**LAPORAN AWAL STRUKTUR DATA**

**Pertemuan ke-4**

**Single Stack**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan – Banten

# LANDASAN TEORI

Dalam dunia pemrograman, stack (tumpukan) merupakan salah satu struktur data linear yang penting dalam pengelolaan data secara berurutan. Stack menggunakan prinsip LIFO (Last In, First Out), yang berarti elemen yang terakhir kali masuk akan menjadi elemen pertama yang keluar.

Konsep ini mirip dengan kehidupan nyata, seperti tumpukan piring di dapur: piring yang terakhir ditaruh di atas akan menjadi piring pertama yang diambil. Stack banyak digunakan pada sistem komputer yang membutuhkan pengelolaan giliran atau reversibilitas, antara lain:

* Undo-redo pada aplikasi editor (misal text editor, spreadsheet)
* Evaluasi ekspresi matematika (prefix, postfix, infix)
* Penelusuran rekursi
* Backtracking pada algoritma

## 1. Pengertian Single Stack

Single Stack adalah struktur data linear di mana elemen hanya bisa dimasukkan dan dihapus dari satu ujung, disebut top. Semua operasi dilakukan di ujung atas stack.

## 2. Ciri-ciri Single Stack

* Memiliki satu pointer utama: top
* Operasi push menambahkan elemen di atas stack
* Operasi pop menghapus elemen dari atas stack
* Stack kosong jika top = -1
* Stack penuh jika top mencapai kapasitas maksimal

## 3. Operasi Dasar Single Stack

* Push → menambahkan elemen ke atas stack
* Pop → menghapus elemen dari atas stack
* Peek/Top → melihat elemen teratas tanpa menghapus
* isEmpty() → mengecek apakah stack kosong
* isFull() → mengecek apakah stack penuh (untuk array)

## 4. Implementasi dengan Array

* Stack menggunakan array bertipe data tertentu
* Pointer top menandai elemen teratas
* Kekurangan: setelah mencapai kapasitas, stack penuh dan memerlukan reset atau perluasan

## 5. Implementasi dengan Linked List

* Node menyimpan data dan pointer ke node berikutnya
* top menunjuk ke node paling atas
* Ukuran stack menjadi dinamis, menyesuaikan jumlah elemen

## 6. Kelebihan Single Stack

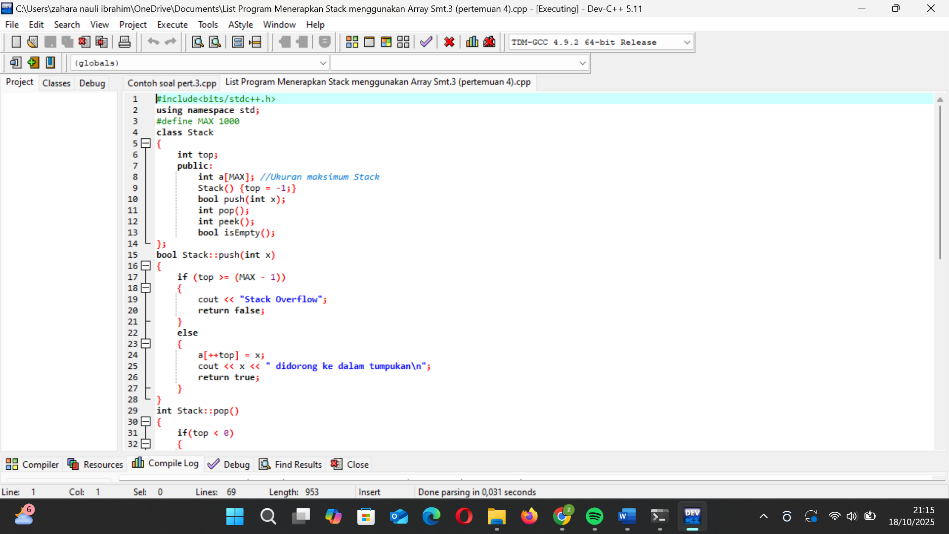
* Sederhana dan mudah diimplementasikan
* Cocok untuk proses sementara dan backtracking

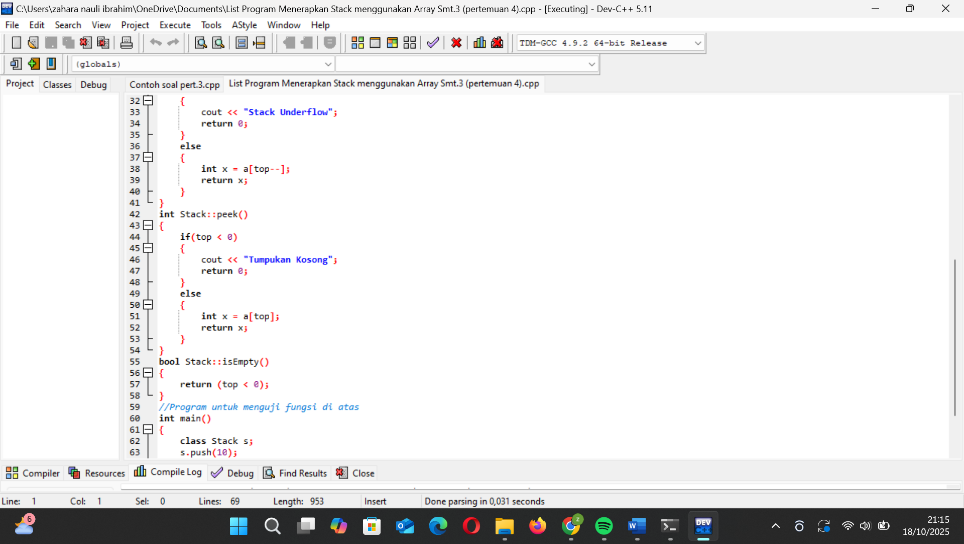
## 7. Kekurangan Single Stack

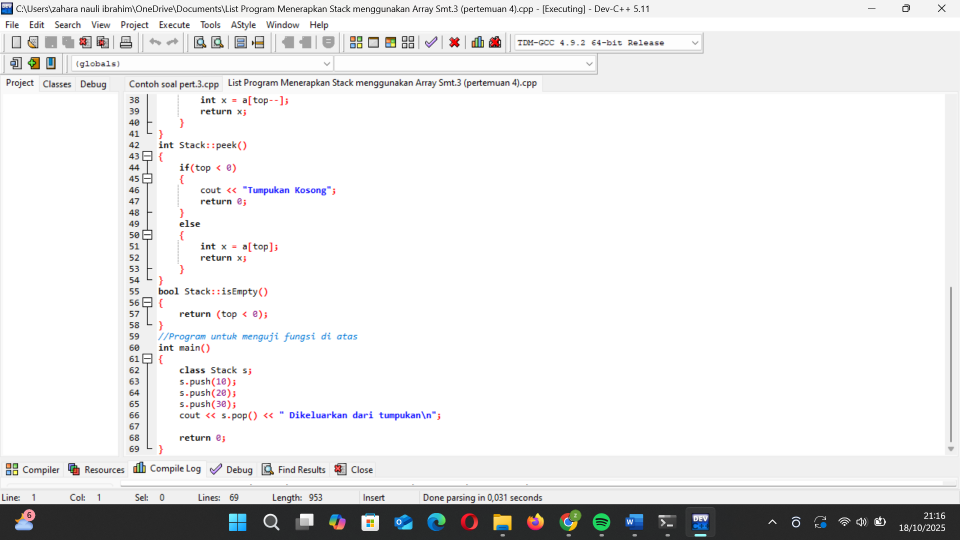
* Array statis memiliki keterbatasan kapasitas
* Linked list memerlukan memori tambahan untuk pointer

