VERSI 1.3

JULI, 2023

[PEMROGRAMAN DASAR]

MODUL 5 - ARRAY

DISUSUN OLEH:

KIARA AZZAHRA

ANNISA ARTANTI WIDYADHANA

DIAUDIT OLEH:

HARDIANTO WIBOWO, S.KOM, M.T

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[PEMROGRAMAN DASAR]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain :

- Array
- Loop Array
- Array Multidimensi

TUJUAN

- Mahasiswa memahami konsep array sebagai struktur data untuk menyimpan sekumpulan nilai yang serupa dalam satu variabel, sehingga dapat mengelola dan memanipulasi data secara efisien.

TARGET MODUL

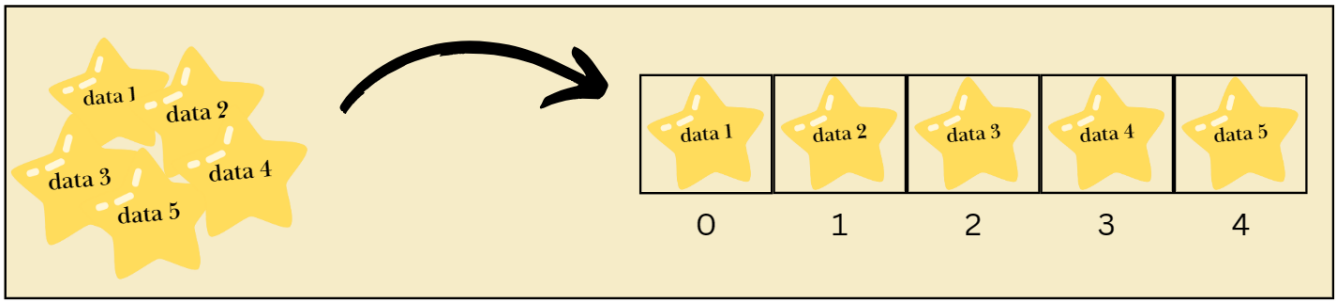
- Mampu mendeklarasikan, menginisialisasi, dan mengakses elemen array, serta memahami penggunaan array dalam berbagai kasus pemrograman.

PERSIAPAN SOFTWARE/APLIKASI

- Laptop/komputer
- Software IDE (Falcon/Dev C++)

ARRAY

Array adalah kumpulan elemen dengan tipe data yang sama, yang diidentifikasi oleh nama dan indeks. Array memungkinkan kita untuk menyimpan sekumpulan data dalam satu variabel, sehingga memudahkan akses dan pengelolaan data secara berurutan. Dalam suatu array, setiap elemennya diidentifikasi oleh nomor posisi atau disebut juga dengan **nomor index** yang dimulai dari 0 hingga jumlah elemen dikurangi 1. Dengan menggunakan array, kita bisa lebih mudah untuk mengakses dan mengelola data dan penyimpanan data yang terstruktur sehingga data di dalam array lebih terorganisir.



Untuk memperjelas apa itu array, perhatikan gambar berikut :

Egg tray (nampan telur) adalah ibarat dari **array**, sedangkan telur adalah **elemennya.** Kalian bisa memodifikasi nampan telur itu dengan **mengeluarkan** sebuah telur atau kalian juga bisa **menggunakan** telur apapun secara khusus di posisi manapun.

