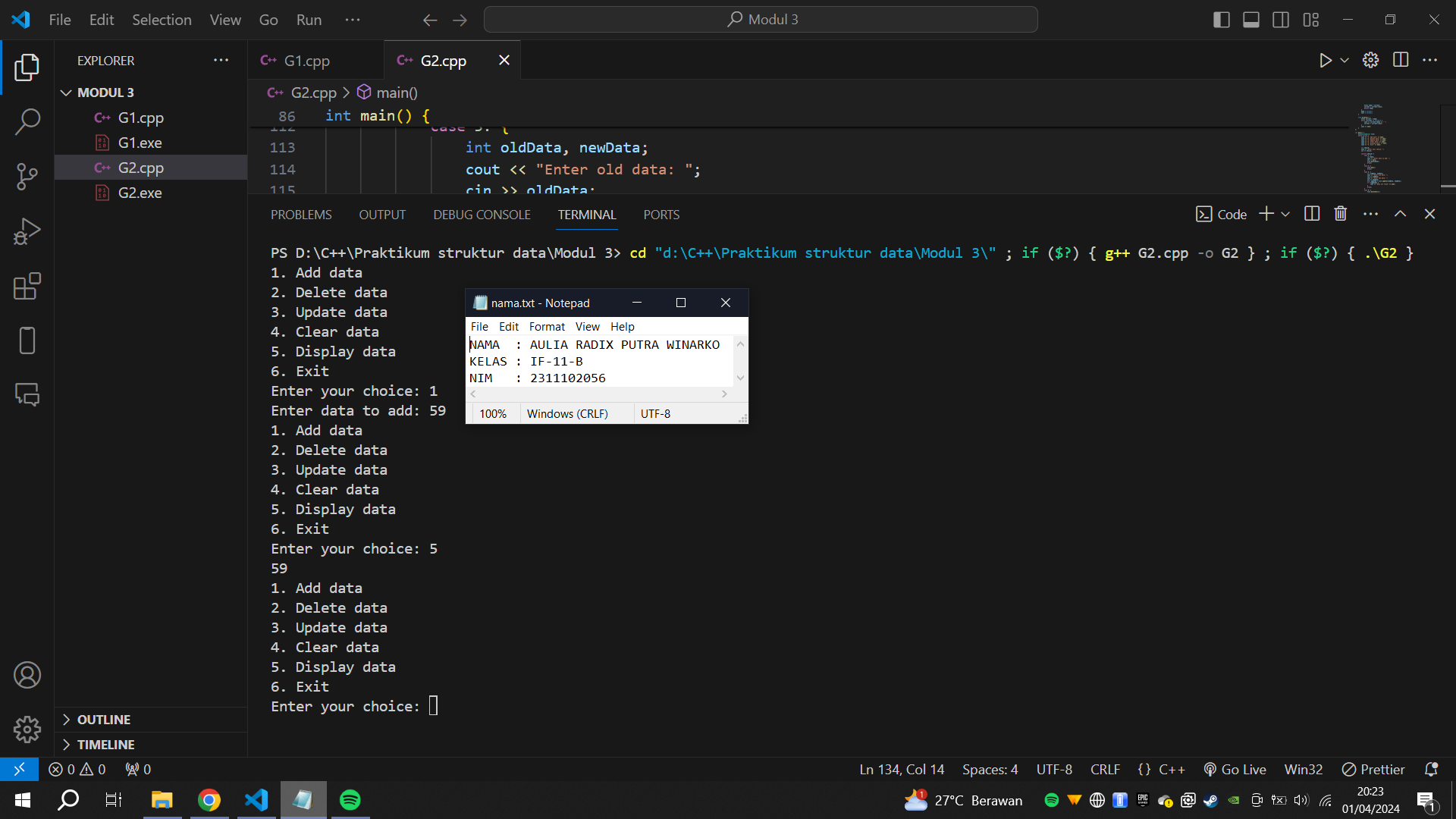
Program C++ di atas adalah sebuah program yang memungkinkan user untuk memasukkan elemen-elemen ke dalam sebuah array tiga dimensi dan kemudian menampilkan array tersebut.

Guided 2

Latihan Double Linked List

|  |
| --- |
| #include <iostream>  using namespace std;  class Node {  public:      int data;      Node\* prev;      Node\* next;  };  class DoublyLinkedList {  public:      Node\* head;      Node\* tail;      DoublyLinkedList() {          head = nullptr;          tail = nullptr;      }      void push(int data) {          Node\* newNode = new Node;          newNode->data = data;          newNode->prev = nullptr;          newNode->next = head;            if (head != nullptr) {              head->prev = newNode;          } else {              tail = newNode;          }            head = newNode;      }      void pop() {          if (head == nullptr) {              return;          }          Node\* temp = head;          head = head->next;            if (head != nullptr) {              head->prev = nullptr;          } else {              tail = nullptr;          }            delete temp;      }      bool update(int oldData, int newData) {          Node\* current = head;            while (current != nullptr) {              if (current->data == oldData) {                  current->data = newData;                  return true;              }              current = current->next;          }          return false;      }      void deleteAll() {          Node\* current = head;          while (current != nullptr) {              Node\* temp = current;              current = current->next;              delete temp;          }          head = nullptr;          tail = nullptr;      }      void display() {          Node\* current = head;          while (current != nullptr) {              cout << current->data << " ";              current = current->next;          }          cout << endl;      }  };  int main() {      DoublyLinkedList list;      while (true) {          cout << "1. Add data" << endl;          cout << "2. Delete data" << endl;          cout << "3. Update data" << endl;          cout << "4. Clear data" << endl;          cout << "5. Display data" << endl;          cout << "6. Exit" << endl;            int choice;          cout << "Enter your choice: ";          cin >> choice;            switch (choice) {              case 1: {                  int data;                  cout << "Enter data to add: ";                  cin >> data;                  list.push(data);                  break;              }              case 2: {                  list.pop();                  break;              }              case 3: {                  int oldData, newData;                  cout << "Enter old data: ";                  cin >> oldData;                  cout << "Enter new data: ";                  cin >> newData;                  bool updated = list.update(oldData, newData);                  if (!updated) {                      cout << "Data not found" << endl;                  }                  break;              }              case 4: {                  list.deleteAll();                  break;              }              case 5: {                  list.display();                  break;              }              case 6: {                  return 0;              }              default: {                  cout << "Invalid choice" << endl;                  break;              }          }      }      return 0;  } |