# LAPORAN PRAKTIKUM

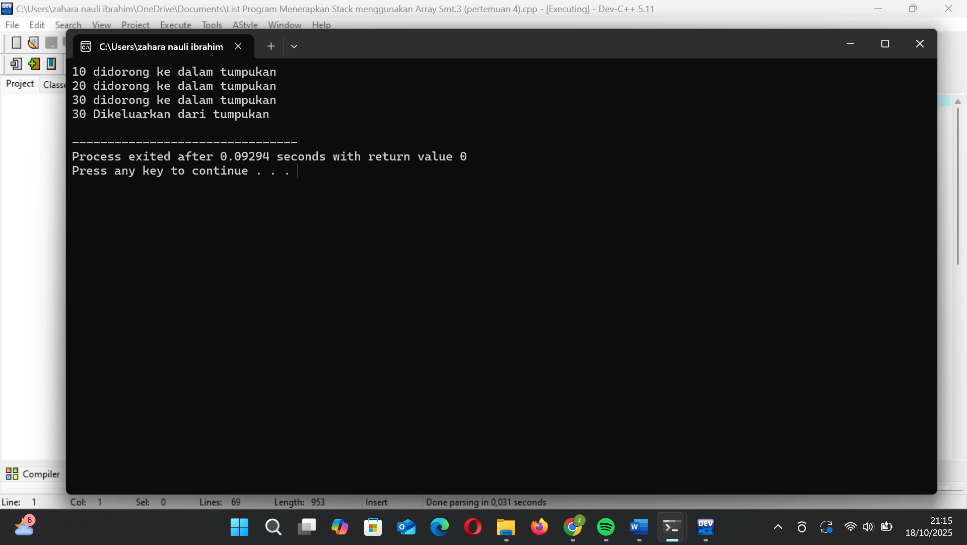
### 1) Program untuk mengimplementasikan operasi dasar pada Stack





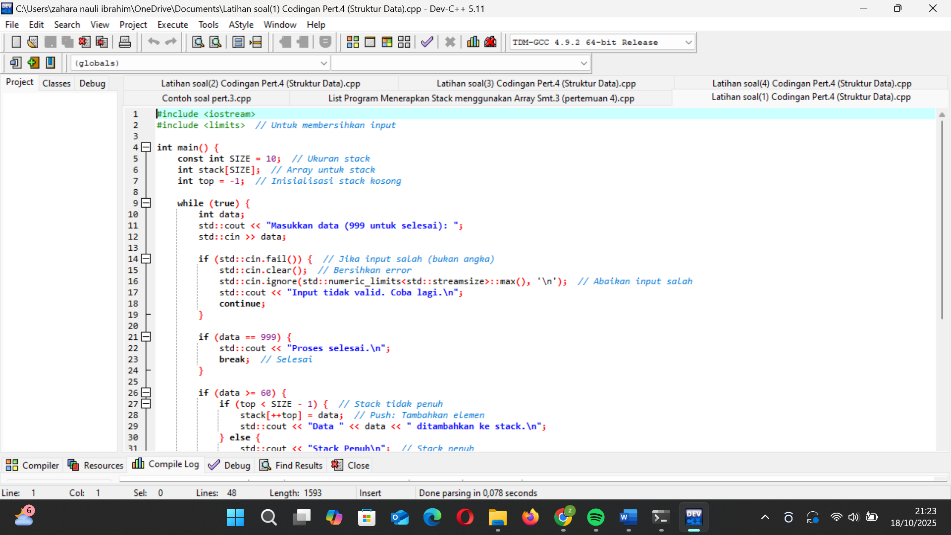


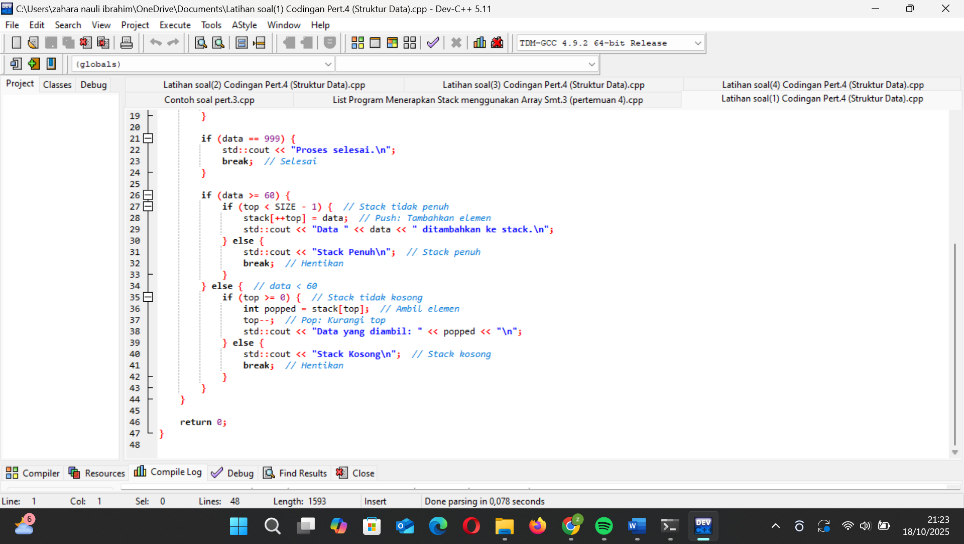
## Hasil Output



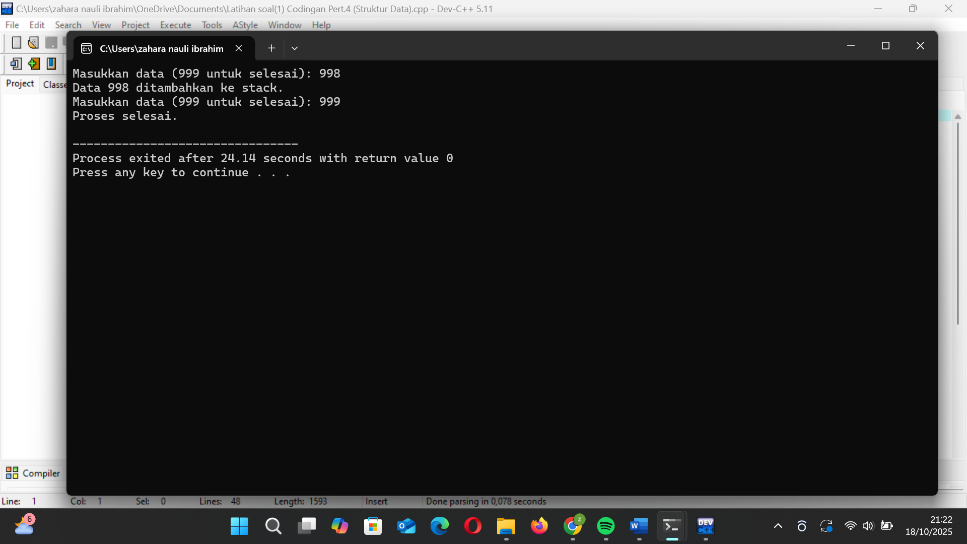
## LATIHAN 4

### 1) Program C++ stack 10 elemen: PUSH ≥ 60, POP < 60, berhenti 999.





## Hasil Output



**Penjelasan**

**Proses Input:**

* Program menggunakan std::cin >> data; untuk membaca input numerik dari keyboard. Perintah ini membaca satu nilai integer dari pengguna.
* Jika input gagal (misalnya, pengguna memasukkan huruf), program memeriksa dengan std::cin.fail(), kemudian membersihkan error menggunakan std::cin.clear() dan mengabaikan input salah dengan std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');. Ini memastikan program tidak crash dan meminta input ulang.
* Input terjadi dalam loop while (true), sehingga pengguna bisa memasukkan beberapa nilai berturut-turut, seperti 70, 50, 80, atau 999.

**Proses Pemrosesan:**

* Loop Utama: while (true) menjalankan program dalam loop tak terbatas hingga kondisi break terpenuhi. Ini memungkinkan input berulang.
* Pengecekan Input Salah: if (std::cin.fail()) memeriksa apakah input valid. Jika ya, program melanjutkan dengan continue; untuk iterasi berikutnya.
* Kondisi Utama: if (data == 999) memeriksa apakah input adalah 999, lalu break; keluar dari loop.
* Push ke Stack: if (data >= 60) memeriksa nilai. Jika benar, if (top < SIZE - 1) memeriksa apakah stack tidak penuh. Jika ya, stack[++top] = data; menambahkan elemen (increment top terlebih dahulu), lalu lanjut ke iterasi berikutnya. Jika stack penuh, break; menghentikan loop.
* Pop dari Stack: else (jika data < 60), if (top >= 0) memeriksa apakah stack tidak kosong. Jika ya, int popped = stack[top]; menyimpan elemen teratas, kemudian top--; menghapusnya (decrement top), dan lanjut. Jika stack kosong, break; menghentikan loop.
* Variabel kunci: top melacak posisi teratas stack, dan array stack[SIZE] menyimpan data.