Jadi array digunakan untuk menyimpan elemen/data secara kontinu, satu demi satu seperti nampan telur dalam menyimpan telur. Index pada array membantu kalian untuk melakukan berbagai operasi pada array seperti mengakses elemen, memodifikasi array,

Menggunakan array dalam pemrograman memiliki beberapa keuntungan yang membuatnya menjadi struktur data yang sangat berguna. Berikut adalah beberapa keuntungan utama dalam menggunakan array:

1. Penyimpanan Data yang Terstruktur: Array memungkinkan kalian menyimpan data dalam struktur teratur dan terstruktur. Kalian dapat mengatur dan mengakses data dengan mudah berdasarkan indeksnya.
2. Alokasi Memori Efisien: Array mengalokasikan memori secara berurutan untuk elemen-elemen datanya. Hal ini menghasilkan alokasi memori yang efisien dan pengaksesan data yang cepat.
3. Mengelompokkan Data: Array memungkinkan kalian mengelompokkan data yang serupa dalam satu entitas. Ini membantu dalam pengelolaan data yang lebih baik dan memudahkan akses ke data yang relevan.
4. Pengaksesan Data Cepat: Kalian dapat mengakses elemen dalam array dengan cepat menggunakan indeks. Proses ini memiliki kompleksitas waktu O(1), yang berarti waktu aksesnya tidak tergantung pada ukuran array.
5. Iterasi dan Pengulangan Mudah: Array sangat cocok untuk pengulangan (looping) dan iterasi data. Kalian dapat dengan mudah mengakses setiap elemen dalam array menggunakan loop.
6. Penyimpanan Data Homogen: Array hanya dapat menyimpan elemen dengan tipe data yang sama. Ini membantu menjaga integritas dan konsistensi data di dalam array.
7. Pemeliharaan Kode yang Mudah: Penggunaan array dapat membuat kode lebih mudah dipelihara karena elemen-elemen terkait tersimpan dalam satu struktur.
8. Penggunaan Fungsi Lebih Efisien: Dalam banyak kasus, array dapat digunakan sebagai argumen dalam fungsi, memungkinkan kalian untuk memproses sejumlah data dengan lebih efisien.
9. Pengurangan Kode Duplikasi: Array memungkinkan kalian menyimpan data yang sama dalam satu tempat. Ini membantu mengurangi duplikasi kode dan memastikan konsistensi data.
10. Implementasi Struktur Data Lain: Banyak struktur data lain, seperti stack, queue, dan matrix, dapat diimplementasikan menggunakan array sebagai dasarnya.
11. Kemampuan untuk Merepresentasikan Konsep Matematis: Array memungkinkan representasi yang lebih baik dari konsep matematis, seperti vektor, matriks, dan larik.

12. Efisien untuk Pencarian dan Pengurutan Sederhana: Jika data dalam array sudah terurut, pencarian dan pengurutan sederhana dapat dilakukan dengan cukup efisien.

CARA DEKLARASI DAN MENGAKSES ARRAY

Untuk mendeklarasi array mirip seperti kalian akan mendeklarasikan variabel. Bedanya adalah array membutuhkan tanda kurung siku (“**[]**”) sebagai inisialisasinya dan menentukan panjangnya. Caranya adalah dengan menentukan tipe data kemudian diikuti nama variabelnya dan menambahkan tanda kurung siku sebagai penanda array. Jangan lupa untuk menentukan jumlah elemen (panjang) yang harus disimpan.

```
int number [5];
```

Kalian juga bisa menambahkan value dari array tersebut, contohnya :

```
int number [5] = {2, 4, 6, 8, 10};
char arr[6] = { 'G', 'e', 'k', 'k', 'o', '\0' };
```

Dari contoh diatas, setiap value akan otomatis memiliki indeks masing-masing dimulai dari 0. Sehingga untuk mengakses array, kita bisa memanggil salah satu atau lebih data menggunakan nomor indeksnya. Berikut adalah contoh untuk memanggil salah satu indeks pada array :

```c
#include <stdio.h>

int main()
{
    // membuar array char
    char arr[6] = { 'G', 'e', 'k', 'k', 'o', '\0' };

    // mengakses salah satu data
    printf("Huruf ke-2 adalah: %c", arr[1]);
    return 0;
}
```

Kode diatas akan mendeklarasikan array dengan panjang 6 indeks. Dari setiap indeks akan diisi oleh hutuf yang sudah di inisialisasikan. Kemudian pada bagian printf, terdapat pemanggilan variabel

arr[1] untuk memanggil data yang berada pada array. Kenapa **arr[1]**? Ingat bahwa array dimulai dari 0 sehingga huruf “e” berada di indeks [1].

