VERSI 1.0

AGUSTUS, 2024

[PEMROGRAMAN DASAR]

MODUL 5 - ARRAY

DISUSUN OLEH:

WEMPY ADITYA WIRYAWAN

MUHAMMAD ZAKY DARAJAT

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

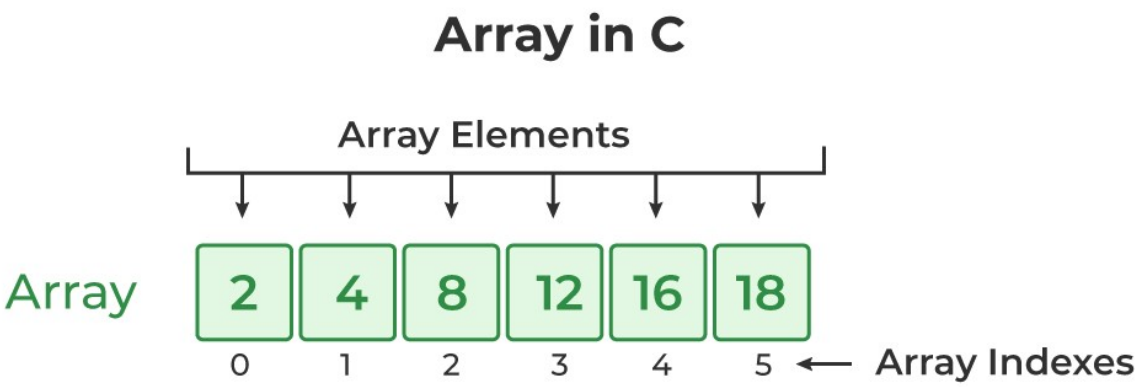
- Mampu mendeklarasikan, menginisialisasi, dan mengakses elemen array, serta memahami penggunaan array dalam berbagai kasus pemrograman.

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.

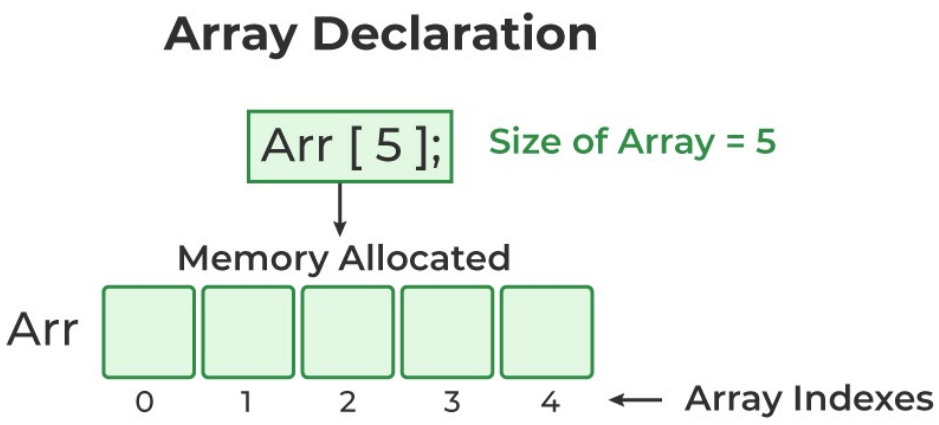
Jadi array digunakan untuk menyimpan elemen/data secara kontinu, satu demi satu seperti nampan telur dalam menyimpan telur. Index pada array membantu kalian untuk melakukan berbagai operasi pada array seperti mengakses elemen, memodifikasi array,

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 :

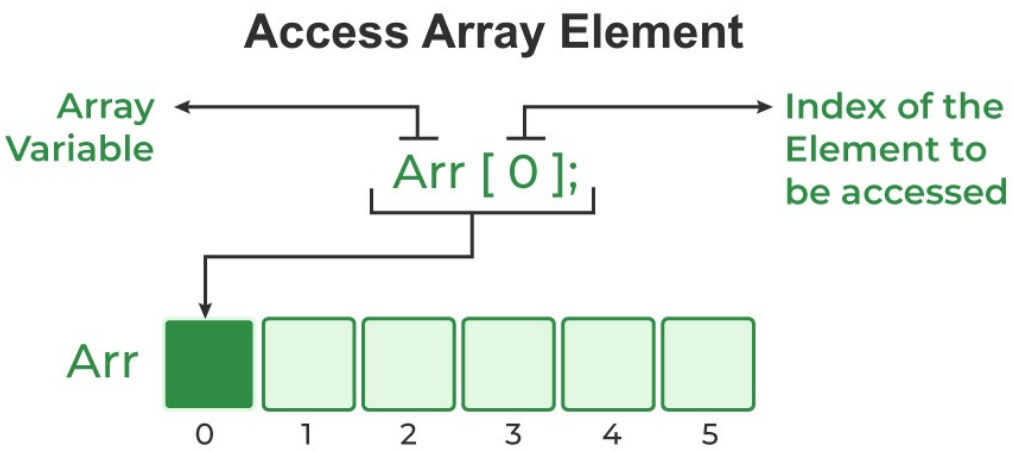
```
int number [5] = {2, 4, 6, 8, 10};
char arr[6] = { 'G', 'e', 'k', 'k', 'o', '\0' };
```

