**LAPORAN PRATIKUM STRUKTUR DATA**

**JOBSHEET 6**



**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**

**FAKULITAS TEKNIK**

**UNIVERSITAS NEGERI PADANG**

**2023**

**1.**

1. **Source Code**

//created by 23343079\_Najwa Eldiara Owilia Tiksa

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Definisi struktur untuk data mahasiswa

typedef struct {

char nim[20];

char nama[50];

float nilai;

} Mahasiswa;

// Definisi struktur untuk node stack

typedef struct Node {

Mahasiswa data;

struct Node\* next;

} Node;

// Definisi struktur untuk stack

typedef struct {

Node\* top;

} Stack;

// Fungsi untuk membuat node baru

Node\* createNode(Mahasiswa data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Fungsi untuk membuat stack kosong

void initializeStack(Stack\* stack) {

stack->top = NULL;

}

// Fungsi untuk mengecek apakah stack kosong

int isEmpty(Stack\* stack) {

return stack->top == NULL;

}

// Fungsi untuk menambahkan data mahasiswa ke dalam stack

void push(Stack\* stack, Mahasiswa data) {

Node\* newNode = createNode(data);

newNode->next = stack->top;

stack->top = newNode;

}

// Fungsi untuk menghapus data mahasiswa dari stack

Mahasiswa pop(Stack\* stack) {

if (isEmpty(stack)) {

Mahasiswa emptyMahasiswa = {"", "", 0};

return emptyMahasiswa;

}

Node\* temp = stack->top;

stack->top = stack->top->next;

Mahasiswa poppedData = temp->data;

free(temp);

return poppedData;

}

// Fungsi untuk menampilkan data mahasiswa dalam stack

void display(Stack\* stack) {

Node\* current = stack->top;

printf("\nData Mahasiswa:\n");

while (current != NULL) {

printf("NIM: %s\n", current->data.nim);

printf("Nama: %s\n", current->data.nama);

printf("Nilai: %.2f\n", current->data.nilai);

printf("-----------------\n");

current = current->next;

}

}

// Fungsi untuk mengurutkan data mahasiswa dalam stack berdasarkan NIM

void sort(Stack\* stack) {

// Metode pengurutan stack tidak diperlukan, karena akses langsung ke elemen tidak diperbolehkan dalam stack.

// Jadi, kita harus menyalin data stack ke dalam array, mengurutkan array, dan kemudian menyalin kembali ke stack.

Mahasiswa arr[100];

int count = 0;

// Memasukkan data dari stack ke dalam array

Node\* current = stack->top;

while (current != NULL) {

arr[count++] = current->data;

current = current->next;

}

// Pengurutan array berdasarkan NIM

for (int i = 0; i < count - 1; i++) {

for (int j = 0; j < count - i - 1; j++) {

if (strcmp(arr[j].nim, arr[j + 1].nim) > 0) {

Mahasiswa temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

// Memasukkan data dari array ke dalam stack

initializeStack(stack);

for (int i = count - 1; i >= 0; i--) {

push(stack, arr[i]);

}

printf("Data Mahasiswa berhasil diurutkan berdasarkan NIM.\n");

}

// Fungsi untuk membersihkan stack dari semua node

void clearStack(Stack\* stack) {

while (!isEmpty(stack)) {

pop(stack);

}

}

int main() {

Stack stack;

initializeStack(&stack);

int choice;

Mahasiswa newMahasiswa;

while (1) {

printf("\n=================== Menu ===================\n");

printf("|1. Tambah Data Mahasiswa |\n");

printf("|2. Hapus Data Mahasiswa |\n");

printf("|3. Tampilkan Data Mahasiswa |\n");

printf("|4. Urutkan Data Mahasiswa berdasarkan NIM |\n");

printf("|5. Keluar |\n");

printf("============================================\n");

printf("\nMasukkan pilihan : ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("\nMasukkan NIM : ");

scanf("%s", newMahasiswa.nim);

printf("Masukkan Nama : ");

scanf(" %[^\n]", newMahasiswa.nama);

printf("Masukkan Nilai : ");

scanf("%f", &newMahasiswa.nilai);

push(&stack, newMahasiswa);

printf("Data mahasiswa berhasil ditambahkan!\n");

break;

case 2:

if (!isEmpty(&stack)) {

pop(&stack);

printf("Data mahasiswa berhasil dihapus!\n");

} else {

printf("Stack kosong, tidak ada data yang dihapus.\n");

}

break;

case 3:

if (!isEmpty(&stack)) {

display(&stack);

} else {

printf("Stack kosong\n");

}

break;

case 4:

if (!isEmpty(&stack)) {

sort(&stack);

} else {

printf("Stack kosong\n");

}

break;

case 5:

clearStack(&stack);

printf("Terima kasih! Program selesai.\n");

exit(0);

default:

printf("Pilihan tidak valid. Silakan pilih menu yang sesuai.\n");

}

}

return 0;

}

1. **Penjelasan Program**

Program di atas adalah sebuah implementasi dari struktur data stack untuk menyimpan dan mengelola data mahasiswa. Program ini menggunakan bahasa C dan mendefinisikan beberapa fungsi untuk melakukan operasi dasar pada stack, seperti menambah (push), menghapus (pop), menampilkan, dan mengurutkan data berdasarkan NIM mahasiswa.

Algoritma yang Digunakan:

1. Push dan Pop: Implementasi stack menggunakan operasi push dan pop. Push menambahkan elemen ke top stack dan pop menghapus elemen dari top stack.

2. Sort: Sorting data dalam stack menggunakan metode Bubble Sort. Data dari stack dipindahkan ke array, diurutkan, lalu dipindahkan kembali ke stack.

Flow Program:

1. Inisialisasi stack kosong.

2. Menampilkan menu:

- Tambah Data Mahasiswa: Menambahkan data mahasiswa ke stack.

- Hapus Data Mahasiswa: Menghapus data mahasiswa dari stack.