**Proses Output:**

* std::cout << "Masukkan data (999 untuk selesai): "; menampilkan prompt untuk input.
* Untuk push sukses: std::cout << "Data " << data << " ditambahkan ke stack.\n"; mencetak pesan.
* Untuk pop sukses: std::cout << "Data yang diambil: " << popped << "\n"; mencetak elemen yang diambil.
* Untuk stack penuh: std::cout << "Stack Penuh\n"; dan program berhenti.
* Untuk stack kosong: std::cout << "Stack Kosong\n"; dan program berhenti.
* Untuk selesai: std::cout << "Proses selesai.\n";.

**Contoh (input: 70, 50, 80, 999):**

Masukkan data (999 untuk selesai): 998

Data 998 ditambahkan ke stack.

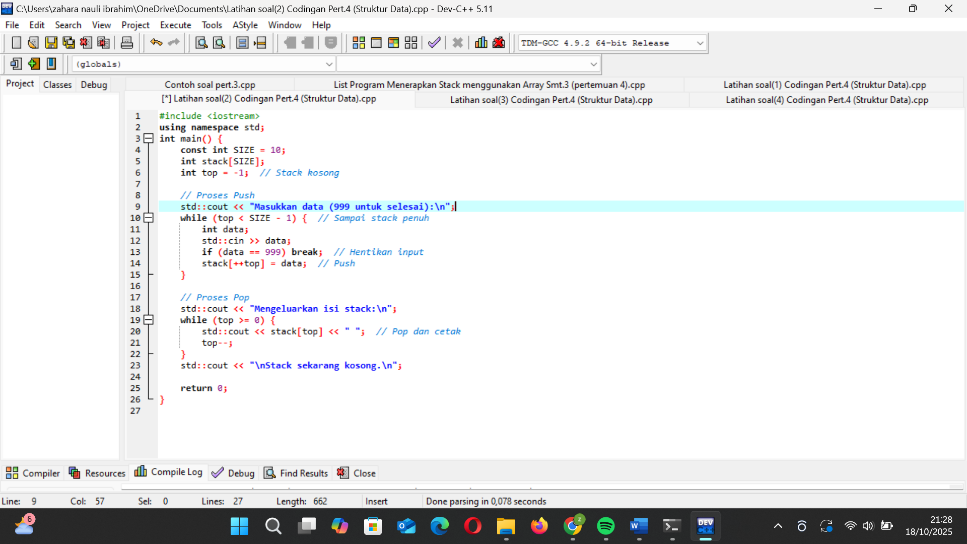
Masukkan data (999 untuk selesai): 999

Proses selesai.

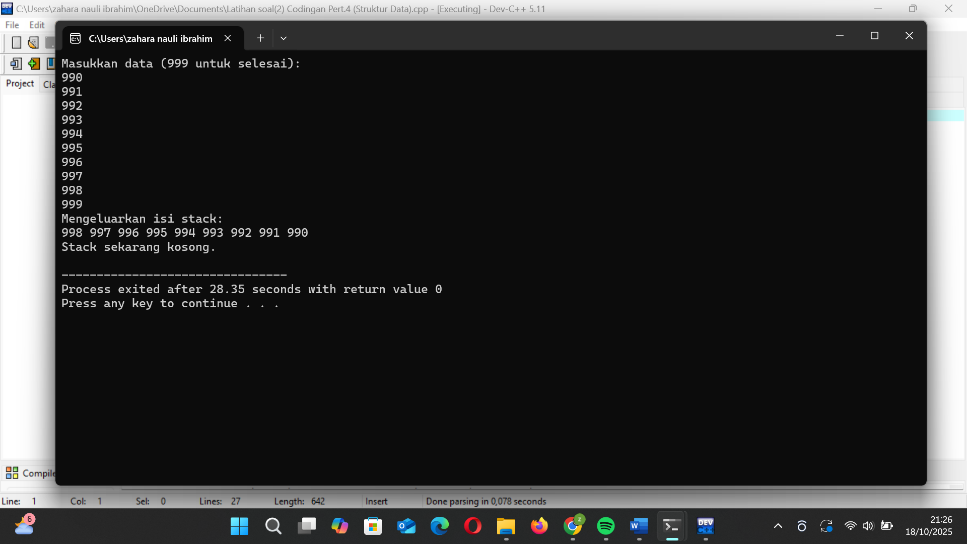
**Kesimpulan:**

Program ini menunjukkan penerapan stack dengan kondisi push/pop, menggunakan loop dan if-else untuk mengontrol alur. Ini berguna untuk memahami struktur data stack dalam skenario dinamis, seperti pengolahan data dengan aturan, dan menekankan pentingnya error handling untuk program yang robust.

### 2.) Program PUSH hingga penuh/999, lalu POP semua isi stack.



## Hasil Output



**Penjelasan**

**Proses Input:**

Program menggunakan std::cin >> data; dalam loop pertama untuk membaca input numerik dari keyboard. Pengguna memasukkan nilai satu per satu, dan loop berhenti jika input = 999 atau stack penuh. Tidak ada penanganan error input secara eksplisit, jadi asumsi input adalah numerik. Contoh input: 10, 20, 30, 999.

**Proses Pemrosesan:**

* Loop Push: while (top < SIZE - 1) menjalankan loop selama stack tidak penuh. Di dalam loop, std::cin >> data; membaca input, if (data == 999) break; menghentikan loop jika 999, dan stack[++top] = data; menambahkan elemen (increment top).
* Loop Pop: while (top >= 0) menjalankan loop selama stack tidak kosong. Di dalam loop, std::cout << stack[top] << " "; mencetak elemen, dan top--; menghapus elemen (decrement top).
* Variabel kunci: top mengontrol kedua loop, dan array stack[SIZE] menyimpan data.

**Proses Output:**

* std::cout << "Masukkan data (999 untuk selesai):\n"; menampilkan prompt awal.
* Tidak ada output selama push.
* Selama pop: std::cout << stack[top] << " "; mencetak setiap elemen, diikuti std::cout << "\nStack sekarang kosong.\n"; setelah loop selesai.

**Contoh (input: 10, 20, 30, 999):**

Masukkan data (999 untuk selesai):

Mengeluarkan isi stack:

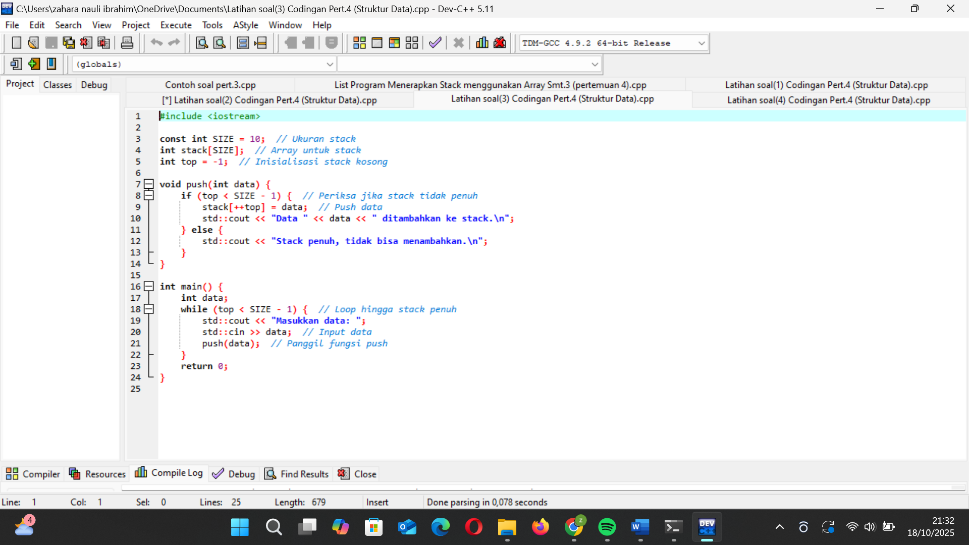
30 20 10

Stack sekarang kosong.