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;
}
```

```
Huruf ke-2 adalah: e

=== Code Execution Successful ===
```

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;
}
```

```
Elemen ke-0: 10
Elemen ke-1: 20
Elemen ke-2: 30
Elemen ke-3: 40
Elemen ke-4: 58


=== Code Execution Successful ===
```

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;
}
```

```
Elemen ke-0: 10
Elemen ke-1: 20
Elemen ke-2: 30
Elemen ke-3: 40
Elemen ke-4: 50


=== Code Execution Successful ===
```

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;
}
```

```
Elemen ke-0: 10
Elemen ke-1: 20
Elemen ke-2: 30
Elemen ke-3: 40
Elemen ke-4: 50


=== Code Execution Successful ===
```

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.

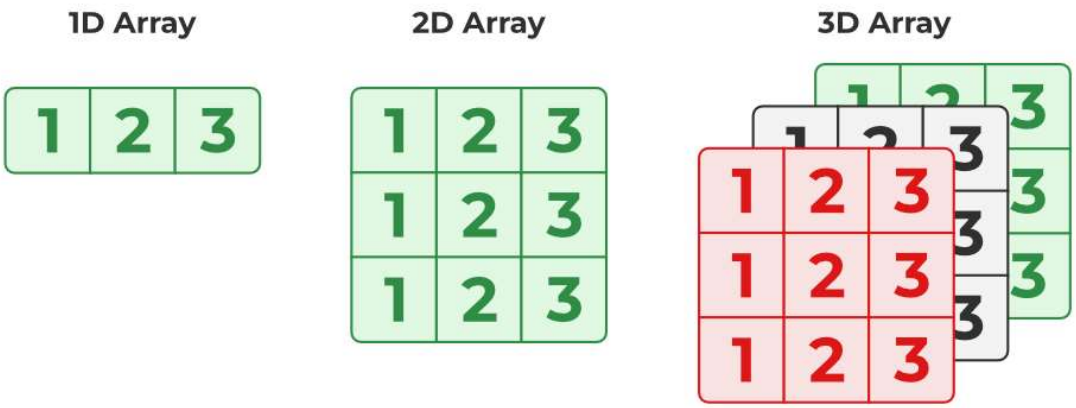
```c
#include <stdio.h>
int main()
{
    int age[5] = {16,45,67,14,27};
    printf("Before : %d", age[3]);

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

    return 0;
}
```

```
Before: 14
After: 23


=== Code Execution Successful ===
```

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 di C bisa berdimensi tunggal seperti array 1D atau multidimensi seperti array 2D, array 3D, dan seterusnya. Itu dapat memiliki sejumlah dimensi. Jumlah elemen dalam array multidimensi adalah hasil kali ukuran semua dimensi.



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

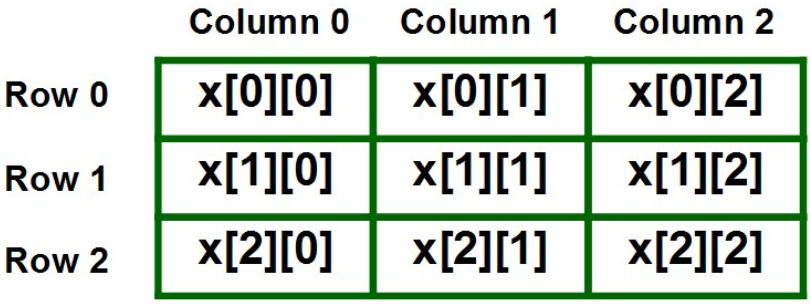
Deklarasi array multidimensi dilakukan dengan menyediakan jumlah dimensi yang diinginkan. Umumnya, array multidimesi memiliki dua dimensi, seperti matriks 2D, tetapi kita juga dapat memiliki tiga dimensi atau lebih untuk representasi data yang lebih kompleks.

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.



Deklarasi Array Dua Dimensi dalam C

Bentuk dasar untuk mendeklarasikan array 2D dengan x baris dan y kolom dalam C adalah sebagai berikut:

```
data_type array_name[x][y];
```

di mana:

- data_type: Tipe data yang akan disimpan di setiap elemen.
- array_name: Nama array.
- x: Jumlah baris.
- y: Jumlah kolom.