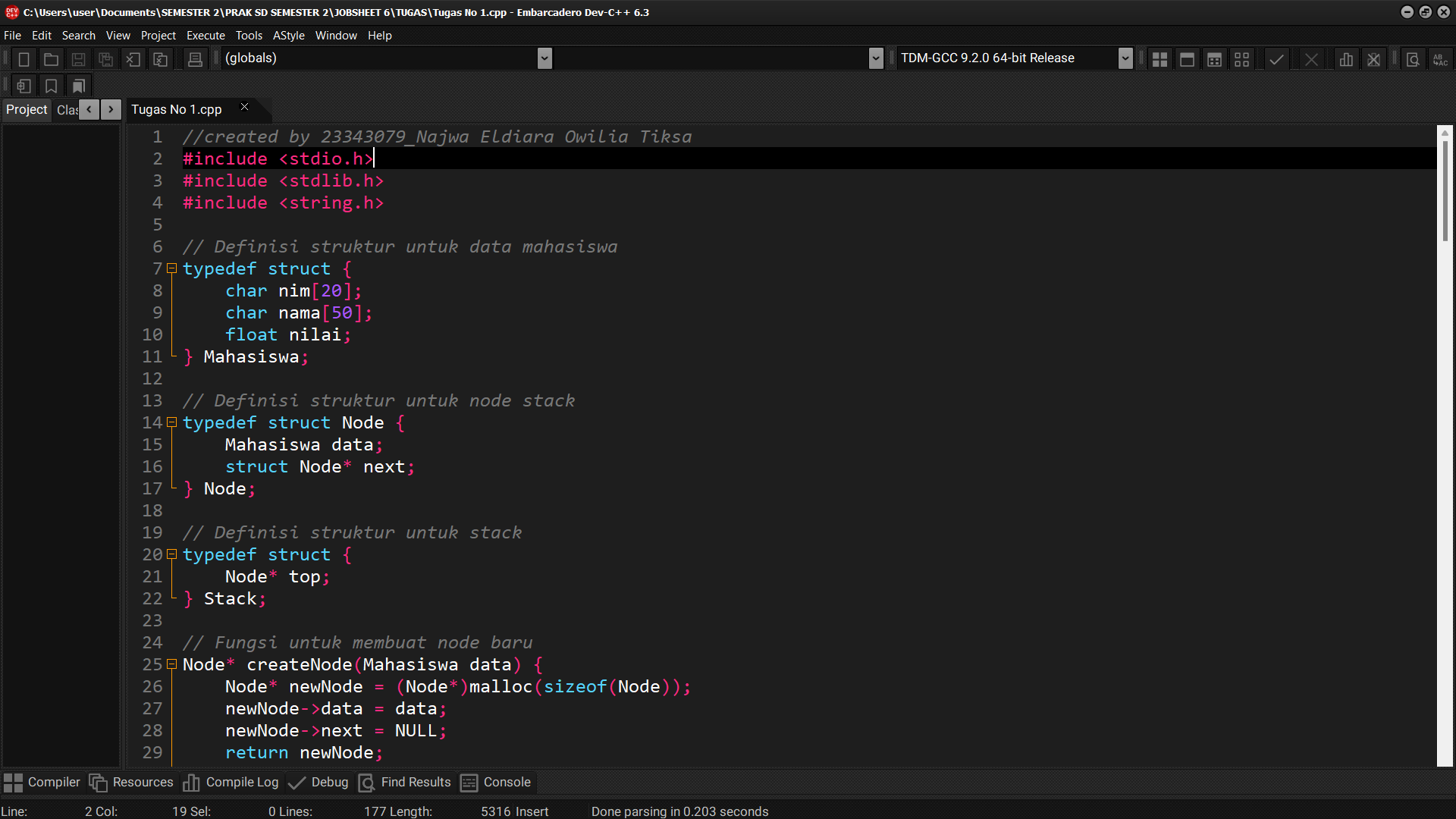
- Tampilkan Data Mahasiswa: Menampilkan semua data mahasiswa dalam stack.

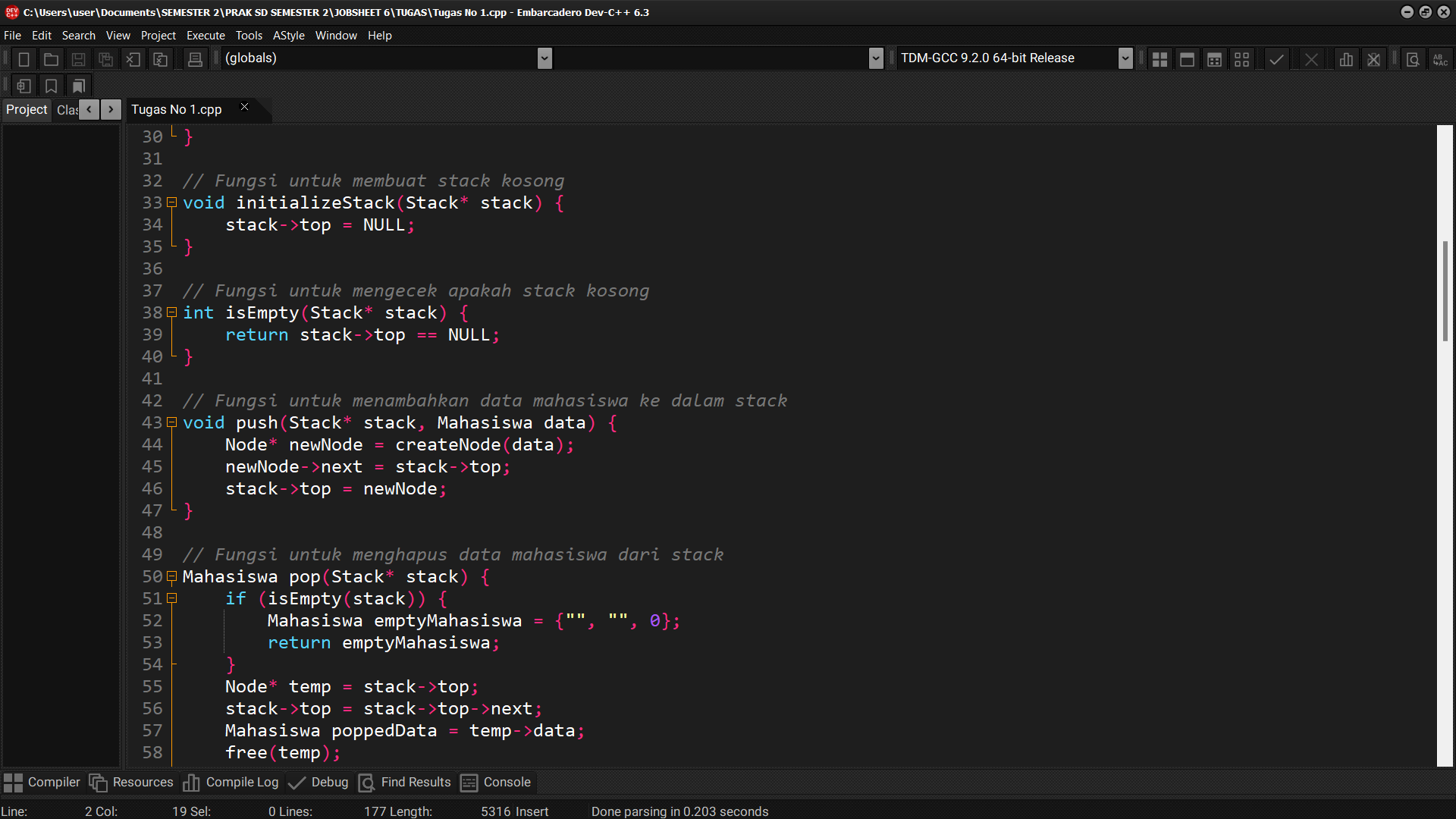
- Urutkan Data Mahasiswa berdasarkan NIM: Mengurutkan data mahasiswa dalam stack berdasarkan NIM.

