LAPORAN PRAKTIKUM

PEMROGRAMAN DASAR

ARRAY

NAMA: Farid Gantari Siwanda

NIM: 25104410016

PERIODE: SEMESTER GANJIL 2024/2025

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ISLAM BALITAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, efisiensi pengelolaan data merupakan aspek yang sangat krusial. Seringkali, seorang programmer perlu menangani sekumpulan data yang memiliki tipe yang sama, seperti daftar nilai mahasiswa atau koordinat titik dalam ruang. Menggunakan variabel terpisah untuk setiap data tersebut tentu tidak praktis. Oleh karena itu, konsep **Array** (larik) hadir sebagai solusi untuk menyimpan sekumpulan data dengan tipe yang sama dalam satu nama variabel tunggal.

Array tidak hanya terbatas pada satu dimensi. Dalam implementasi perhitungan matematika yang lebih kompleks, kita sering menggunakan array dua dimensi yang lebih dikenal sebagai **Matriks**. Matriks sangat berperan penting dalam berbagai bidang, mulai dari pengolahan citra digital, grafika komputer, hingga penyelesaian sistem persamaan linear.

1.2 Tujuan Praktikum

Adapun tujuan dari praktikum ini adalah:

1. Memahami konsep dasar penggunaan array 1 dimensi dan 2 dimensi (matriks) dalam C++.
2. Mempelajari cara menghitung panjang array secara dinamis menggunakan operator **sizeof**.
3. Mengimplementasikan operasi aritmatika pada matriks, khususnya penjumlahan dan perkalian.

BAB II

DASAR TEORI

Array atau larik merupakan salah satu struktur data dasar dalam pemrograman yang digunakan untuk menyimpan sekumpulan data dengan tipe yang sama dalam satu variabel. Setiap elemen dalam array memiliki indeks yang digunakan untuk mengakses nilai tertentu, sehingga memudahkan pengelolaan data dalam jumlah besar. Penggunaan array memungkinkan program menjadi lebih efisien, terorganisir, dan mudah dikembangkan dibandingkan dengan penggunaan banyak variabel terpisah.

Array dapat memiliki satu dimensi maupun lebih dari satu dimensi. Array satu dimensi umumnya digunakan untuk menyimpan data dalam bentuk daftar atau deret, seperti nilai mahasiswa atau data sensor. Sementara itu, array dua dimensi dikenal sebagai matriks, yang menyusun data dalam bentuk baris dan kolom. Matriks banyak digunakan dalam berbagai bidang, seperti pengolahan citra digital, grafika komputer, dan penyelesaian sistem persamaan linear, karena mampu merepresentasikan data yang bersifat dua arah secara sistematis.

Dengan memahami konsep array dan matriks, seorang programmer dapat mengelola data secara lebih efektif serta menerapkan solusi pemrograman yang efisien untuk permasalahan yang bersifat kompleks dan multidimensional.

BAB III

3.1 Penjelasan Program Penjumlahan Matriks dan Hasil Outputnya

Program ini (**PrakArray_matriks_penjumlahan.cpp**) menjumlahkan dua matriks berukuran **3X3**. Penjumlahan matriks dilakukan dengan menjumlahkan elemen yang berada pada posisi (indeks) yang sama

a. Bagian Utama Kode:

- **Deklarasi Matriks**: **int A[3][3], B[3][3], C[3][3];** menyiapkan tiga wadah array dua dimensi untuk menyimpan angka-angka matriks.
- **Input Data**: Program menggunakan dua *nested loop* (perulangan bersarang) untuk meminta pengguna memasukkan nilai ke dalam Matriks A dan Matriks B.
- **Logika Penjumlahan**:

```
C[i][j] = A[i][j] + B[i][j];
```

  Bagian ini adalah inti dari program. Nilai pada baris **i** dan kolom **j** dari matriks A ditambahkan dengan nilai pada posisi yang sama di matriks B, lalu hasilnya disimpan di matriks C.

b. Hasil Output

Output tersebut menunjukkan keberhasilan program dalam memproses input pengguna hingga menghasilkan matriks baru dan informasi kapasitas memori. Berikut poin-poin penjelasannya:

- **Input Matriks A dan B**: User memasukkan 9 elemen untuk Matriks A dan 9 elemen untuk Matriks B dalam format **3X3**. Data ini disimpan ke dalam array dua dimensi **A[3][3]** dan **B[3][3]**.
- **Proses Penjumlahan**: Program menjumlahkan setiap elemen pada baris dan kolom yang sama. Sebagai contoh:
  - Baris 1: 6+3=9, 6+2=8, 6+1=7.
  - Baris 2: 7+4=11, 7+4=11, 7+4=11.
  - Baris 3: 9+5=14, 6+5=11, 3+5=8.
- **Hasil Penjumlahan Matriks**: Program menampilkan isi dari Matriks C yang merupakan hasil akumulasi dari Matriks A dan B tadi. Tampilan dicetak secara rapi dalam bentuk persegi menggunakan manipulasi **cout << endl**; setelah setiap baris selesai.
- Informasi Jumlah Elemen: Pada baris terakhir, muncul angka 9. Ini adalah hasil dari logika **sizeof(C) / sizeof(C[0][0])**. Meskipun matriks diisi dengan angka-angka yang berbeda, kapasitas total sel yang tersedia dalam struktur data tersebut tetaplah