Kalian juga bisa menambahkan data pada array sesuai indeks seperti pada contoh berikut :

```
char arr[7];
arr[0] = 'a';
arr[1] = 'b';
arr[2] = 'c';
arr[3] = 'd';
arr[4] = 'e';
arr[5] = 'f';
arr[6] = 'g';
```

Karena array index dimulai dari 0, maka untuk mengisi data pada array bisa diinisialisasikan sesuai indexnya.

Selain itu, dengan menggunakan loop, kita juga bisa mengakses semua data yang berada di dalam array. Loop **for** adalah bentuk loop paling umum yang digunakan untuk mengulangi serangkaian pernyataan berdasarkan jumlah pengulangan yang telah ditentukan. Berikut contoh kode looping for dalam array :

```
#include <stdio.h>

int main() {
    int angka[5] = {10, 20, 30, 40, 50};

    // Looping untuk mencetak setiap elemen dalam array
    for (int i = 0; i < 5; i++) {
        printf("Elemen ke-%d: %d\n", i, angka[i]);
    }

    return 0;
}
```

Dari kode di atas, data dalam array akan diakses menggunakan **for.** Cara pengunaannya adalah dengan membuat variabel baru pada looping for yaitu **“i”**, yang kemudian akan di *increment* **(i++)** sehingga ketika variabel **angka[i]** dipanggil didalam printf, maka variabel **“i”** dalam looping for akan mengakses setiap data dalam array.

Kalian juga bisa menggunakan **loop while, do-while** dalam array. Looping array memungkinkan kita untuk mengakses dan memanipulasi setiap elemen dalam array secara efisien. Dengan loop, kita dapat melakukan banyak operasi pada seluruh atau sebagian elemen dalam array dan memproses data dengan lebih efektif.

- Loop while dalam array

Dari kode di atas, kita mendeklarasikan variabel **i** bernilai 0 alias true. Seperti pada modul sebelumnya, loop while hanya akan berjalan apabila bernilai true. Kemudian pada kode **while(i < 5),** kode akan berjalan selama kondisi **i < 5** terpenuhi, yaitu ketika kondisi bernilai true, maka blok di dalam **while** akan dieksekusi. Namun ketika kondisi bernilai false, maka eksekusi dari **while** akan dihentikan

```c
#include <stdio.h>

int main() {
    int angka[5] = {10, 20, 30, 40, 50};
    int i = 0;

    // Looping untuk mencetak setiap elemen dalam array
       menggunakan while
    while (i < 5) {
        printf("Elemen ke-%d: %d\n", i, angka[i]);
        i++;
    }

    return 0;
}
```

dan program akan melanjutkan ke baris kode setelah blok **while.** Kemudian, pada printf terdapat pemanggilan variabel **angka[i]** yaitu untuk mengakses elemen-elemen dalam array. Jadi, variabel **i** ini dipanggil melalui loop for yang diimplementasikan pada array. Increment **i++** adalah sebagai meningkatkan nilai variabel **i** sehingga loop dapat melanjutkan iterasinya untuk mengakses elemen berikutnya dalam array.

- Loop do-while dalam array

```c
#include <stdio.h>

int main() {
    int angka[5] = {10, 20, 30, 40, 50};
    int i = 0;

    // Looping untuk mencetak setiap elemen dalam array
menggunakan do-while
    do {
        printf("Elemen ke-%d: %d\n", i, angka[i]);
        i++;
    } while (i < 5);

    return 0;
}
```

Pada contoh diatas, kita mendeklarasikan sebuah array yang Bernama **angka** dan didalamnya terdapat beberapa data dengan panjang 5. Kemudian kita mendeklarasikan variabel **i** dengan inisialisasi nilai 0. Variabel ini akan digunakan sebagai indeks untuk mengakses elemen - elemen dalam array. Kemudian, pada loop do-while kode akan selalu dieksekusi sekali sebelum kondisi di evaluasi. Sama seperti loop while, variabel **angka[i]** digunakan untuk mengakses setiap elemen dengan menggunakan increment sebagai meningkatkan nilai variabel supaya dapat melanjutkan iterasinya untuk setiap elemen dalam array.