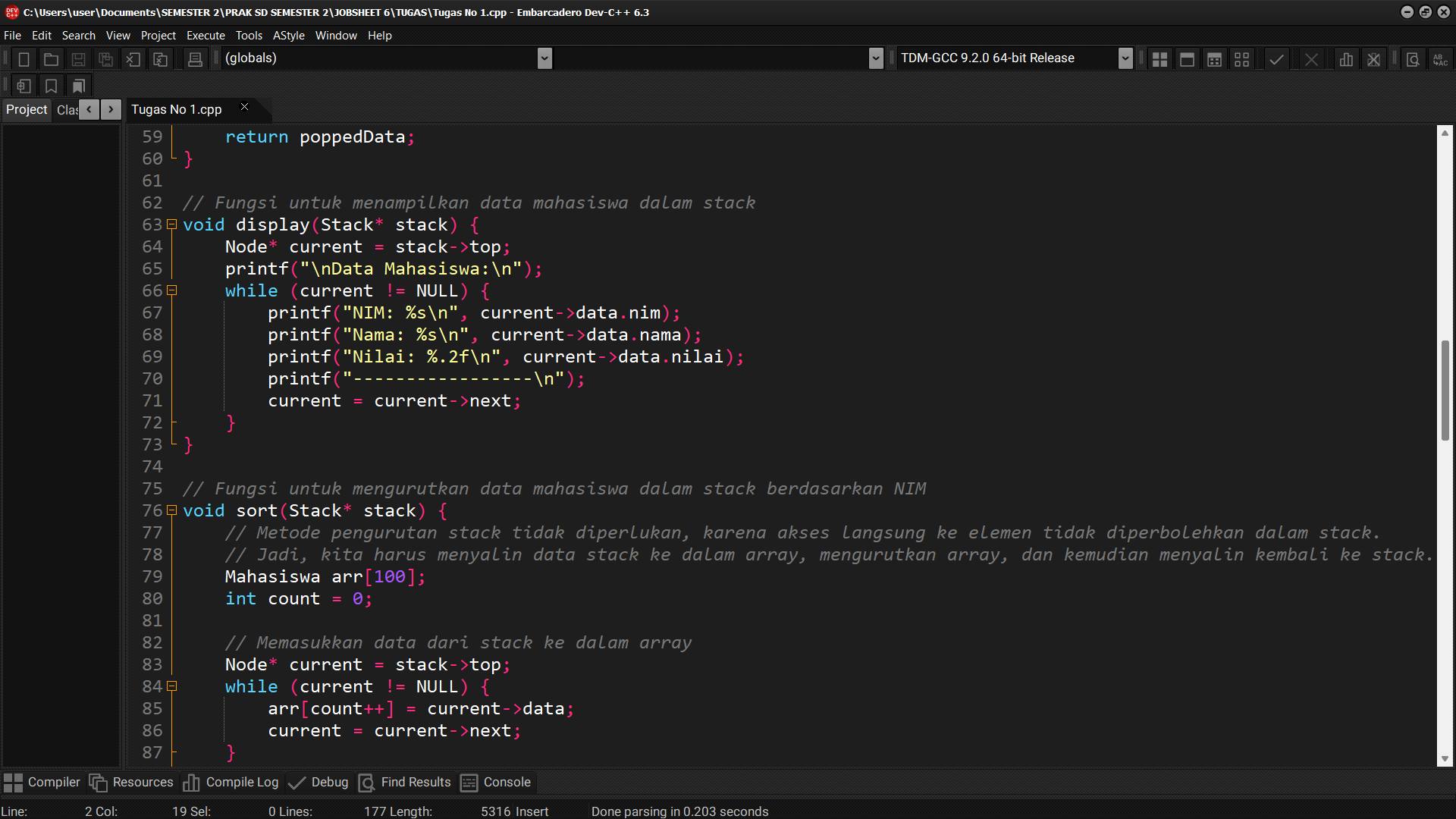
- Keluar: Menghapus semua node dalam stack dan keluar dari program.