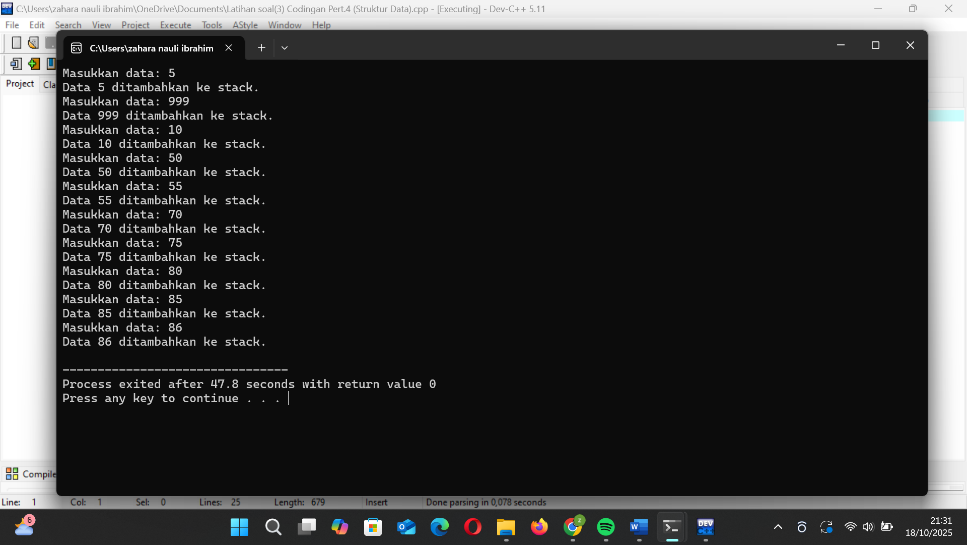
**Kesimpulan:**

Program ini mengilustrasikan siklus stack lengkap, menggunakan loop sederhana untuk push dan pop. Ini membantu memahami LIFO dalam praktik, ideal untuk aplikasi seperti undo features, tetapi perlu perbaikan untuk error handling agar lebih handal.

### 3) Potongan program PUSH hingga stack penuh.



## Hasil Output



**Penjelasan**

**Proses Input:**

Program menggunakan std::cin >> data; dalam loop while untuk membaca input numerik satu per satu dari keyboard. Perintah ini membaca nilai integer dari pengguna, seperti 10, 20, dll. Loop berlanjut hingga stack penuh, dan tidak ada penanganan error input secara eksplisit.

**Proses Pemrosesan:**

* Loop Utama: while (top < SIZE - 1) menjalankan loop selama stack tidak penuh. Perintah ini memeriksa kondisi sebelum setiap iterasi.
* Panggil Fungsi: push(data); memanggil fungsi untuk memproses data. Di dalam fungsi:
* if (top < SIZE - 1) memeriksa apakah stack bisa diisi. Jika ya, stack[++top] = data; menambahkan elemen (increment top dan simpan data).
* Jika kondisi salah, std::cout << "Stack penuh, tidak bisa menambahkan.\n"; mencetak pesan dan fungsi selesai.
* Variabel: top melacak posisi, dan stack[SIZE] adalah array untuk menyimpan data.

**Proses Output:**

* Untuk push sukses: std::cout << "Data " << data << " ditambahkan ke stack.\n"; mencetak pesan dalam fungsi push.
* Saat stack penuh: std::cout << "Stack penuh, tidak bisa menambahkan.\n"; mencetak pesan dan loop berhenti.

**Contoh (input: 10, 20, ..., hingga 10 elemen):**

Masukkan data: 10

Data 10 ditambahkan ke stack.

Masukkan data: 20

Data 20 ditambahkan ke stack.

... (lanjut)

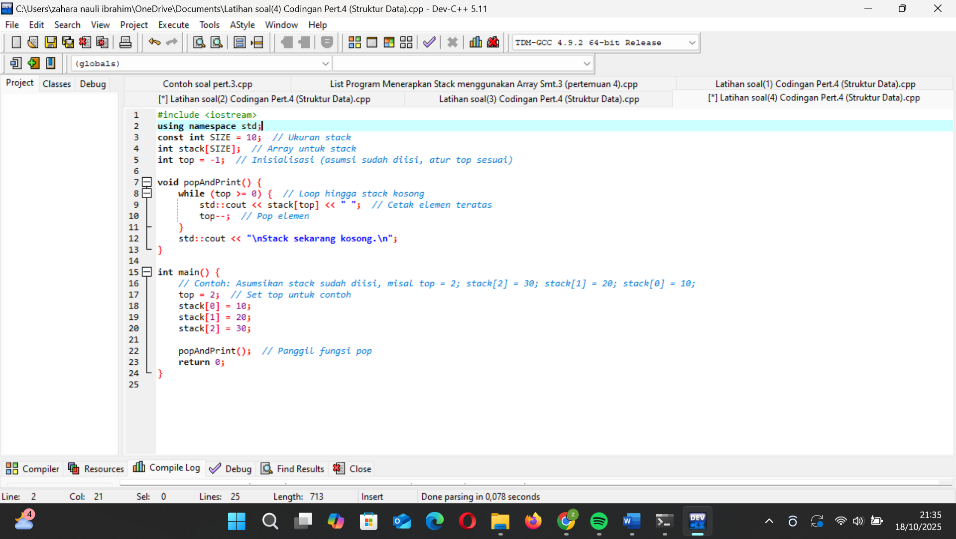
Masukkan data: (elemen ke-11)

Stack penuh, tidak bisa menambahkan.

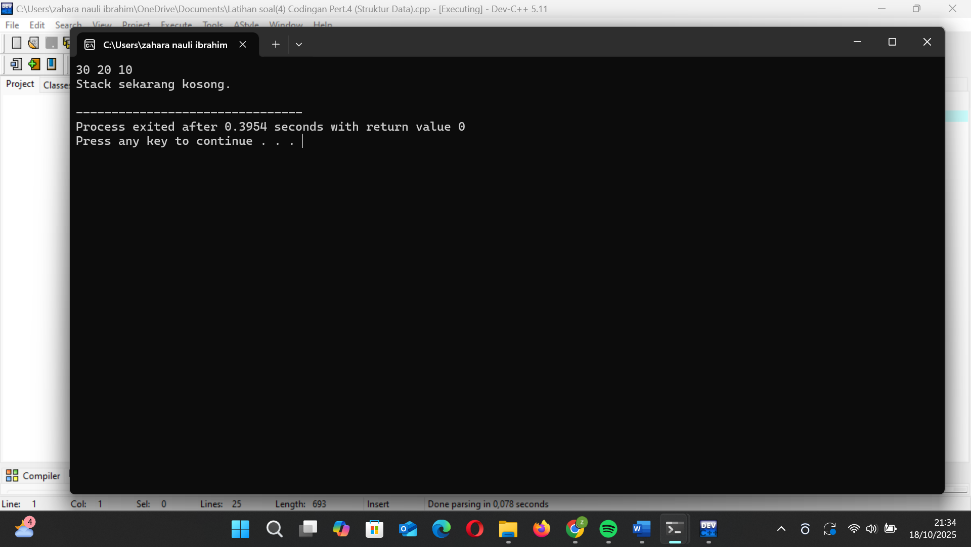
**Kesimpulan:**

Potongan ini menunjukkan cara mengisi stack secara dinamis dengan input berulang, menggunakan loop dan fungsi untuk modularitas. Ini berguna untuk memahami push operation dan batas stack, tetapi kekurangannya adalah kurangnya error handling untuk input non-numerik.

### 4.) Potongan program POP hingga stack kosong.



## Hasil Output



**Penjelasan**

**Proses Input:**

Tidak ada input dari pengguna; program mengasumsikan stack sudah diisi. Dalam main(), array stack dan top diatur secara manual, seperti top = 2; dan stack[0] = 10;.

**Proses Pemrosesan:**

* Fungsi Pop: popAndPrint() menggunakan loop while (top >= 0) untuk memproses stack selama tidak kosong.
* Di dalam loop: std::cout << stack[top] << " "; mencetak elemen teratas, dan top--; menghapus elemen (decrement top).
* Loop berhenti saat top = -1. Variabel top mengontrol alur, dan array stack menyimpan data yang sudah ada.

**Proses Output:**

Selama pop: std::cout << stack[top] << " "; mencetak setiap elemen. Setelah loop, std::cout << "\nStack sekarang kosong.\n"; mencetak pesan akhir.

**Contoh (stack awal: [10, 20, 30], top=2):**

30 20 10

Stack sekarang kosong.

**Kesimpulan:**

Potongan ini menangani operasi pop secara sederhana, menunjukkan LIFO dalam aksi. Ini berguna untuk aplikasi seperti pencetakan stack, tetapi memerlukan inisialisasi eksternal, menekankan pentingnya setup awal dalam struktur data.