9 sel (berasal dari **3X3**).

```
Masukkan elemen matriks A (3x3):
6 6 6
7 7 7
9 6 3
Masukkan elemen matriks B (3x3):
3 2 1
4 4 4
5 5 5

Hasil Penjumlahan Matriks:
9 8 7
11 11 11
14 11 8

Jumlah elemen matriks: 9
```

3.2 Penjelasan Program Perkalian Matriks dan Hasil Outputnya

Program ini (**PrakArray_matriks_perkalian.cpp**) melakukan perkalian dot product antara dua matriks **3X3**.

a. Bagian Utama Kode:

- **Inisialisasi Matriks C**: **C[3][3] = {0};** sangat penting karena pada perkalian matriks, kita melakukan operasi akumulasi (penambahan berulang). Matriks hasil harus dimulai dari angka 0.
- **Logika Perkalian (Triple Nested Loop)**: Berbeda dengan penjumlahan, perkalian matriks membutuhkan tiga lapis perulangan:
  - **Loop i**: Menentukan baris pada matriks A.
  - **Loop j**: Menentukan kolom pada matriks B.
  - **Loop k**: Digunakan untuk berjalan menyamping di baris A dan menurun di kolom B secara bersamaan.

    ```
    C[i][j] += A[i][k] * B[k][j];
    ```

    Rumus ini berarti: Elemen baris **i** kolom **j** pada hasil (C) adalah jumlah dari perkalian elemen baris **i** matriks A dengan elemen kolom **j** matriks B.

b. Hasil Output

Output tersebut menunjukkan program telah berhasil mengeksekusi algoritma perkalian matriks (*dot product*) terhadap dua matriks berukuran **3X3**. Berikut adalah detail penjelasannya:

- **Input Elemen Matriks A dan B**: Pengguna memasukkan sembilan nilai untuk matriks A dan sembilan nilai untuk matriks B. Data ini disimpan ke dalam array dua dimensi **A[3][3]** dan **B[3][3]**.
- **Proses Perkalian (Dot Product)**: Program menghitung setiap elemen matriks hasil dengan cara mengalikan baris pada matriks A dengan kolom pada matriks B, lalu

menjumlahkan hasilnya. Sebagai contoh, nilai **133** pada baris pertama dihasilkan dari:
(7 * 5) + (7 * 5) + (7 * 9) = 35 + 35 + 63 = 133.

- **Hasil Perkalian Matriks**: Program menampilkan matriks hasil **(C)** dalam bentuk grid **3X3** yang rapi. Setiap baris pada output mewakili hasil perhitungan akumulasi dari baris A terhadap seluruh kolom B.

```
Masukkan elemen matriks A (3x3):
7 7 7
4 4 4
9 1 1
Masukkan elemen matriks B (3x3):
5 0 5
5 5 5
9 8 7

Hasil Perkalian Matriks:
133 91 119
76 52 68
59 13 57

Jumlah elemen matriks: 9
```

3.3 Penjelasan Perhitungan Jumlah Elemen Matriks Menggunakan Fungsi sizeof ()

```
// Menggunakan sizeof()
int jumlahElemen = sizeof(C) / sizeof(C[0][0]);
cout << "\nJumlah elemen matriks: " << jumlahElemen << endl;
```

Baris kode tersebut diawali dengan sebuah komentar yang menjelaskan bahwa fungsi **sizeof()** akan digunakan untuk menghitung karakteristik array. Program kemudian mendeklarasikan variabel **jumlahElemen** bertipe integer untuk menyimpan hasil perhitungan total sel dalam matriks. Proses penghitungan dilakukan dengan membagi total ukuran memori keseluruhan matriks yang didapat dari **sizeof(C)** dengan ukuran memori satu elemen tunggalnya, yaitu **sizeof(C[0][0])**. Karena matriks ini memiliki dimensi **3X3**, hasil pembagian tersebut akan menghasilkan angka 9 yang merepresentasikan total kapasitas elemen di dalamnya. Terakhir, nilai tersebut ditampilkan ke layar menggunakan perintah **cout** dengan tambahan teks penjelas agar pengguna dapat melihat jumlah elemen yang tersedia dalam matriks tersebut.

BAB IV

KESIMPULAN

Berdasarkan praktikum yang telah dilaksanakan, dapat disimpulkan bahwa penggunaan struktur data array, baik satu dimensi maupun dua dimensi (matriks), merupakan solusi efisien untuk mengelola sekumpulan data bertipe homogen dalam satu variabel tunggal. Melalui program yang telah diuji, praktikan berhasil memahami bahwa operasi penjumlahan matriks dilakukan dengan menjumlahkan elemen pada indeks yang sama melalui perulangan bersarang. Sementara itu, operasi perkalian matriks memerlukan logika yang lebih kompleks berupa *triple nested loop* untuk melakukan akumulasi hasil perkalian antara elemen baris matriks pertama dengan elemen kolom matriks kedua. Selain itu, penggunaan operator **sizeof** terbukti sangat efektif untuk menentukan jumlah elemen dalam struktur data secara dinamis dengan cara membagi total ukuran memori matriks dengan ukuran satu elemen tunggalnya. Ketelitian dalam inisialisasi nilai awal, seperti pada matriks hasil perkalian, serta teknik pemformatan output sangat menentukan keakuratan dan keterbacaan hasil akhir program dalam bentuk grid yang rapi.