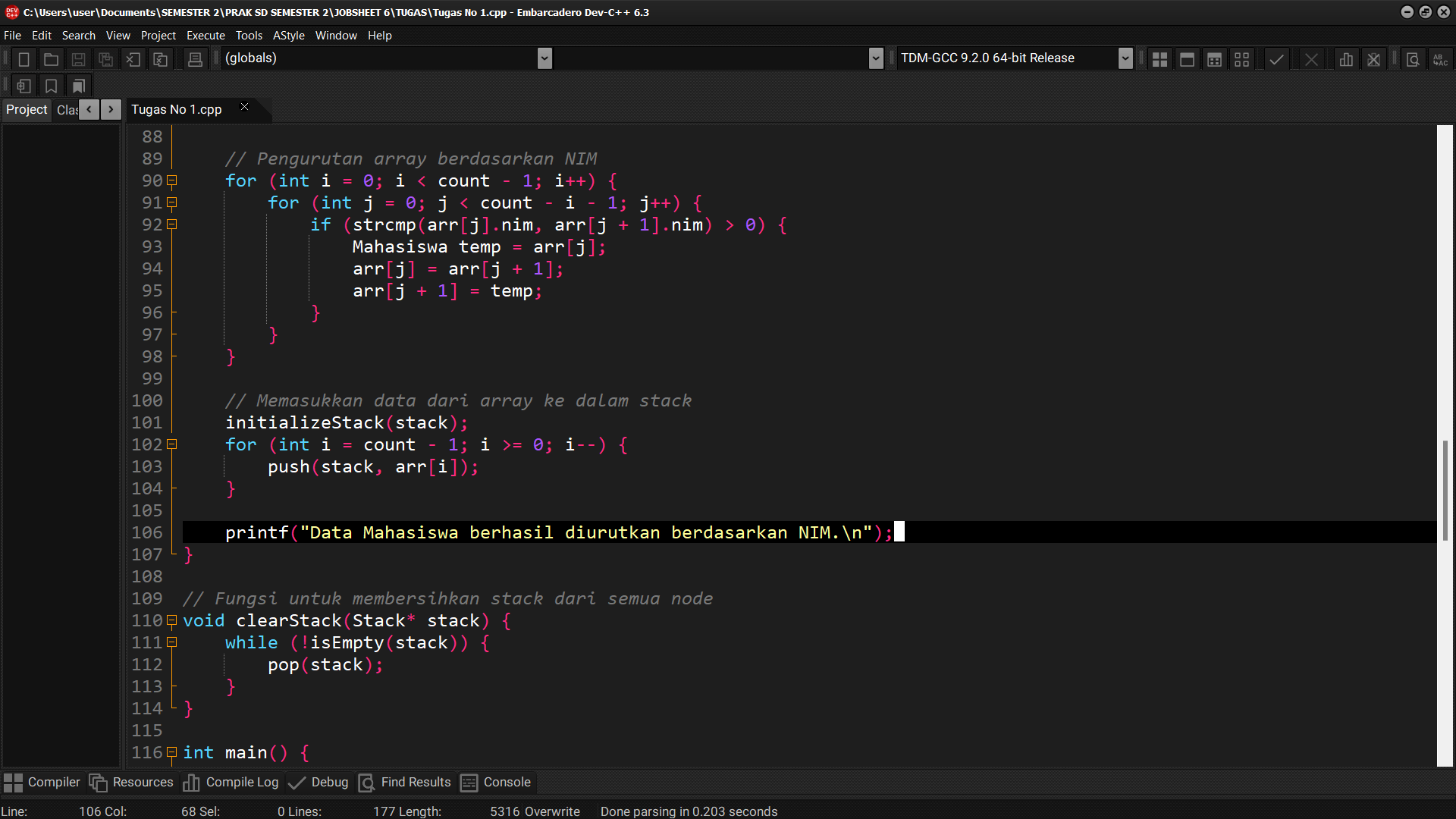
Program ini dirancang untuk latihan dasar dalam memahami struktur data stack, implementasi stack menggunakan linked list, dan operasi dasar seperti push, pop, serta menampilkan dan mengurutkan data.