MENGUBAH DATA DALAM ARRAY

```c
int array[5] = {1,2,3,4,5]; //mendeklarasikan array
array[2] = 6; //mengubah salah satu indeks pada array
```

Selain mengakses data di dalam array, kalian juga bisa mengubah data yang berada di dalam array. Mengubah data dalam array berarti mengganti nilai atau isi dari salah satu atau beberapa elemen dalam array dengan nilai baru, sesuai dengan kebutuhan program. Manipulasi data dalam array sangat penting karena memungkinkan kita untuk melakukan perubahan pada data yang disimpan dalam struktur array seperti meng-*update*, menghitung, atau memanipulasi elemen dalam array. Untuk mengubah data pada elemen tertentu dalam array, kita perlu mengakses elemen tersebut menggunakan indeks dan menetapkan nilai baru ke elemen tersebut.

```c
#include <stdio.h>
int main()
{
    int age[5] = {16,45,67,14,27};
    printf("Before : %d", age[3]);

    age [3] = 23; //mengubah nilai pada indeks ketiga
    printf("After : %d", age[3]);

    return 0;
}
```

Contoh dari kode di atas, di awal kita sudah mendeklarasikan array beserta nilainya. Namun ternyata, kita perlu mengubah salah satu indeks pada array. Lalu, bagaimana cara mengubahnya tanpa harus mendeklarasikan ulang? Caranya adalah dengan memanggil salah satu indeks array yang ingin diganti, contoh **age[3]** yang semula bernilai 14 akan diganti menjadi bernilai 23. Kemudian, setelah memanggilnya, baru kita ganti nilai pada indeks tersebut dan memanggil variabel **age[3]** pada printf yang akan menghasilkan output berbeda. Jadi, dapat disimpulkan bahwa untuk mengganti salah satu indeks pada array, kita bisa memanggil salah satu indeksnya saja dan memberikan nilai baru pada indeks tersebut. Namun, karena **array pada bahasa C bersifat statis**, maka kita tidak dapat mengubah ukuran array atau menambah/mengurangi elemen array secara langsung. Misalnya, pada kode awal, kita tidak bisa menambahkan elemen baru ke array angka atau mengubah ukuran array angka dari 5 menjadi 6 tanpa mendeklarasikan ulang.

ARRAY MULTIDIMENSI

Array multidimensi adalah array yang terdiri dari lebih satu dimensi. Array multidimensi disebut juga dengan array 2D yang diidentifikasi oleh dua indeks, yaitu **indeks baris** dan **indeks kolom**.

	col 1	col 2	col 3
row 1	1	2	3
row 2	4	5	6
row 3	7	8	9

Untuk menggunakan array 2D, kalian harus mendeklarasikan tipe data dan ukuran array dalam setiap dimensi. Inisialisasi array 2D dapat dilakukan secara langsung pada saat deklarasi atau dengan mengisi nilai pada setiap elemen array menggunakan indeks baris dan kolom (matriks).

DEKLARASI ARRAY MULTIDIMENSI

Deklarasi array multidimensi dilakukan dengan menyediakan jumlah dimensi yang diinginkan. Umumnya, array multidimesi memiliki dua dimensi, seperti matriks 2D, tetapi kita juga dapat memiliki tiga dimensi atau lebih untuk representasi data yang lebih kompleks.