Screenshots Output



Deskripsi:

Program C++ di atas merupakan implementasi dari sebuah Doubly Linked List (atau senarai berantai ganda). Doubly Linked List adalah struktur data linear di mana setiap node memiliki dua pointer, yaitu satu untuk menunjuk ke node sebelumnya (prev) dan satu untuk menunjuk ke node berikutnya (next).

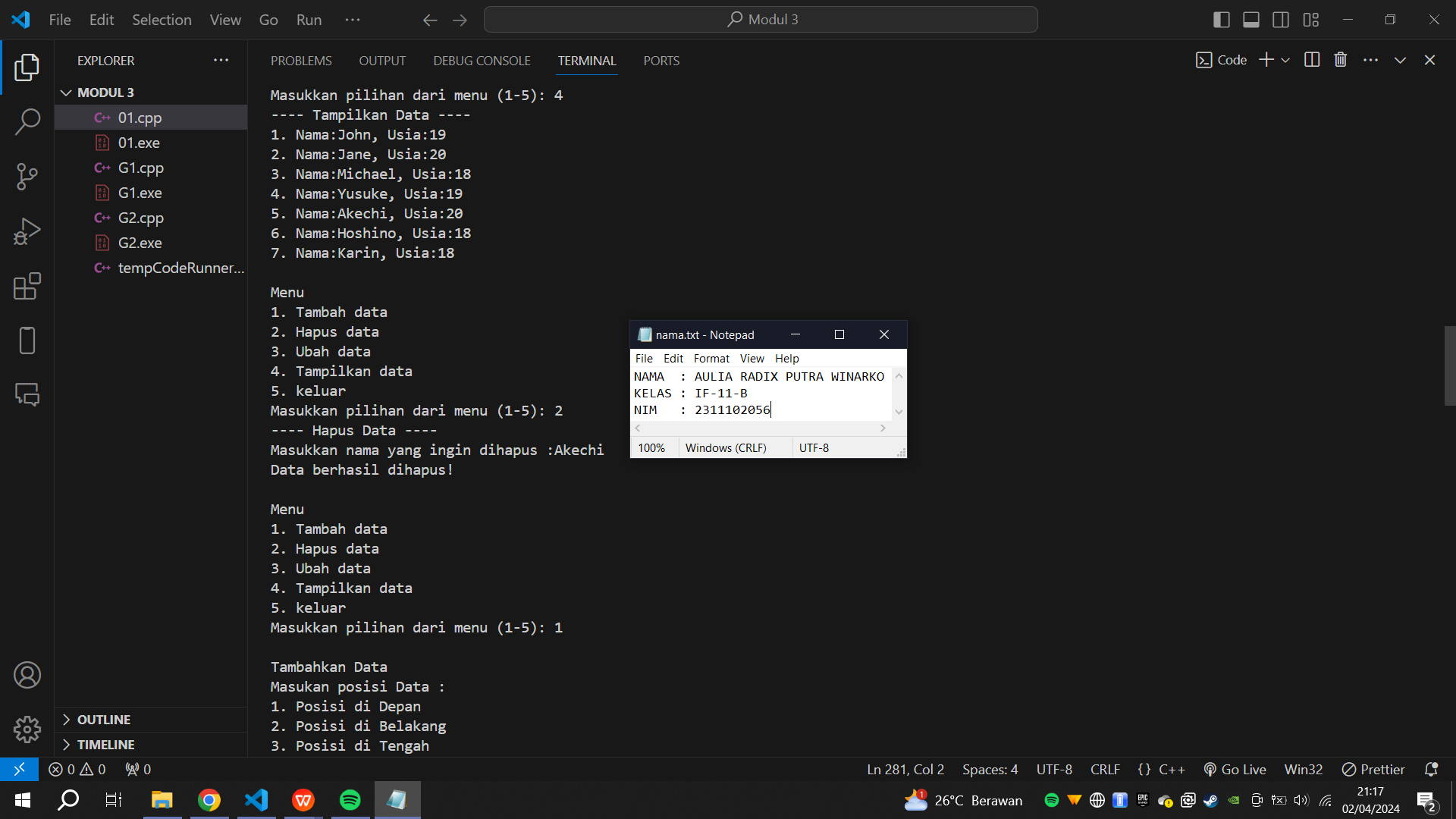
1. **Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)**