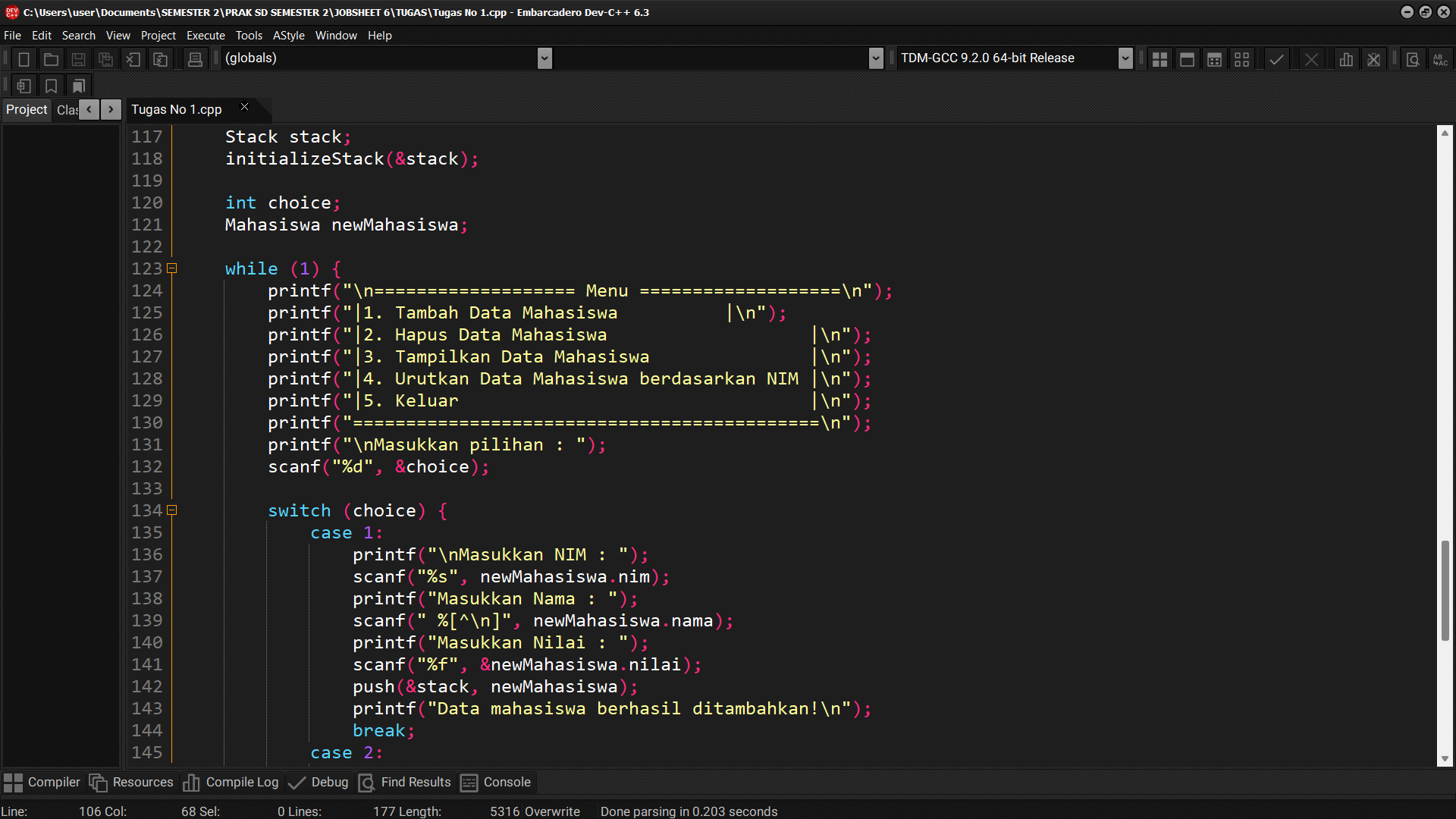
1. **Screenshot program**

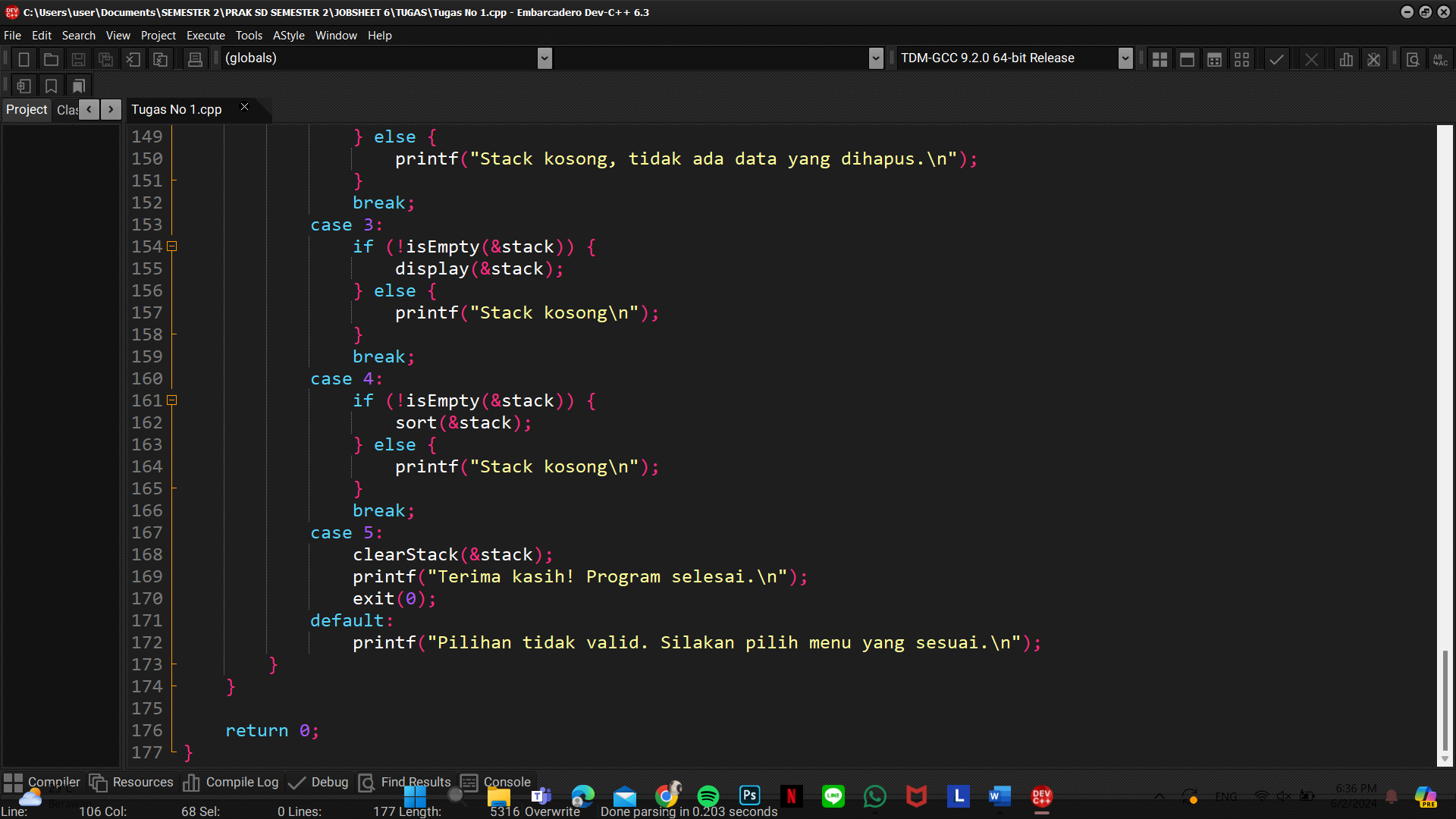












1. **Output**



**2.**

1. **Source Code**

//created by 23343079\_Najwa Eldiara Owilia Tiksa

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LENGTH 100

// Definisi struktur untuk node stack

typedef struct Node {

char data;

struct Node\* next;

} Node;

// Definisi struktur untuk stack

typedef struct {

Node\* top;

} Stack;

// Fungsi untuk membuat node baru

Node\* createNode(char data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Fungsi untuk membuat stack kosong

void initializeStack(Stack\* stack) {

stack->top = NULL;