Jumlah total elemen yang disimpan dalam array multidimensi dapat dihitung dengan memgalikan ukuran semua dimensi. Misalnya :

- int **arr[20][30]** dapat menyimpan total 600 elemen, karena 20*30 = 600
- int **arr[10][25][35]** dapat menyimpan 8.750 elemen, karena 10*25*35

- **ARRAY 2 DIMENSI**

Array 2 dimensi adalah bentuk array multidimensi paling sederhana.

Berikut contoh deklarasi array 2D dengan ukuran 3x3:

```
int matriks[3][3]
```

```
int matriks[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Selain itu, kalian bisa meninisialisasi pada saat deklarasi atau setelah deklarasi dengan menggunakan tanda kurung kurawal **({ }).** Berikut contoh inisialisasi array 2D :

Array multidimensi memungkinkan kita menyimpan dan memanipulasi data dalam bentuk matriks atau struktur data berbentuk grid. Dengan array multidimensi, kita dapat melakukan berbagai operasi matriks seperti penjumlahan, perkalian, dan operasi lainnya.

Supaya lebih jelas, kalian bisa mencoba source code berikut untuk array 2D :

```
int matriks1[3][3] = {{1, 2, 3},
                      {4, 5, 6},
                      {7, 8, 9}};

int matriks2[3][3] = {{9, 8, 7},
                      {6, 5, 4},
                      {3, 2, 1}};

int hasil[3][3];

// Penjumlahan matriks
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        hasil[i][j] = matriks1[i][j] + matriks2[i][j];
    }
}
```

Kode diatas mendeklarasikan dan menginisialisasi 2 matriks menggunakan array 2D. Pada bagian loop for pertama **(i loop)** digunakan untuk mengiterasi baris dari 0 hingga 2 (indeks baris dari matriks). Loop for kedua **(j loop)** digunakan untuk mengiterasi kolom dari 0 hingga 2 (indeks kolom dari matriks). Pada setiap iterasi, elemen **hasil[i][j]** diisi dengan hasil penjumlahan dari elemen yang bersesuaian di matriks1[i][j] dan matriks2[i][j]. Setelah loop selesai dieksekusi, array hasil akan berisi hasil penjumlahan matriks matriks1 dan matriks2.

Kesimpulannya, array multidimensi adalah struktur data yang berguna untuk merepresentasikan data dalam bentuk matriks atau grid. Dengan array multidimensi, kita dapat menyimpan dan memanipulasi data dalam bentuk 2D, 3D, dan lebih banyak dimensi sesuai kebutuhan. Penggunaan array multidimensi mempermudah operasi pada data berbentuk grid seperti matriks, tabel, atau data dalam beberapa dimensi.

- **ARRAY 3 DIMENSI**

Array 3D adalah struktur data dalam pemrograman yang digunakan untuk menyimpan elemen-elemen dalam tiga dimensi. Ini sering digunakan untuk mempresentasikan data tiga dimensi seperti citra volume medis, data cuaca dalam tiga dimensi, dan sejenisnya. Dalam array 3D, elemen-elemen

diindeks menggunakan tiga indeks, yaitu indeks baris, indeks kolom, dan indeks kedalaman. Berikut contoh untuk mendeklarasikan array 3D :

```
tipe_data nama_array [x][y][z];
```

- x = jumlah array 2D
- y = jumlah baris di setiap array 2D
- z = jumlah kolom di setiap array 2D

Dari keterangan diatas, kenapa ada kaitan dengan 2D? karena array 3D merupakan kumpulan array dua dimensi. Ibaratnya seperti beberapa array 2D ditumpuk di atas satu sama lain.

Cara menginisialisasikannya seperti contoh berikut :