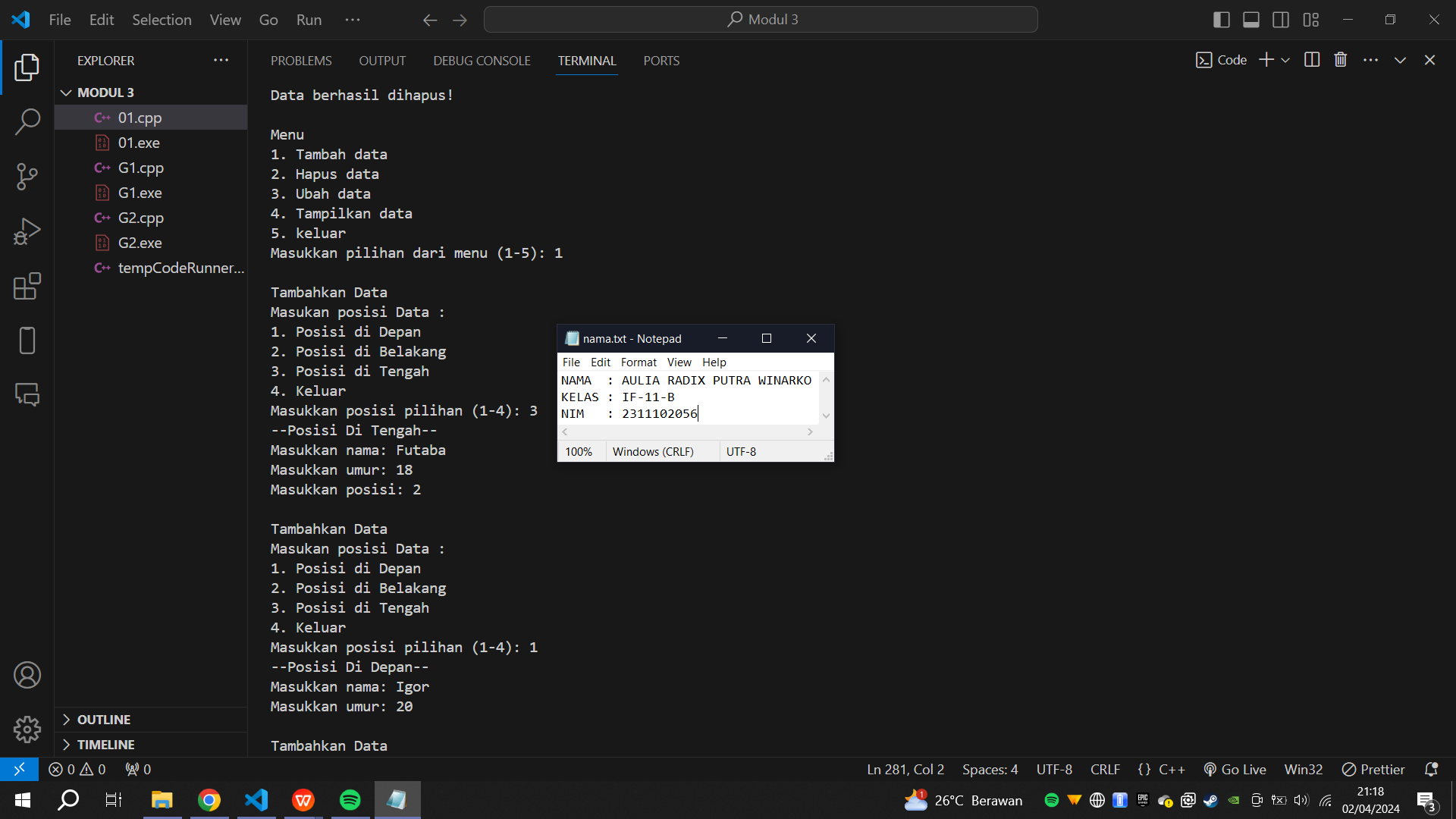
Unguided 1

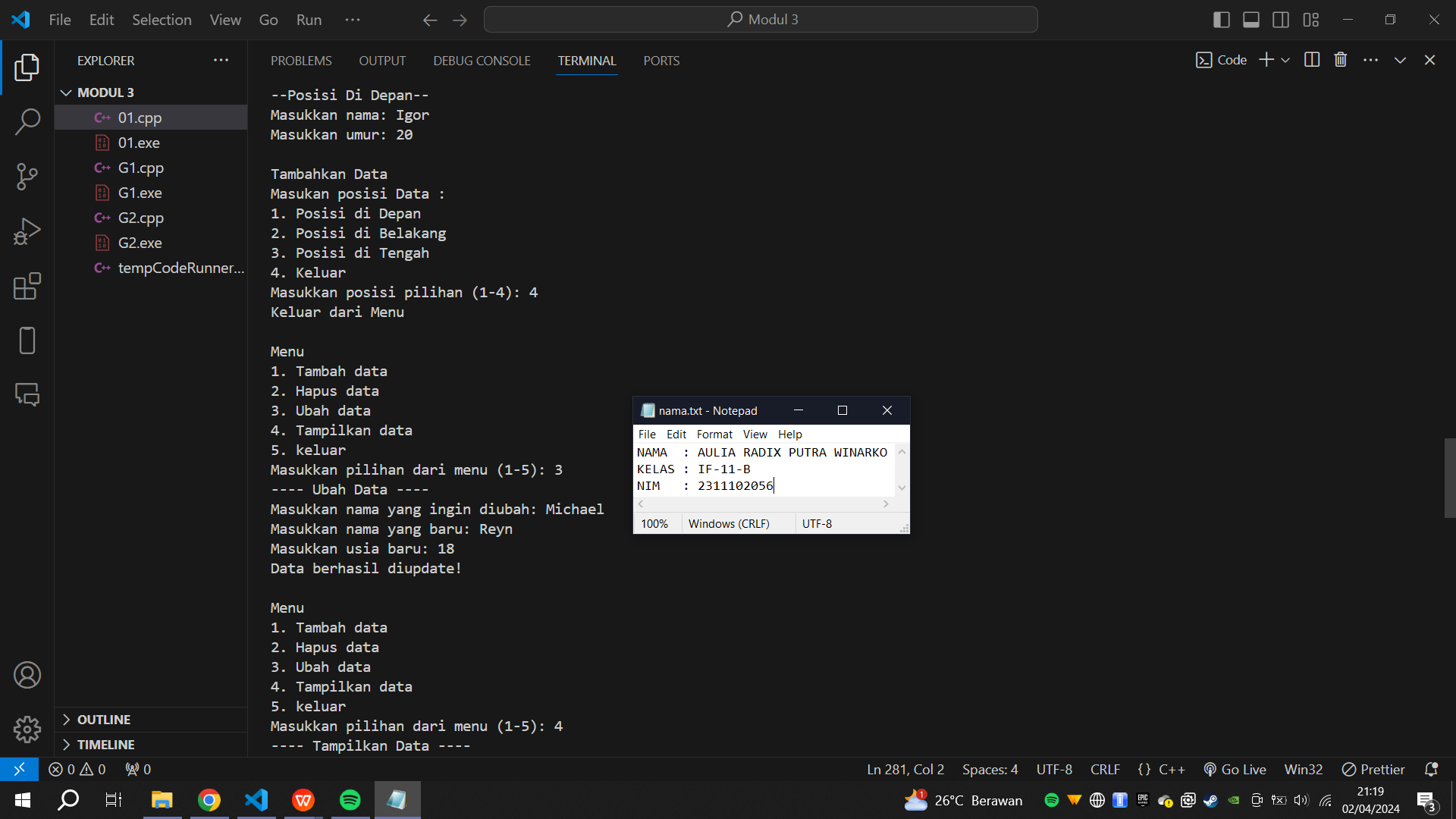
***Soal mengenai Single Linked List***

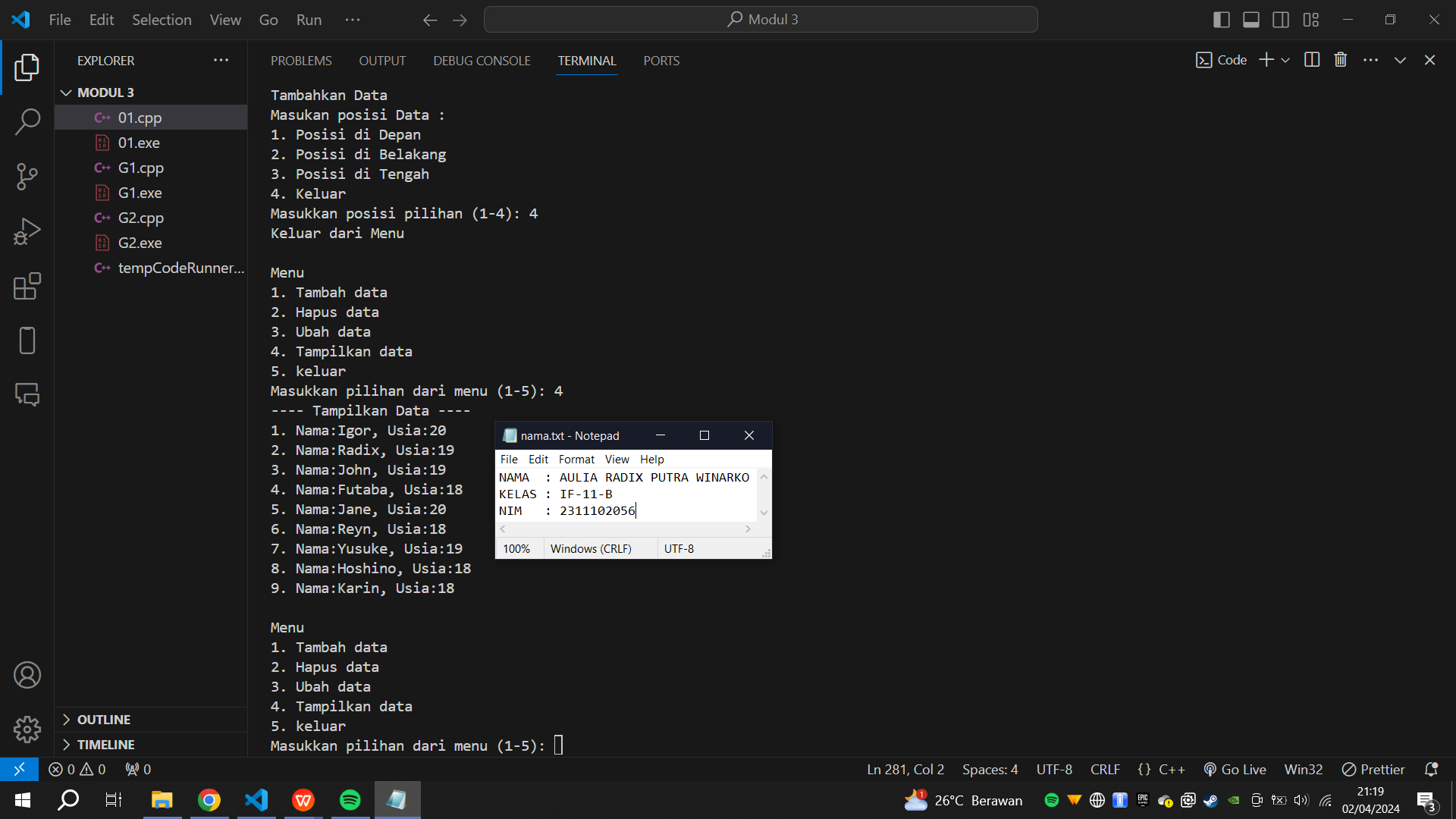
|  |
| --- |
| #include <iostream>  using namespace std;  /// PROGRAM SINGLE LINKED LIST NON-CIRCULAR  // Deklarasi Struct Node  struct Node  {   string nama;   int umur;   Node \*next;  };  Node \*head = NULL;  Node \*tail = NULL;  // Pengecekan  bool isEmpty()  {   if (head == NULL)   return true;   else   return false;  }  // Tambah Depan  void insertDepan(string nama, int umur)  {   // Buat Node baru   Node \*baru = new Node;   baru->nama = nama;   baru->umur = umur;   baru->next = NULL;   if (isEmpty() == true)   {   head = tail = baru;   tail->next = NULL;   }   else   {   baru->next = head;   head = baru;   }  }  // Tambah Belakang  void insertBelakang(string nama, int umur)  {   // Buat Node baru   Node \*baru = new Node;   baru->nama = nama;   baru->umur = umur;   baru->next = NULL;   if (head == NULL)   {   head = baru;   tail = baru;   }   else   {   tail->next = baru;   tail = baru;   }  }  // Hitung Jumlah List  int hitungList()  {   Node \*hitung;   hitung = head;   int jumlah = 0;   while (hitung != NULL)   {   jumlah++;   hitung = hitung->next;   }   return jumlah;  }  // Tambah Tengah  void insertTengah(string nama, int umur, int posisi)  {   if (posisi < 1 || posisi > hitungList())   {   cout << "Posisi diluar jangkauan" << endl;   }   else if (posisi == 1)   {   cout << "Posisi bukan posisi tengah" << endl;   }   else   {   // buat node baru   Node \*baru;   baru = new Node();   baru->nama = nama;   baru->umur = umur;   int i = 1;   Node \*current = head;   while (i < posisi - 1 && current->next != NULL)   {   current = current->next;   i++;   }   baru->next = current->next;   current->next = baru;   }  }  void remove(string nama)  {   if (head == NULL)   {   cout << "List kosong!" << endl;   return;   }   if (head->nama == nama)   {   Node \*temp = head;   head = head->next;   delete temp;   cout << "Data berhasil dihapus!" << endl;   return;   }   Node \*current = head;   while (current->next != NULL && current->next->nama !=  nama)   {   current = current->next;   }   if (current->next == NULL)   {   cout << "Data tidak ditemukan!" << endl;   return;   }   Node \*temp = current->next;   current->next = temp->next;   delete temp;   cout << "Data berhasil dihapus!" << endl;  }  void update(string oldnama, string newnama, int newumur)  {   if (head == NULL)   {   cout << "List kosong!" << endl;   return;   }   Node \*current = head;   while (current != NULL)   {   if (current->nama == oldnama)   {   current->nama = newnama;   current->umur = newumur;   cout << "Data berhasil diupdate!" << endl;   return;   }   current = current->next;   }   cout << "Data tidak ditemukan!" << endl;  }  // Tampilkan List  void tampil()  {   if (head == NULL)   {   cout << "List Kosong!" << endl;   return;   }   Node \*current = head;   int o = 1;   while (current != NULL)   {   cout << o++ << ". "   << "Nama:" << current->nama << ", Usia:" <<  current->umur << endl;   current = current->next;   }  }  int main()  {   int choice;   int choice\_data;   string nama;   int umur;   int posisi;  loop\_menu:   cout << "\nMenu" << endl;   cout << "1. Tambah data" << endl;   cout << "2. Hapus data" << endl;   cout << "3. Ubah data" << endl;   cout << "4. Tampilkan data" << endl;   cout << "5. keluar" << endl;   cout << "Masukkan pilihan dari menu (1-5): ";   cin >> choice;   switch (choice)   {   case 1:   {   loop\_choice\_data:   