}

// Fungsi untuk mengecek apakah stack kosong

int isEmpty(Stack\* stack) {

return stack->top == NULL;

}

// Fungsi untuk menambahkan karakter ke dalam stack

void push(Stack\* stack, char data) {

Node\* newNode = createNode(data);

newNode->next = stack->top;

stack->top = newNode;

}

// Fungsi untuk menghapus karakter dari stack

char pop(Stack\* stack) {

if (isEmpty(stack)) {

return '\0'; // Karakter null jika stack kosong

}

Node\* temp = stack->top;

stack->top = stack->top->next;

char poppedData = temp->data;

free(temp);

return poppedData;

}

// Fungsi untuk membalikkan kata menggunakan stack

void reverseWord(char\* word) {

Stack stack;

initializeStack(&stack);

// Push semua karakter dari kata ke dalam stack

int i;

for (i = 0; word[i] != '\0'; i++) {

push(&stack, word[i]);

}

// Pop dan cetak karakter dari stack untuk membalikkan kata

printf("Kata dibalik: ");

while (!isEmpty(&stack)) {

printf("%c", pop(&stack));

}

printf("\n");

}

int main() {

char kata[MAX\_LENGTH];

int choice;

while (1) {

printf("\nMenu:\n");

printf("1. Input Kata\n");

printf("2. Balik Kata\n");

printf("3. Exit\n");

printf("Masukkan pilihan: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Masukkan kata: ");

scanf("%s", kata);

break;

case 2:

if (strlen(kata) > 0) {

reverseWord(kata);

} else {

printf("Kata belum diinputkan.\n");

}

break;

case 3:

printf("Terima kasih! Program selesai.\n");

exit(0);

default:

printf("Pilihan tidak valid. Silakan pilih menu yang sesuai.\n");

}

}

return 0;