### 5.) Ciri stack: kosong, penuh, bisa diisi, ada isinya.

**Penjelasan Teori:**

Single stack adalah struktur data LIFO yang menggunakan array. Berikut ciri-cirinya berdasarkan kondisi, dengan penjelasan sederhana:

1. Kosong:

Ciri: top = -1 (atau nilai awal, menandakan tidak ada elemen).

* Penjelasan: Stack tidak memiliki elemen, jadi operasi pop tidak mungkin. Operasi push masih bisa dilakukan jika stack tidak penuh.

1. Penuh:

Ciri: top = SIZE - 1 (misalnya, untuk SIZE=10, top=9).

* Penjelasan: Stack mencapai kapasitas maksimal, jadi operasi push tidak mungkin. Operasi pop masih bisa dilakukan jika ada elemen.

1. Bisa diisi:

Ciri: top < SIZE - 1 (ada ruang kosong di array).

* Penjelasan: Stack memiliki kapasitas untuk menambahkan elemen, yang merupakan kondisi normal sebelum mencapai batas. Ini memungkinkan operasi push.

1. Ada isinya:

Ciri: top >= 0 (setidaknya satu elemen ada).

* Penjelasan: Stack memiliki elemen, jadi operasi pop mungkin. Ini menandakan stack tidak kosong.

**Kesimpulan:**

Penjelasan ini menggambarkan dasar pengelolaan stack, membantu dalam mendeteksi kondisi untuk operasi yang aman. Ini penting untuk aplikasi seperti manajemen memori atau algoritma rekursif, di mana stack digunakan untuk tumpukan operasi, dan memahami variabel top adalah kunci untuk implementasi yang benar.

**LAPORAN AKHIR**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

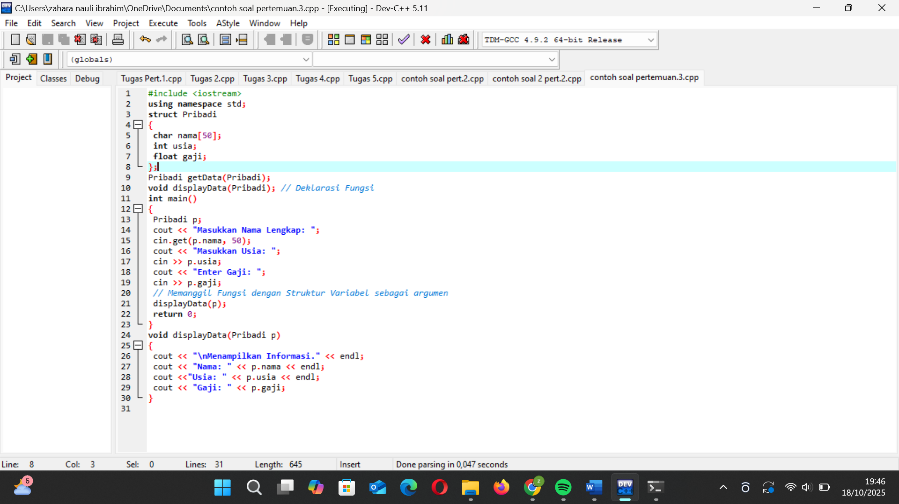
Tangerang Selatan – Banten

# LAPORAN PRAKTIKUM

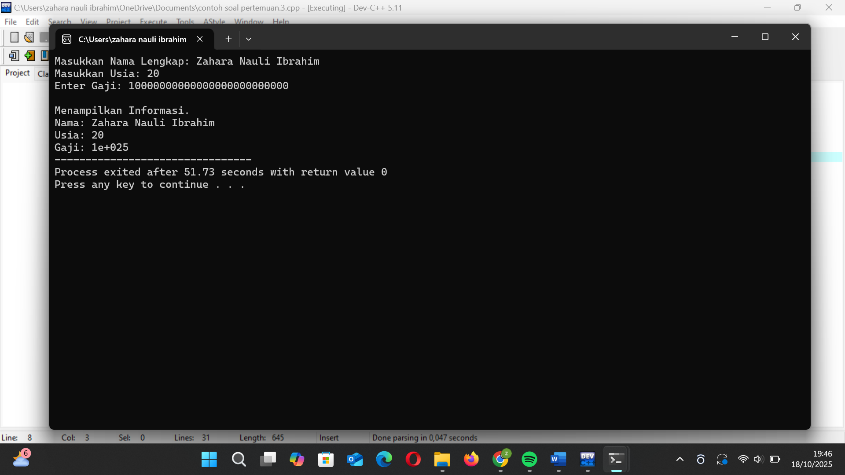
1) Program untuk menetapkan data anggota struktur variabel