cout << endl;   cout << "Tambahkan Data\n";   cout << "Masukan posisi Data :\n";   cout << "1. Posisi di Depan\n";   cout << "2. Posisi di Belakang\n";   cout << "3. Posisi di Tengah\n";   cout << "4. Keluar\n";   cout << "Masukkan posisi pilihan (1-4): ";   cin >> choice\_data;   if (choice\_data == 1)   {   cout << "--Posisi Di Depan--\n";   cout << "Masukkan nama: ";   cin >> nama;   cout << "Masukkan umur: ";   cin >> umur;   insertDepan(nama, umur);   }   else if (choice\_data == 2)   {   cout << "--Posisi Di Belakang--\n";   cout << "Masukkan nama: ";   cin >> nama;   cout << "Masukkan umur: ";   cin >> umur;   insertBelakang(nama, umur);   }   else if (choice\_data == 3)   {   cout << "--Posisi Di Tengah--\n";   cout << "Masukkan nama: ";   cin >> nama;   cout << "Masukkan umur: ";   cin >> umur;   cout << "Masukkan posisi: ";   cin >> posisi;   insertTengah(nama, umur, posisi);   }   else if (choice\_data == 4)   {   cout << "Keluar dari Menu\n";   goto loop\_menu;   }   else   {   cout << "Pilihan Anda Tidak Valid\n";   }   goto loop\_choice\_data;   break;   }   case 2:   {   cout << "---- Hapus Data ----\n";   cout << "Masukkan nama yang ingin dihapus :";   cin >> nama;   remove(nama);   break;   }   case 3:   {   string oldnama, newnama;   cout << "---- Ubah Data ----\n";   cout << "Masukkan nama yang ingin diubah: ";   cin >> oldnama;   cout << "Masukkan nama yang baru: ";   cin >> newnama;   cout << "Masukkan usia baru: ";   cin >> umur;   update(oldnama, newnama, umur);   break;   }   case 4:   {   cout << "---- Tampilkan Data ----\n";   tampil();   break;   }   case 5:   {   cout << "Terima Kasih" << endl;   return 0;   }   default:   {   cout << "Pilihan tidak valid" << endl;   }   }   goto loop\_menu;   return 0;  }  } |