}

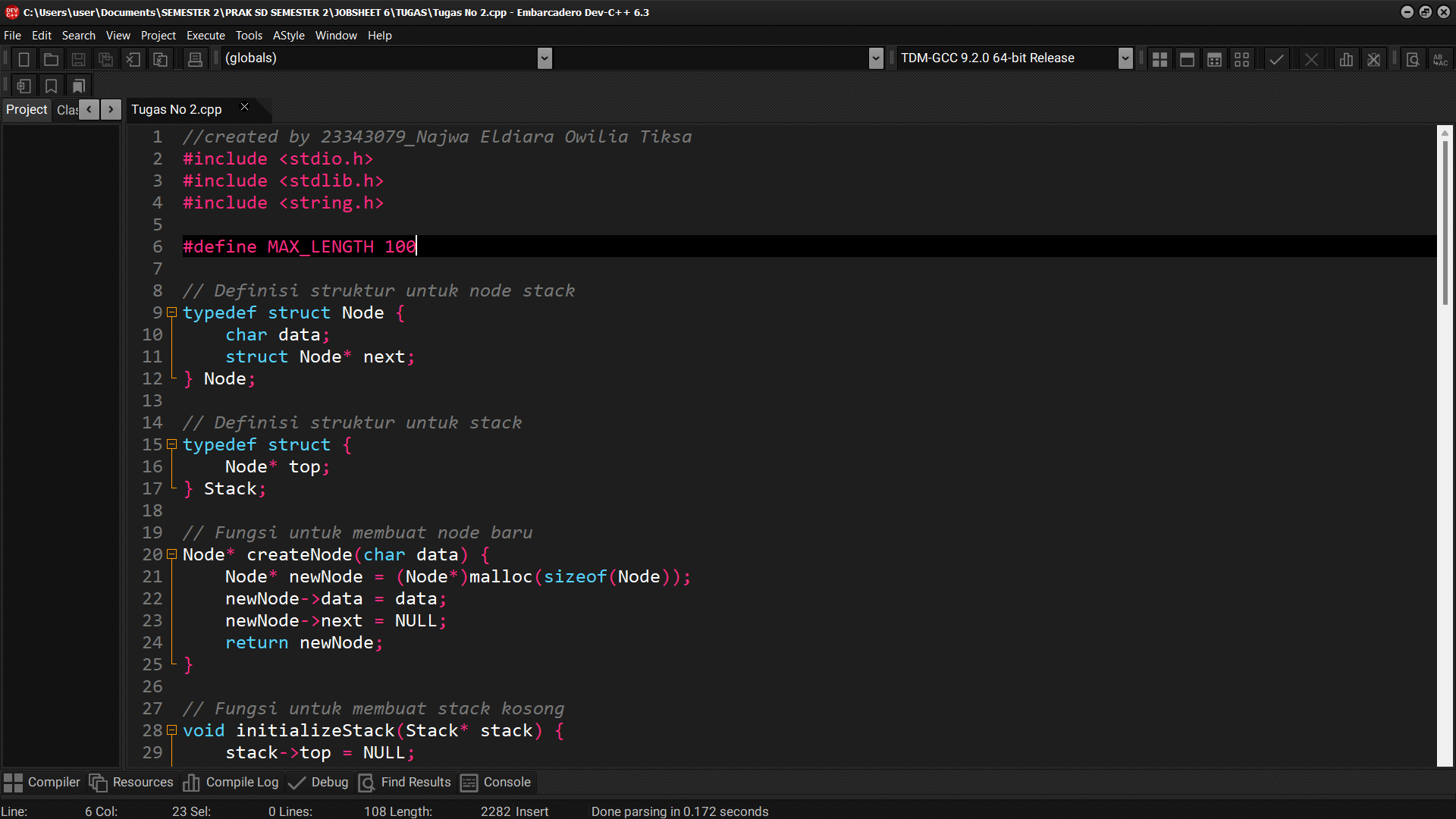
1. **Penjelasan Program**

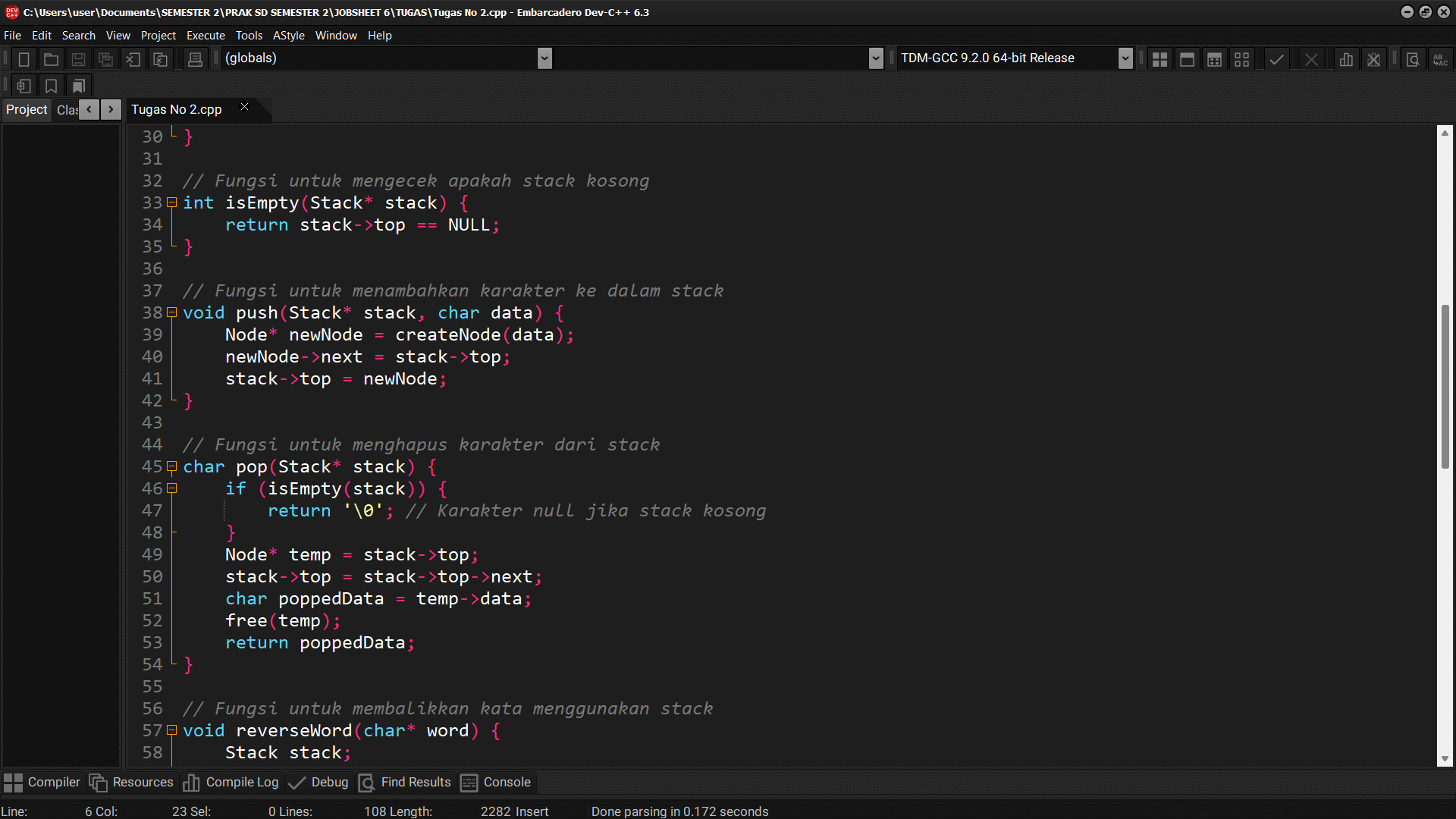
Program di atas adalah sebuah program C yang digunakan untuk membalikkan kata yang dimasukkan oleh pengguna. Program ini menggunakan struktur data stack untuk melakukan operasi pembalikan kata.

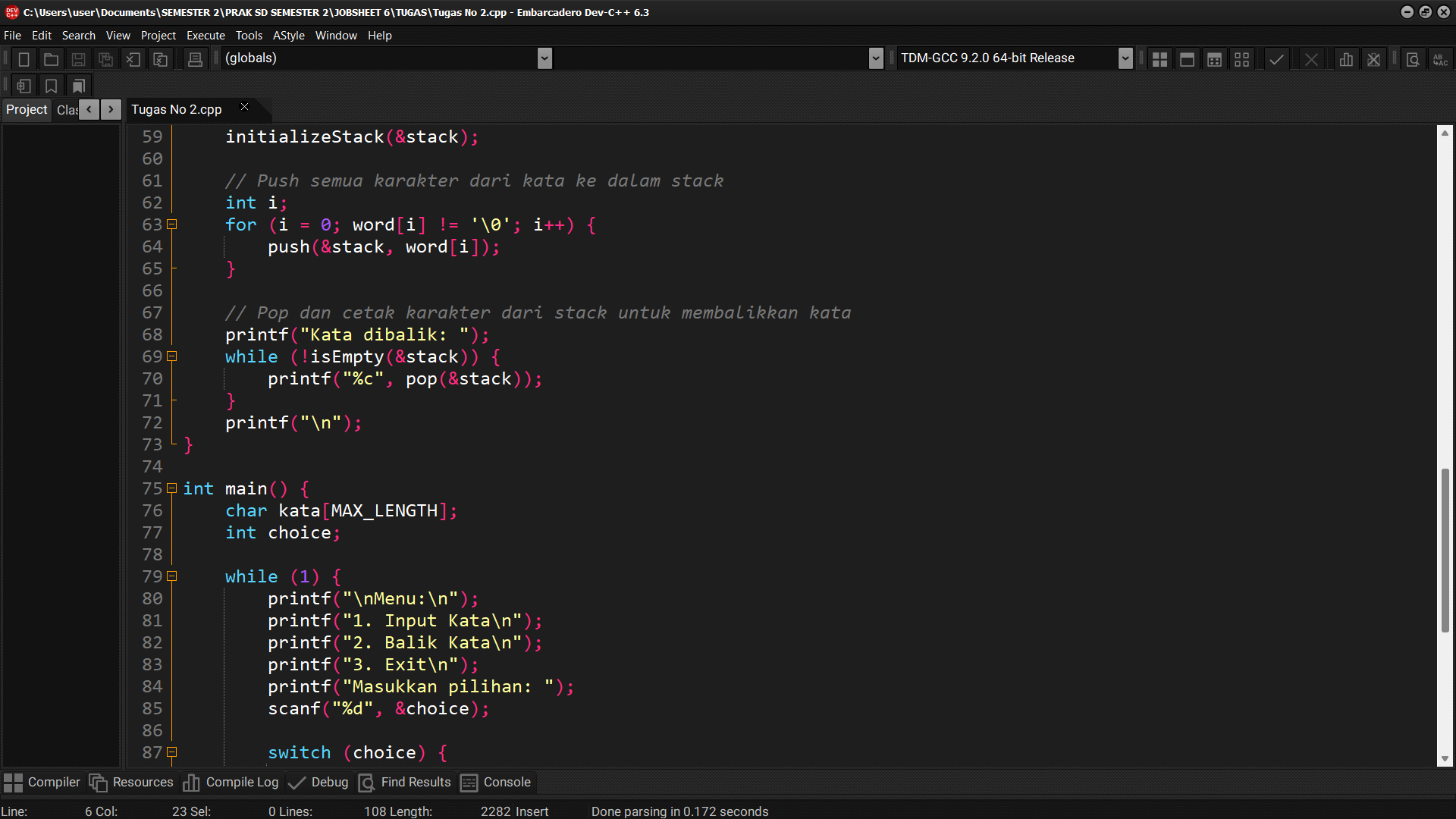
Program ini menggunakan algoritma stack untuk membalikkan kata. Berikut adalah langkah-langkah algoritmanya:

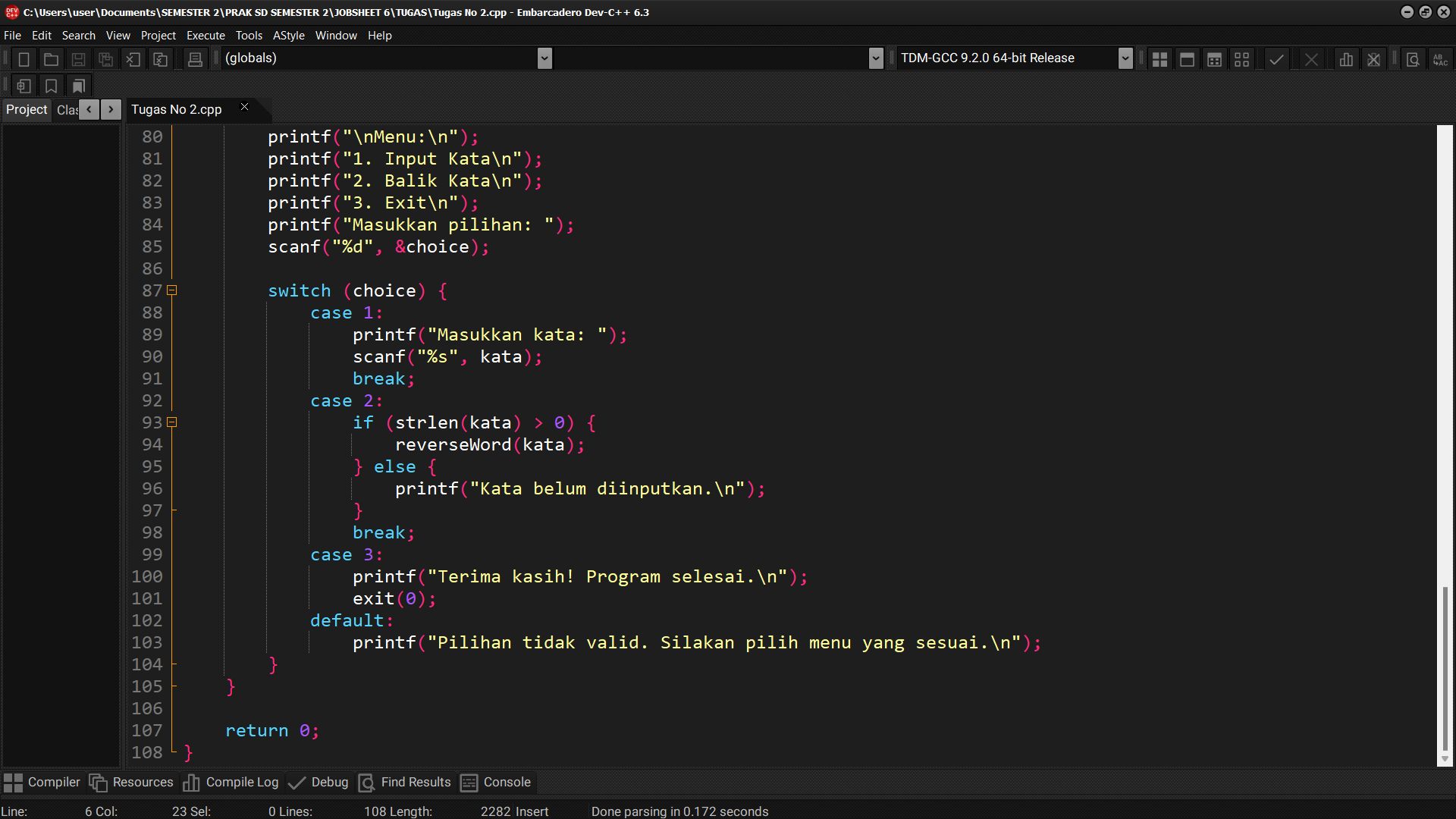
1. Push : Setiap karakter dalam kata didorong ke dalam stack satu per satu. Dalam stack, karakter akan ditumpuk sehingga karakter terakhir dari kata berada di paling atas stack.

2. Pop: Setelah semua karakter didorong ke dalam stack, karakter kemudian di-pop satu per satu dari stack. Karena sifat LIFO (Last In First Out) dari stack, karakter yang terakhir masuk akan keluar pertama, menghasilkan kata yang dibalik.

1. **Screenshot program**







1. **Output**