## 

## 2) Program penggunaan fungsi getData() dan displayData()

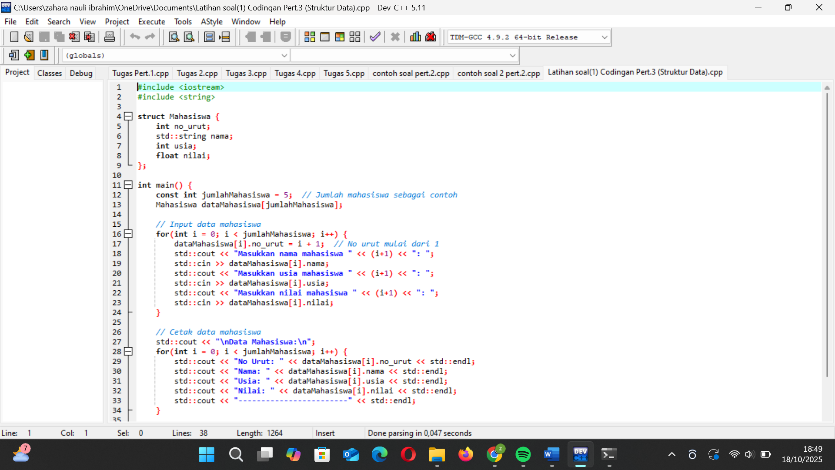


**Hasil Output**

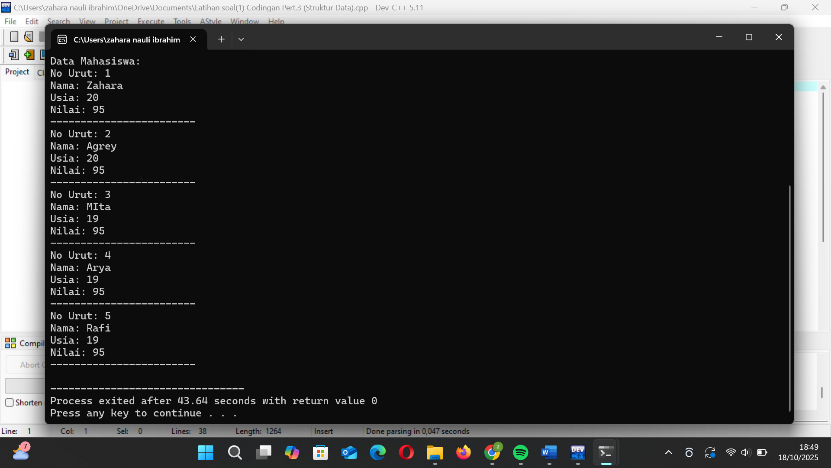


## LATIHAN 3

### 1) Tulis program untuk menyimpan dan mencetak No urut, Nama, usia dan nilai mahasiswa menggunakan struktur.



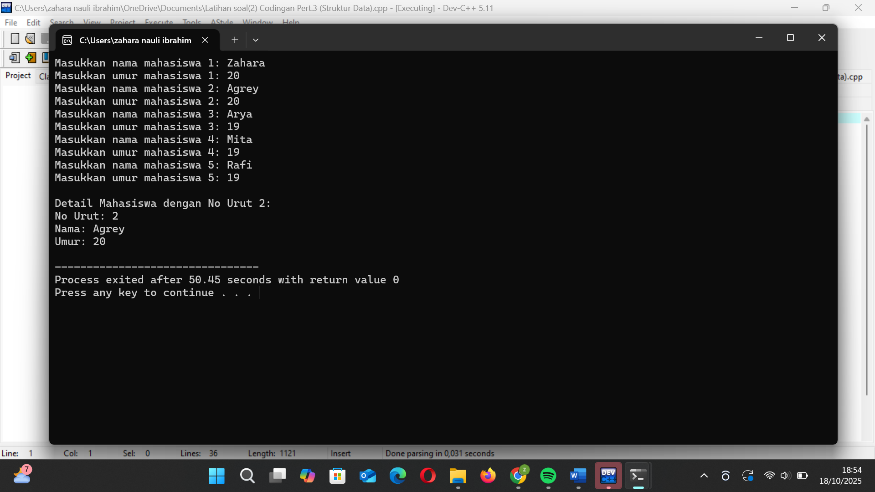
## Hasil Output



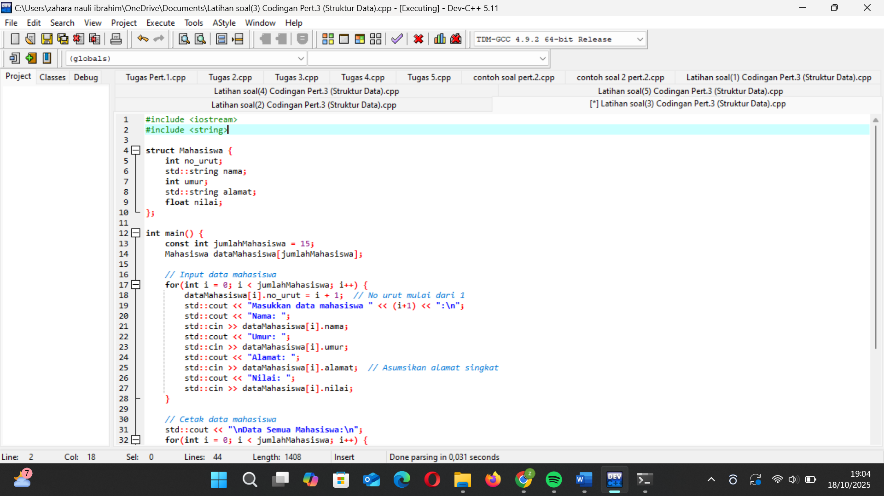
### 2) Tulis program untuk menyimpan No urut. (mulai dari 1), nama dan umur 5 Mahasiswa kemudian cetak detail Mahasiswa dengan No urut 2.

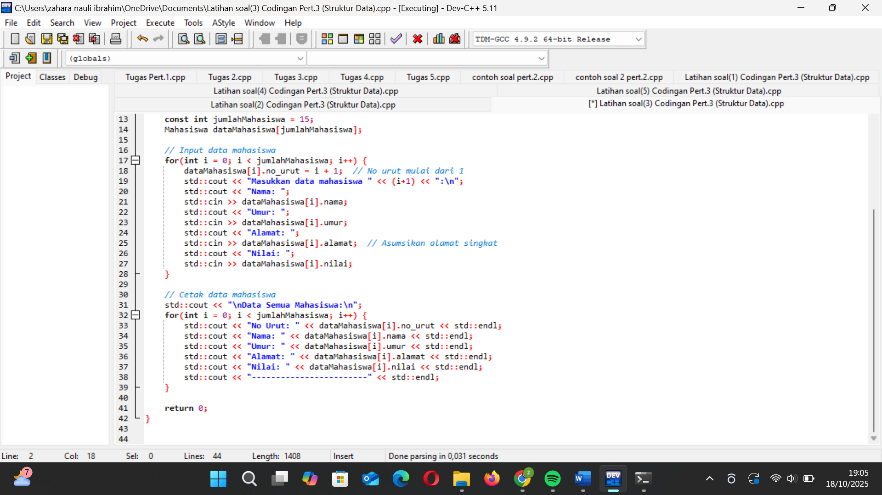


## Hasil Output

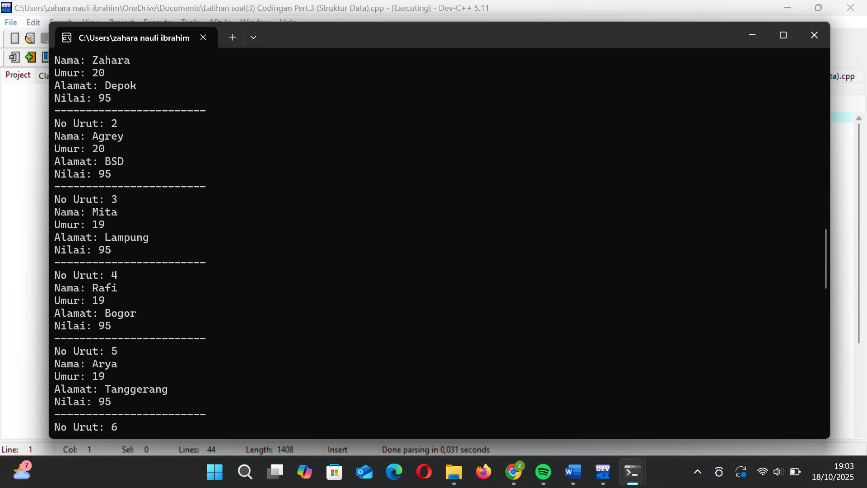


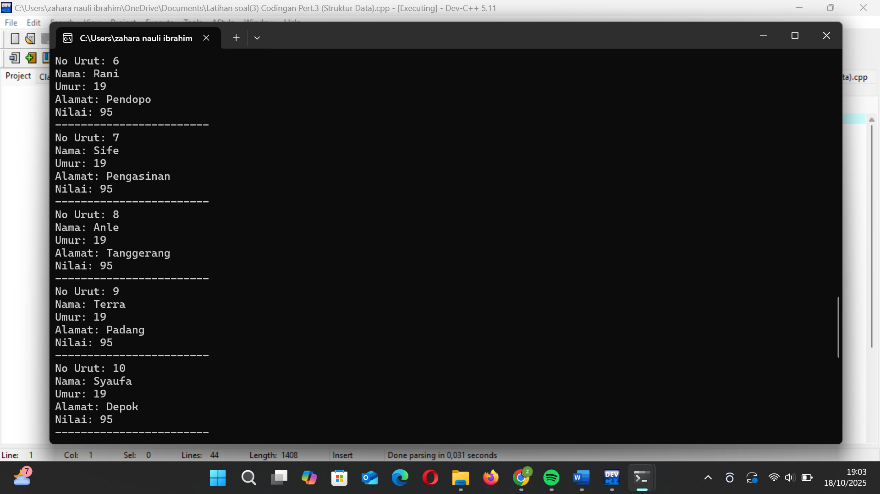
### 3) Tulis Program untuk menyimpan dan mencetak no. urut, nama, umur, alamat dan nilai 15 mahasiswa menggunakan struktur.