Screenshots Output









Deskripsi:

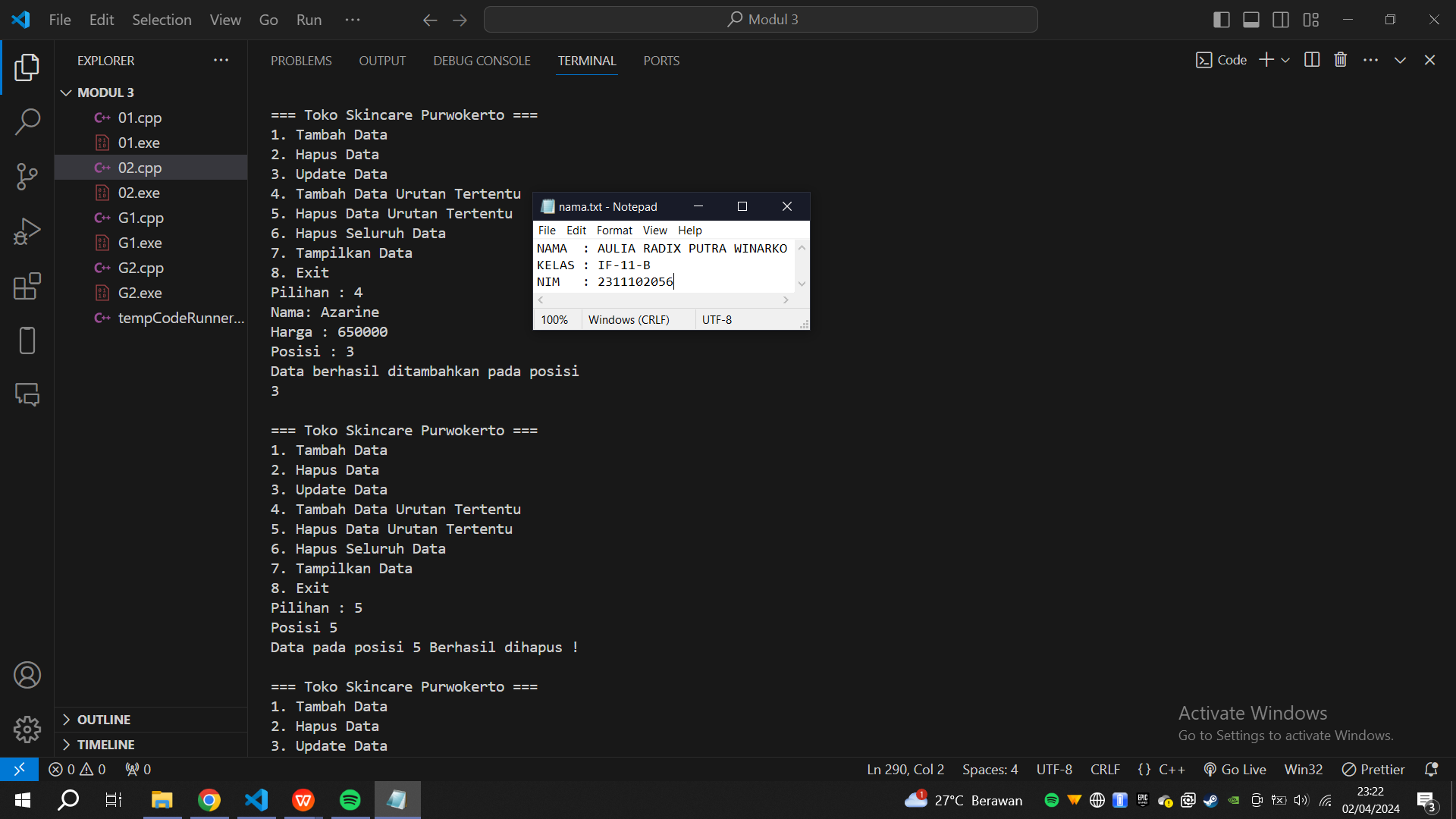
Program diatas merupakan implementasi dari sebuah linked list non-circular dengan menggunakan bahasa pemrograman C++. Linked list adalah struktur data yang terdiri dari urutan elemen-elemen yang terhubung satu sama lain melalui pointer.