Inisialisasi Array Dua Dimensi dalam C

Berbagai cara untuk menginisialisasi array 2D adalah sebagai berikut:

1. **Inisialisasi Array 2D Menggunakan Daftar Inisialisasi**

Kita dapat menginisialisasi array 2D dalam C dengan menggunakan daftar inisialisasi seperti yang ditunjukkan dalam contoh berikut.

```
int x[3][4] = {{0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11}};
```

Jenis inisialisasi ini menggunakan kurung kurawal bersarang. Setiap set kurung kurawal bagian dalam mewakili satu baris. Dalam contoh di atas, ada total tiga baris sehingga ada tiga set kurung kurawal bagian dalam. Keuntungan dari metode ini adalah lebih mudah dipahami.

Catatan: Jumlah elemen dalam daftar inisialisasi harus selalu kurang dari atau sama dengan total jumlah elemen dalam array.

Kita juga dapat mendeklarasikan array tanpa mendefinisikan ukuran baris jika kita menggunakan inisialisasi daftar. Namun, mendefinisikan jumlah kolom tetap wajib. Kompiler akan secara otomatis menentukan ukuran array dalam kasus ini:

```
data_type array_name[][y] = {...};
```

Inisialisasi Array 2D Menggunakan Loop

Kita dapat menggunakan loop apa pun dalam bahasa C untuk menginisialisasi setiap elemen dari array 2D satu per satu seperti yang ditunjukkan dalam contoh di bawah ini.

```
int x[3][4];

for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 4; j++) {
        x[i][j] = i + j;
    }
}
```

Mengakses Elemen Array Dua Dimensi dalam C

Elemen-elemen dalam array 2D diakses menggunakan indeks baris dan indeks kolom. Setiap elemen dalam array 2D dapat dirujuk dengan:

Sintaks:

```
array_name[i][j]
```

di mana:

- i: Indeks baris.
- j: Indeks kolom.

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];
    }
}
```

```
Hasil penjumlahan matriks:
10 10 10
10 10 10
10 10 10


=== Code Execution Successful ===
```

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.

```c
#include <stdio.h>
void main ()
{
    int arr[3][3],i,j;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            printf("Enter a[%d][%d]: ",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("\n printing the elements ....\n");
    for(i=0;i<3;i++)
    {
        printf("\n");
        for (j=0;j<3;j++)
        {
            printf("%d\t",arr[i][j]);
        }
    }
}
```

Output:

```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34

Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78

printing the elements ....

56      10      30
34      21      34
45      56      78
```

Program di atas adalah contoh penggunaan array dua dimensi dalam bahasa C untuk menerima input dari pengguna dan kemudian mencetak matriks yang dihasilkan. Array `arr[3][3]` dideklarasikan untuk menyimpan matriks 3x3. Program terdiri dari dua bagian utama: bagian pertama adalah pengisian elemen-elemen matriks dengan nilai-nilai yang dimasukkan oleh pengguna, dan bagian kedua adalah pencetakan elemen-elemen matriks tersebut. Dalam bagian pengisian, dua loop `for` bersarang digunakan, di mana loop luar `for (i=0;i<3;i++)` mengiterasi baris, dan loop dalam `for (j=0;j<3;j++)`

mengiterasi kolom. Pada setiap iterasi, pengguna diminta memasukkan nilai untuk elemen matriks yang sesuai melalui `scanf("%d",&arr[i][j])`. Setelah semua elemen diisi, program mencetak elemen-elemen tersebut dalam format matriks dengan dua loop bersarang yang sama, tetapi kali ini untuk menampilkan setiap elemen menggunakan `printf("%d\t",arr[i][j])`. Baris baru dicetak setelah setiap baris matriks selesai ditampilkan. Program ini memungkinkan pengguna untuk melihat matriks yang mereka masukkan dalam bentuk yang terstruktur.

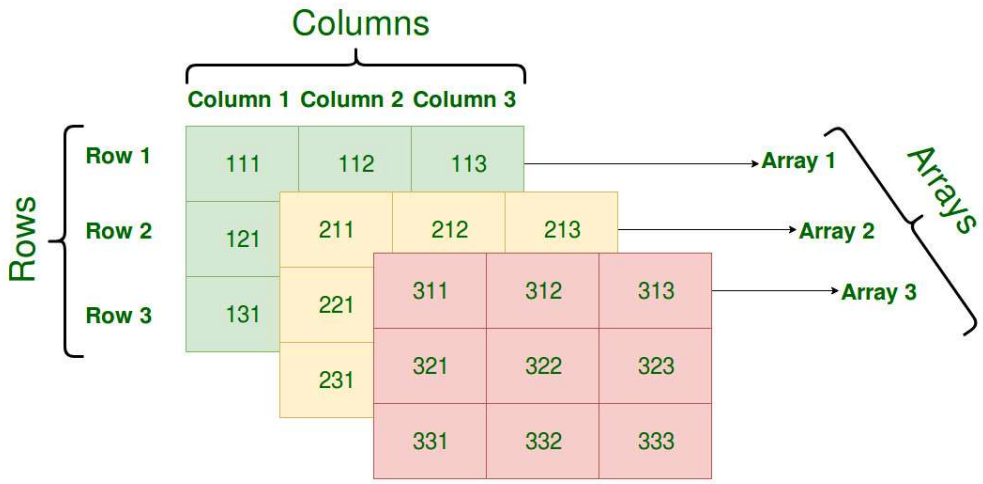
Elemen-elemen dari array 2D harus disimpan berurutan dalam memori. Karena komputer memiliki alamat memori yang linier, array 2D harus diubah menjadi bentuk linier agar bisa disimpan dengan benar. Ada dua cara untuk mengubah array 2D menjadi bentuk linier:

Row-major Order - Dalam metode ini, baris pertama dari array disimpan terlebih dahulu, kemudian baris kedua, lalu baris ketiga, dan seterusnya. Artinya, elemen-elemen disimpan berdasarkan baris. Jadi, semua elemen dalam satu baris disimpan sebelum beralih ke baris berikutnya.

Column-major Order - Dalam metode ini, kolom pertama dari array disimpan terlebih dahulu, kemudian kolom kedua, lalu kolom ketiga, dan seterusnya. Artinya, elemen-elemen disimpan berdasarkan kolom. Jadi, semua elemen dalam satu kolom disimpan sebelum beralih ke kolom berikutnya.

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 :

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.

Inisialisasi Array 3D menggunakan Loop

Ini juga mirip dengan array 2D dengan satu loop bersarang tambahan untuk mengakses satu dimensi tambahan.