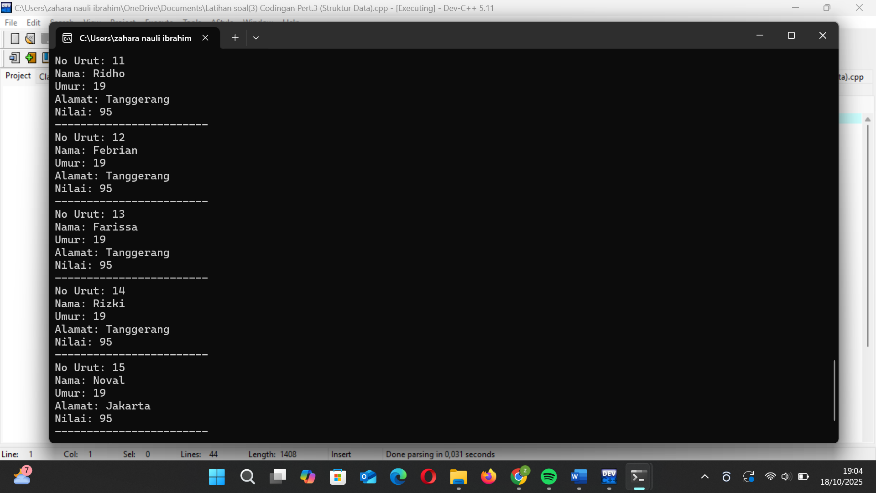




## Hasil Output

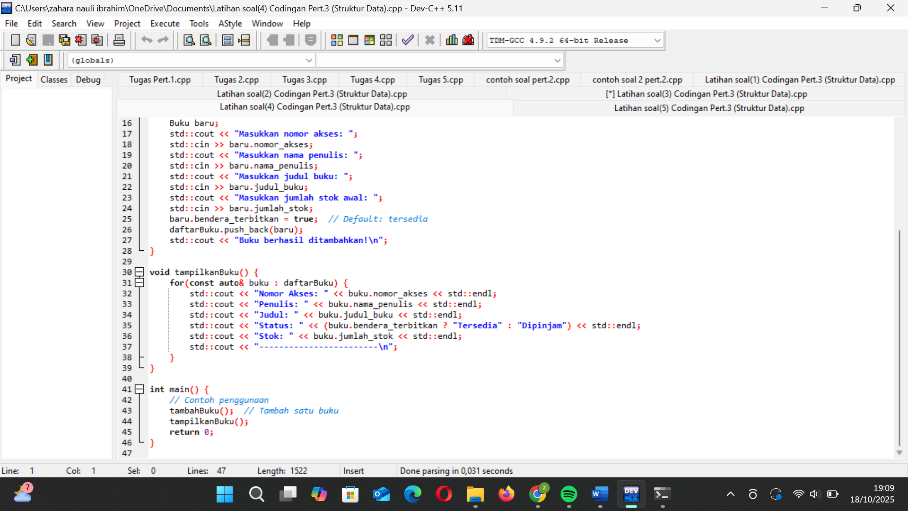




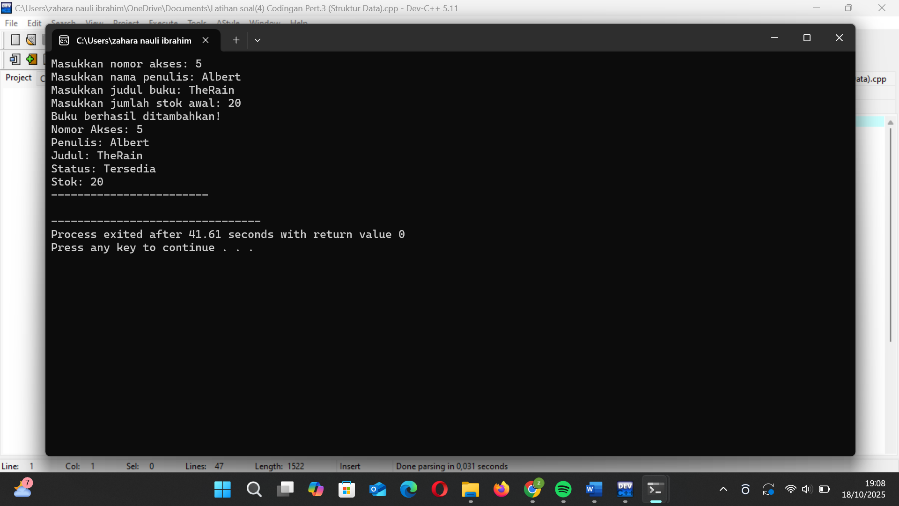


### 4) Kerjakan menu perpustakaan. Buat struktur yang berisi informasi buku seperti nomor akses, nama penulis, judul buku untuk mengetahui apakah buku tersebut diterbitkan atau tidak.





## Hasil Output



### 5) Buat menu di mana hal berikut dapat dilakukan.

### - Menampilkan informasi buku

### - Tambahkan buku baru

### - Tampilkan semua buku di perpustakaan penulis tertentu

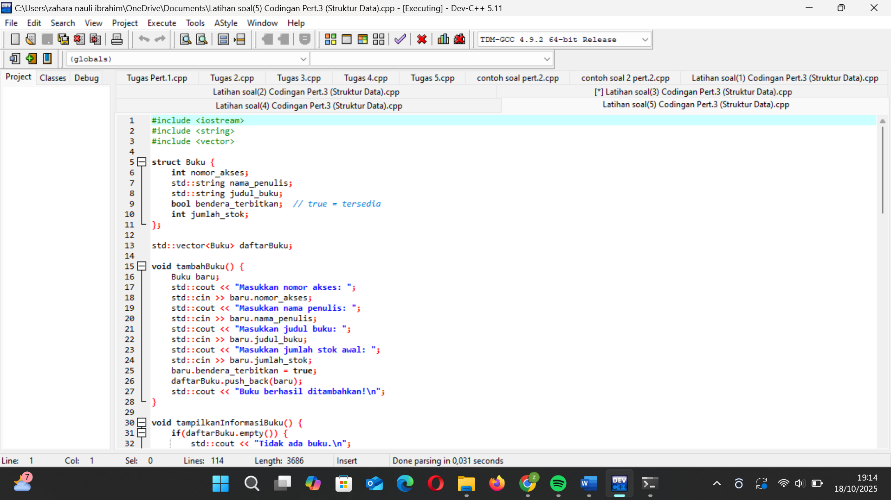
### - Menampilkan jumlah buku dengan judul tertentu

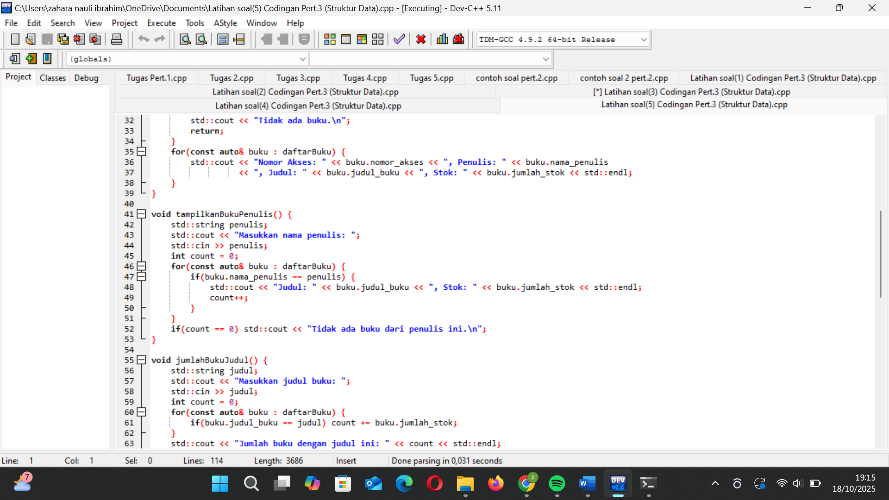
### - Menampilkan jumlah total buku di perpustakaan

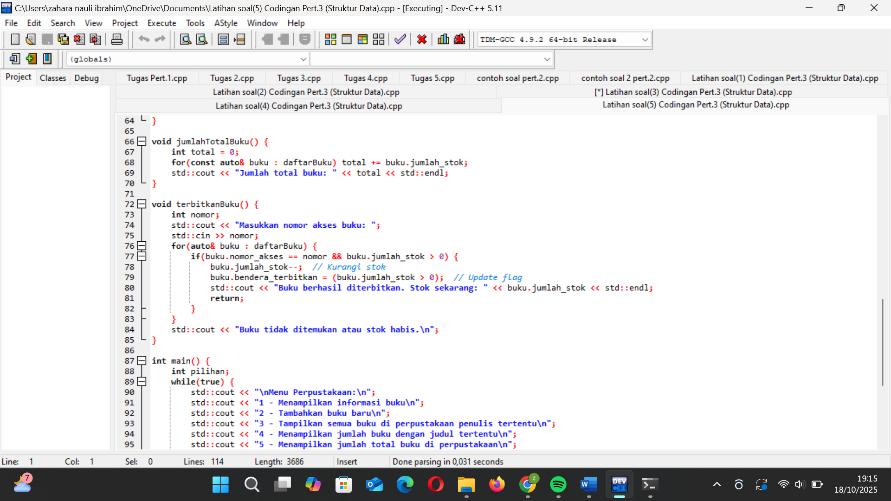
### - Terbitkan buku

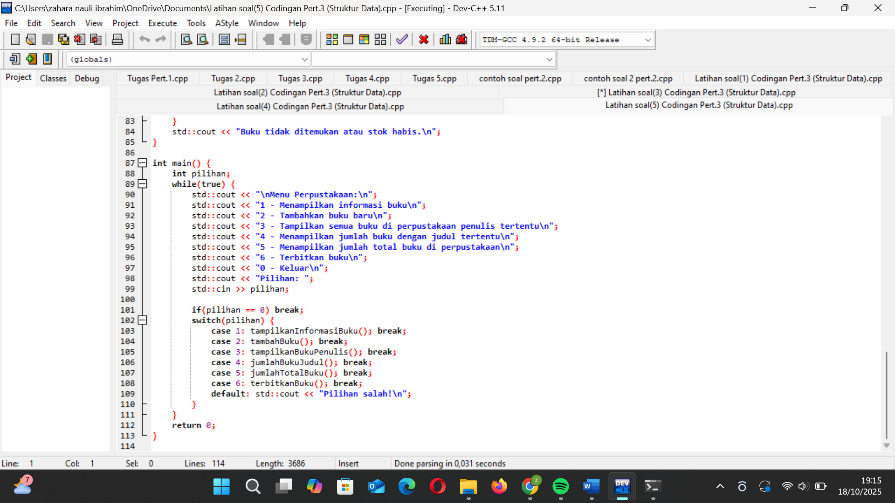
### (Jika kita menerbitkan buku, maka jumlahnya akan berkurang 1 dan jika