```
int array[2][3][4] =
 {
  { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} },
  { {12,13,14,15}, {16,17,18,19}, {20,21,22,23} }
 };
```

Dari contoh diatas, **int array[2][3][4]** adalah deklarasi array 3D yang memiliki 24 elemen. Pada bagian awal, terdapat deklarasi dan inisialisasi elemen-elemen dalam array. Setiap elemen dalam array diberikan nilai yang disesuaikan dengan indeksnya. Contohnya : **array[0][0][0]** memiliki nilai 0, **array[0][0][1]** memiliki nilai 1, dan seterusnya. Elemen - elemen dalam array diinisialisasi dengan susunan lapisan-baris-kolom. Misalnya, **{0, 1, 2, 3}** adalah elemen pertama pada lapisan pertama, **{4, 5, 6, 7}** adalah elemen kedua pada lapisan pertama, dan seterusnya.

ARRAY DAN POINTER

Dalam bahasa C, terdapat hubungan erat antara array dan pointer. Array dapat dianggap sebagai kumpulan elemen-elemen yang memiliki tipe data yang sama dan diindeks secara berurutan. Pada dasarnya, array dapat diinterpretasikan sebagai pointer ke elemen pertama dari array tersebut.

```c
#include <stdio.h>

int main() {
    int numbers[5] = {10, 20, 30, 40, 50};

    printf("Value of numbers[0]: %d\n", numbers[0]);
    printf("Value of *numbers: %d\n", *numbers);

    int *ptr = numbers;

    printf("Value of *ptr: %d\n", *ptr);
    printf("Value of *(ptr+1): %d\n", *(ptr + 1));
    printf("Value of *(ptr+2): %d\n", *(ptr + 2));

    return 0;
}
```

Perhatikan kode program di atas. Pertama, kita mendeklarasikan sebuah array bernama numbers dengan panjang 5 dan menginisialisasi elemennya dengan nilai {10, 20, 30, 40, 50}. Ini adalah array yang akan digunakan sebagai contoh untuk memahami hubungan antara array dan pointer. Di baris selanjutnya, kita menggunakan perintah printf untuk mencetak nilai dari elemen pertama array numbers menggunakan indeks 0, yaitu numbers[0]. Ini adalah cara umum untuk mengakses elemen array dengan indeks tertentu.

Kemudian, kita menggunakan perintah printf lagi untuk mencetak nilai dari elemen pertama array numbers menggunakan pointer. Di sini, kita menggunakan operator dereference (*) pada numbers, yaitu ***numbers**, yang mengacu pada nilai elemen pertama dari array. Pada langkah ini, kita mendeklarasikan sebuah pointer ptr yang menunjuk pada elemen pertama array numbers. Ini dilakukan dengan menyamakan ptr dengan numbers. Dengan demikian, ptr sekarang menunjuk pada alamat elemen pertama dari array numbers. Setelah pointer ptr dideklarasikan dan ditetapkan, kita menggunakan perintah printf untuk mencetak nilai elemen pertama array menggunakan pointer, yaitu *ptr. Hasilnya akan sama dengan nilai dari numbers[0]. Selanjutnya, kita menggunakan pointer ptr untuk mengakses elemen kedua dan ketiga array numbers dengan cara yang serupa seperti yang dilakukan pada langkah 3 dan 4. *(ptr + 1) mengacu pada elemen kedua array, dan *(ptr + 2) mengacu pada elemen ketiga array.

CODELAB 1

Buatlah program yang berisi daftar nama-nama berikut :

1. neon
2. gekko
3. omen
4. sage
5. jett

Buat array 2D dengan ukuran 5 x 10 untuk menyimpan nama-nama di atas. Kemudian tampilkan daftar nama di atas dan ubah salah satu huruf pada elemen “sage” menjadi “kage”. Tampilkan daftar nama sebelum diubah dan sesudah diubah.

Contoh output :

```
Daftar Agen:
Agent ke-1: neon
Agent ke-2: gekko
Agent ke-3: omen
Agent ke-4: sage
Agent ke-5: jett

Daftar Agen setelah perubahan:
Agent ke-1: neon
Agent ke-2: gekko
Agent ke-3: omen
Agent ke-4: kage
Agent ke-5: jett