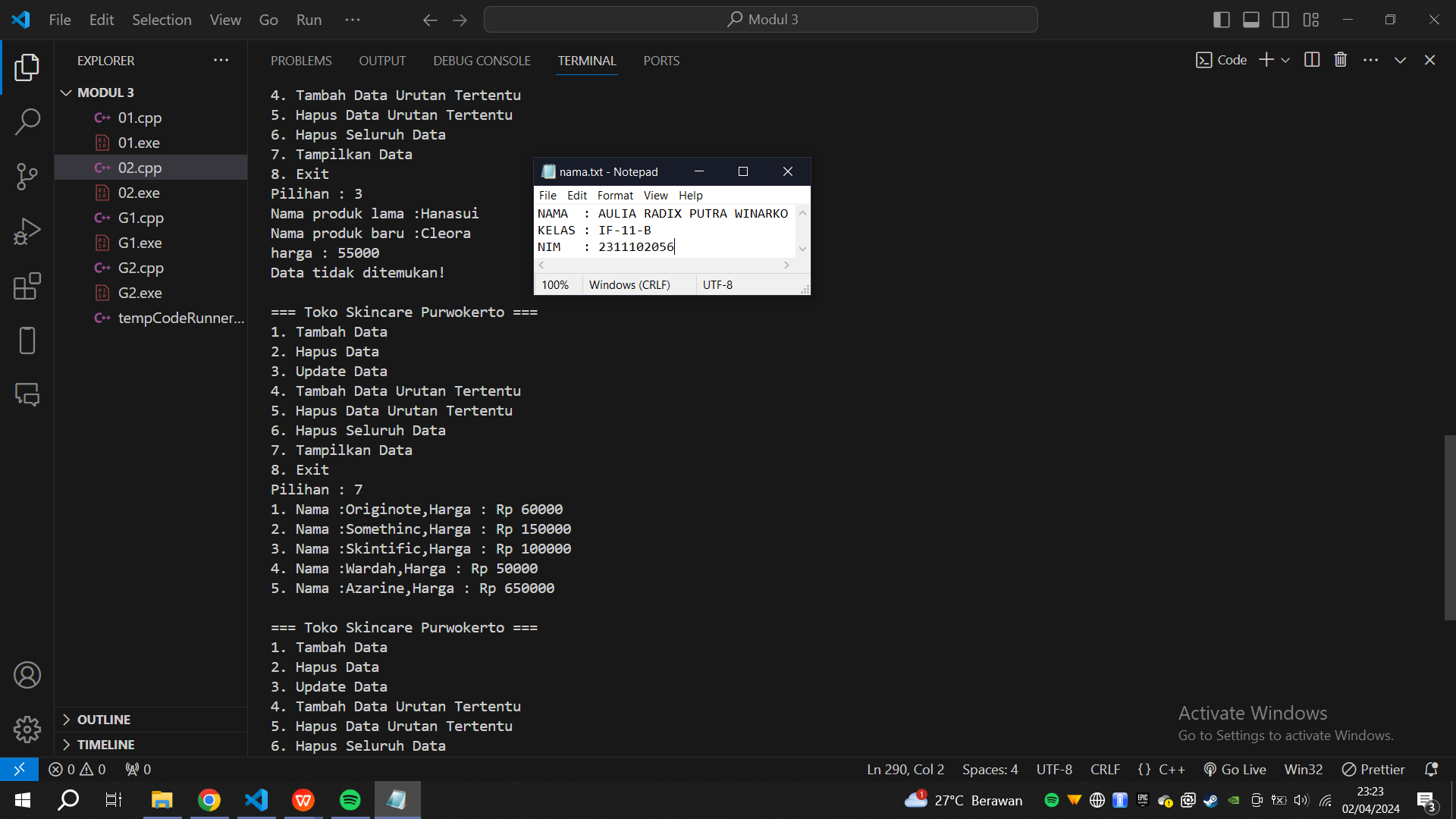
Unguided 2

Soal mengenai Double Linked List

|  |
| --- |
| #include <iostream>  using namespace std;  class skincare  {  public:      string nama, oldnama, newnama;      int harga;      skincare \*prev;      skincare \*next;      skincare(string \_nama = "", int \_harga = 0, skincare \*\_prev = nullptr, skincare \*\_next = nullptr)      {          nama = \_nama;          harga = \_harga;          prev = \_prev;          next = \_next;      }  };  class tokoskincare  {  private:      skincare \*head;      skincare \*tail;  public:      tokoskincare()      {          head = nullptr;          tail = nullptr;      }      void tambahdata(string \_nama, int \_harga)      {          skincare \*newNode = new skincare(\_nama, \_harga);          if (head == nullptr)          {              head = newNode;              tail = newNode;          }          else          {              tail->next = newNode;              newNode->prev = tail;              tail = newNode;          }          cout << "Data berhasil ditambahkan" << endl;      }      void hapusData(string nama)      {          if (head == nullptr)          {              cout << "Tidak ada data yang dapat dihapus!" << endl;              return;          }          skincare \*temp = head;          while (temp != nullptr)          {              if (temp->nama == nama)              {                  if (temp == head)                  {                      head = temp->next;                      if (head != nullptr)                          head->prev = nullptr;                  }                  else if (temp == tail)                  {                      tail = temp->prev;                      if (tail != nullptr)                          tail->next = nullptr;                  }                  else                  {                      temp->prev->next = temp->next;                      temp->next->prev = temp->prev;                  }                  delete temp;                  cout << "Data dengan nama " << nama << "Berhasil dihapus !" << endl;                  return;              }              temp = temp->next;          }          cout << "data dengan nama " << nama << ", tidak ditemukan " << endl;      }      bool updatedata(string \_oldnama, string \_newnama, int \_harga)      {          skincare \*current = head;          while (current != nullptr)          {              if (current->nama == \_oldnama)              {                  current->nama = \_newnama;                  current->harga = \_harga;                  cout << "Data berhasil di update!" << endl;                  return true;              }              current = current->next;          }          cout << "Data tidak ditemukan!" << endl;          return false;      }      void tambahdataurutan(string \_nama, int \_harga, int pos)      {          skincare \*current = head;          skincare \*prevNode = nullptr;          skincare \*newNode = new skincare(\_nama, \_harga);          for (int i = 1; pos && current != nullptr; i++)          {              prevNode = current;              current = current->next;          }          if (prevNode == nullptr)          {              head = newNode;          }          else          {              prevNode->next = newNode;              prevNode->prev = prevNode;          }          newNode->next = current;          if (current != nullptr)          {              current->prev = newNode;          }          else          {              tail = newNode;          }          cout << "Data berhasil ditambahkan pada posisi " << pos << endl;      }      void hapusdataurutan(int pos)      {          skincare \*current = head;          skincare \*prevNode = nullptr;          for (int i = 1; i < pos && current != nullptr; i++)          {              prevNode = current;              current = current->next;          }          if (current == nullptr)          {              cout << "Posisi" << pos << "tidak valid" << endl;              return;          }          if (prevNode == nullptr)          {              head = current->next;          }          else          {              prevNode->next = current->next;          }          if (current->next != nullptr)          {              current->next->prev = prevNode;          }          else          {              tail = prevNode;          }          delete current;          cout << "Data pada posisi " << pos << " Berhasil dihapus !" << endl;      }      void hapusseluruhdata()      {          skincare \*current = head;          skincare \*nextNode = nullptr;          while (current != nullptr)          {              nextNode = current->next;              delete current;              current = nextNode;          }          head = nullptr;          tail = nullptr;          cout << "seluruh data berhasil dihapus!" << endl;      }      void tampilkandata()      {          if (head == nullptr)          {              cout << "tidak ada data yang dapat ditampilkan" << endl;              return;          }          skincare \*current = head;          int i = 1;          while (current != nullptr)          {              cout << i << ". Nama :" << current->nama << ", Harga : Rp " << current->harga << endl;              current = current->next;              i++;          }      }  };  int main()  {      tokoskincare toko;      int pilihan, harga, pos;      string nama;      do      {          cout << "\n=== Toko Skincare Purwokerto ===" << endl;          cout << "1. Tambah Data" << endl;          cout << "2. Hapus Data" << endl;          cout << "3. Update Data" << endl;          cout << "4. Tambah Data Urutan Tertentu" << endl;          cout << "5. Hapus Data Urutan Tertentu" << endl;          cout << "6. Hapus Seluruh Data" << endl;          cout << "7. Tampilkan Data" << endl;          cout << "8. Exit" << endl;          cout << "Pilihan : ";          cin >> pilihan;          switch (pilihan)          {          case 1:          {              int jumlah;              cout << "Masukan jumlah yang diinputkan : ";              cin >> jumlah;              for (int q = 1; q <= jumlah; q++)              {                  cout << "Nama : ";                  cin.ignore();                  getline(cin, nama);                  cout << "harga : ";                  cin >> harga;                  toko.tambahdata(nama, harga);              }              break;          }          case 2:          {              toko.hapusData(nama);              break;          }          case 3:          {              string oldnama, newnama;              cout << "Nama produk lama :";              cin.ignore();              getline(cin, oldnama);              cout << "Nama produk baru :";              getline(cin, newnama);              cout << "harga : ";              cin >> harga;              toko.updatedata(oldnama, newnama, harga);              break;          }          case 4:          {              cout << "Nama: ";              cin.ignore();              getline(cin, nama);              cout << "Harga : ";              cin >> harga;              cout << "Posisi : ";              cin >> pos;              toko.tambahdataurutan(nama, harga, pos);              break;          }          case 5:          {              cout << "Posisi ";              cin >> pos;              toko.hapusdataurutan(pos);              break;          }          case 6:          {              toko.hapusseluruhdata();              break;          }          case 7:          {              toko.tampilkandata();              break;          }          case 8:          {              cout << "Terima kasih!" << endl;              break;          }          default:          {              cout << "Pilihan tidak valid" << endl;              break;          }          }      } while (pilihan != 8);      return 0;  } |

Screenshots Output





Deskripsi:

Kode C++ di atas merupakan implementasi sederhana dari sebuah program yang memanfaatkan konsep linked list untuk menyimpan data produk skincare dalam sebuah toko. Program ini memungkinkan user untuk menambah, menghapus, mengupdate, menambahkan data pada posisi tertentu, menghapus data pada posisi tertentu, menghapus seluruh data, dan menampilkan data produk skincare dalam toko.

1. Referensi[Bob

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.

TylerMSFT. (n.d.). Collections (C++/CX). diakses dari https://learn.microsoft.com/en-us/cpp/cppcx/collections-c-cx?view=msvc-170ot Penilaian sebesar 15%