### kita menambahkan buku, jumlahnya bertambah 1)

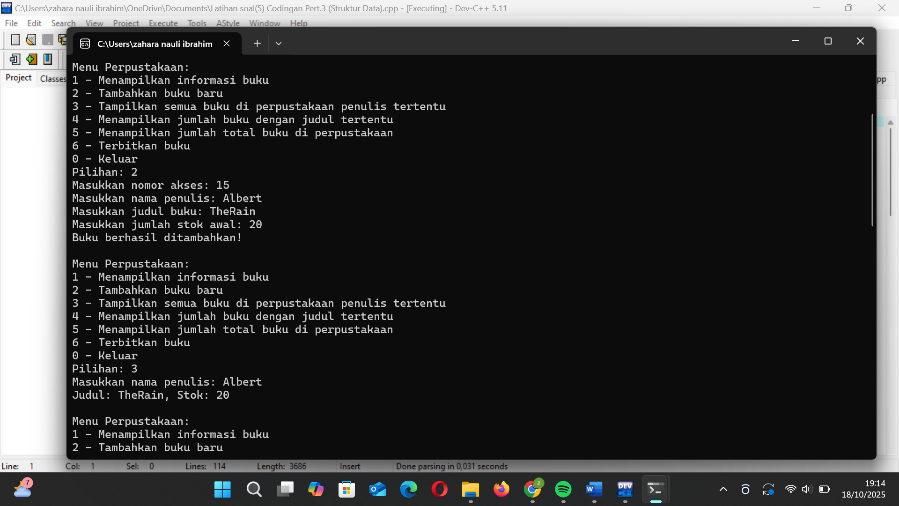


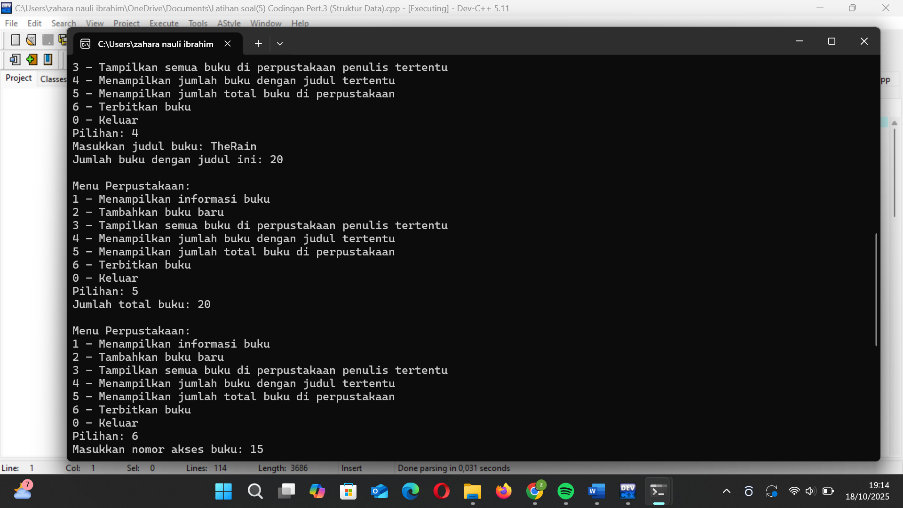






## Hasil Output





# HASIL DAN PEMBAHASAN

## LATIHAN 3

## Soal 1. Program Menyimpan dan Mencetak Data Mahasiswa

### Pembahasan:

* Program membuat satu objek Mahasiswa dengan atribut nomor, nama, usia, dan nilai.
* Pengguna menginput data, lalu program mencetak ulang hasil input.
* Fungsi deklarasi struct adalah menggabungkan beberapa tipe data berbeda agar mudah dikelola dalam satu variabel (mhs).

### Kesimpulan:

Struct mempermudah penyimpanan data kompleks dan meminimalkan penggunaan banyak variabel terpisah.

## Soal 2. Program Menyimpan dan Menampilkan Data Mahasiswa ke-2

### Pembahasan:

* Struct Mahasiswa digunakan sama seperti soal 1, tetapi kali ini dibuat dalam bentuk array mhs[5].
* Program meminta input data nama dan umur untuk 5 mahasiswa.
* Nomor urut otomatis dihasilkan dengan mhs[i].no = i + 1.
* Setelah semua data diinput, hanya data mahasiswa nomor ke-2 yang ditampilkan.

Fungsi Deklarasi Struct:

* Membuat template yang bisa disalin menjadi banyak objek dalam array, sehingga data mahasiswanya dapat diakses melalui indeks (misalnya mhs[1]).

### Kesimpulan:

Array of struct memungkinkan pengelolaan data banyak secara terstruktur dan mudah diakses berdasarkan indeks.

## Soal 3. Program Menyimpan dan Mencetak 15 Data Mahasiswa

### Pembahasan:

* Struct Mahasiswa kali ini memiliki atribut tambahan: alamat dan nilai.
* Program mengulang input sebanyak 15 kali dengan perulangan for.
* Setiap mahasiswa diberi nomor otomatis dan datanya ditampilkan kembali dalam format tabel.
* Deklarasi struct membantu menyimpan data kompleks dari 15 mahasiswa tanpa membuat variabel terpisah untuk setiap atribut.

### Kesimpulan:

Dengan memanfaatkan array dan looping, struct dapat digunakan untuk membuat database sederhana mahasiswa yang mudah dikelola dan ditampilkan.

## Soal 4. Program Struktur Menu Perpustakaan Sederhana

### Pembahasan:

* Struct Buku berisi atribut nomorAkses, penulis, judul, dan diterbitkan.
* Program meminta input data buku, lalu mencetak kembali datanya.
* Nilai boolean diterbitkan digunakan untuk menunjukkan status apakah buku sudah terbit atau belum.
* Tanda cin.ignore() dan getline() digunakan agar input string bisa dibaca dengan spasi.

Fungsi Deklarasi Struct:

* Menjadikan setiap buku sebagai satu objek yang berisi seluruh informasinya. Hal ini memudahkan pengelolaan data perpustakaan yang lebih realistis.

### Kesimpulan:

Struktur dapat digunakan untuk berbagai objek nyata selain mahasiswa, seperti buku, kendaraan, atau pegawai.

## Soal 5. Program Menu Penerbitan Buku Perpustakaan

Mempraktikkan konsep array of struct dan kontrol menu program untuk sistem pengelolaan data yang lebih kompleks.

### Pembahasan:

* Struct Buku menyimpan atribut judul, penulis, jumlah, dan terbit.
* Program menggunakan array struct untuk menyimpan hingga 100 data buku.
* Menu interaktif dibuat menggunakan perulangan do-while dan switch-case.
* Pengguna dapat:

1. Menampilkan daftar buku
2. Menambah buku baru
3. Menampilkan buku berdasarkan penulis
4. Menghitung jumlah buku berdasarkan judul
5. Menghitung total seluruh buku
6. Menandai buku yang diterbitkan

* Deklarasi struct digunakan sebagai model data utama untuk semua operasi di menu.

Fungsi Algoritma:

* Program ini memperlihatkan bagaimana struktur digunakan dalam pengolahan data dinamis, seperti penambahan, pencarian, dan perhitungan data.

### Kesimpulan:

Struktur (struct) dapat digunakan untuk membuat sistem data terapan seperti manajemen buku, karyawan, atau inventaris.

Dengan kombinasi array dan perulangan, program mampu berfungsi sebagai mini database.

# KESIMPULAN

1. Struktur (struct) memungkinkan penggabungan berbagai tipe data dalam satu kesatuan logis.
2. Dengan array dan perulangan, struct dapat menyimpan dan mengelola banyak data sekaligus.
3. Struct mempermudah pembuatan aplikasi berbasis data seperti daftar mahasiswa, sistem perpustakaan, dan inventaris barang.
4. Penerapan konsep struct merupakan langkah awal menuju pemrograman berorientasi objek (OOP).