Process returned 0   execution time : 0.007 s
Press any key to continue.
```

CODELAB 2

Buatlah program penjumlahan matriks yang mengambil dua matriks 2x2. Untuk setiap nilai pada elemen berasal dari kode program (tidak perlu inputan user). Kemudian, tunjukkan hasil outputnya.

Contoh output :

```
Hasil penjumlahan matriks:
6 8
10 12

Process returned 0   execution time : 0.000 s
Press any key to continue.
```

KEGIATAN 1

Buatlah program yang meminta inputan dari user untuk memasukkan sejumlah bilangan bulat ke dalam array. Kemudian, program akan menampilkan angka-angka ganjil dan genap dalam array sesuai dengan pengelompokannya. Kelompokkan angka-angka ganjil dan genap ke dalam dua array terpisah (**arrayGanjil** dan **arrayGenap**).

Berikut contoh outputnya :

```
Masukkan jumlah elemen yang akan diinput: 10
Input 10 elemen:
65
79
14
57
88
32
66
20
12
47
Bilangan Ganjil dalam array : 65 79 57 47
Bilangan Genap dalam array : 14 88 32 66 20 12

Process returned 0   execution time : 13.183 s
Press any key to continue.
```

KEGIATAN 2

Buatlah program matriks 3x3 dengan opsi **penjumlahan, perkalian,** dan **pengurangan.** Kemudian setiap inputan dari user, ditampilkan dalam bentuk matriks 3x3. Apabila user menginput opsi selain menu maka akan muncul pesan “INPUTAN TIDAK VALID” dan program akan mengulang ke awal. Berikut contoh dari hasil outputnya :

```
=== Operasi Matriks ===
1. Penjumlahan
2. Pengurangan
3. Perkalian
Pilih Salah Satu (1/2/3): 1
Masukkan elemen pada matriks pertama(3x3):
1
2
3
4
5
6
7
8
9
Masukkan elemen pada matriks kedua (3x3):
5
6
7
8
9
1
2
3
4
Matriks Pertama:
1 2 3
4 5 6
7 8 9
Matriks Kedua:
5 6 7
8 9 1
2 3 4
Hasil Penjumlahan:
6 8 10
12 14 7
9 11 13

Process returned 0   execution time : 10.493 s
Press any key to continue.
```

Output Pengurangan :

```
=== Operasi Matriks ===
1. Penjumlahan
2. Pengurangan
3. Perkalian
Pilih Salah Satu (1/2/3): 2
Masukkan elemen pada matriks pertama(3x3):
1
2
3
4
5
6
7
8
9
Masukkan elemen pada matriks kedua (3x3):
5
6
7
8
9
1
2
3
4
Matriks Pertama:
1 2 3
4 5 6
7 8 9
Matriks Kedua:
5 6 7
8 9 1
2 3 4
Hasil Pengurangan:
-4 -4 -4
-4 -4 5
5 5 5

Process returned 0   execution time : 11.452 s
Press any key to continue.
```

Output Perkalian :

```
=== Operasi Matriks ===
1. Penjumlahan
2. Pengurangan
3. Perkalian
Pilih Salah Satu (1/2/3): 3
Masukkan elemen pada matriks pertama(3x3):
1
2
3
4
5
6
7
8
9
Masukkan elemen pada matriks kedua (3x3):
5
6
7
8
9
1
2
3
4
Matriks Pertama:
1 2 3
4 5 6
7 8 9
Matriks Kedua:
5 6 7
8 9 1
2 3 4
Hasil Perkalian:
27 33 21
72 87 57
117 141 93

Process returned 0   execution time : 12.877 s
Press any key to continue.
```

KRITERIA & DETAIL PENILAIAN

Kriteria	Poin
Codelab 1	10
Codelab 2	10
Kegiatan 1	25
Kegiatan 2	30
Pemahaman	15
Ketepatan Menjawab	10