```
int x[2][3][4];

for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        for (int k = 0; k < 4; k++) {
            x[i][j][k] = (some_value);
        }
    }
}
```

Contoh program:

```
#include <stdio.h>

int main(void)
{
    // initializing the 3-dimensional array
    int x[2][3][2] = { { { 0, 1 }, { 2, 3 }, { 4, 5 } },
                       { { 6, 7 }, { 8, 9 }, { 10, 11 } } };
```

```
    // output each element's value
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            for (int k = 0; k < 2; ++k) {
                printf("Element at x[%i][%i][%i] = %d\n", i,
                       j, k, x[i][j][k]);
            }
        }
    }
    return (0);
}
```

Output program:

```
Element at x[0][0][0] = 0
Element at x[0][0][1] = 1
Element at x[0][1][0] = 2
Element at x[0][1][1] = 3
Element at x[0][2][0] = 4
Element at x[0][2][1] = 5
Element at x[1][0][0] = 6
Element at x[1][0][1] = 7
Element at x[1][1][0] = 8
Element at x[1][1][1] = 9
Element at x[1][2][0] = 10
Element at x[1][2][1] = 11


=== Code Execution Successful ===
```

Penjelasan Output:

- Program mencetak setiap elemen dari array x dengan format yang menunjukkan indeks array dan nilai elemen.
- Indeks pertama (i) menunjukkan array 2D keberapa yang diakses (dari 0 hingga 1).
- Indeks kedua (j) menunjukkan baris keberapa dalam array 2D (dari 0 hingga 2).
- Indeks ketiga (k) menunjukkan kolom keberapa dalam baris (dari 0 hingga 1).
- Nilai elemen adalah angka yang diinisialisasi dalam array x.

Dengan cara ini, program dapat mengakses dan mencetak setiap elemen dari array tiga-dimensi secara sistematis menggunakan loop bersarang.

Contoh lain penggunaan array 3D:

```c
#include <stdio.h>

int main() {

    int temperatures[3][7] = {
        {30, 32, 31, 29, 28, 30, 31}, // Kota 1
        {25, 26, 27, 24, 25, 26, 28}, // Kota 2
        {20, 22, 21, 19, 20, 22, 23}  // Kota 3
    };

    float average[3] = {0};
    int numCities = 3;
    int numDays = 7;

    for (int city = 0; city < numCities; city++) {
        int sum = 0;
        for (int day = 0; day < numDays; day++) {
            sum += temperatures[city][day];
        }
        average[city] = (float)sum / numDays;
    }

    for (int city = 0; city < numCities; city++) {
        printf("Rata-rata suhu Kota %d: %.2f\n", city + 1, average[city]);
    }

    return 0;
}
```

Output:

```
Rata-rata suhu Kota 1: 30.14
Rata-rata suhu Kota 2: 25.86
Rata-rata suhu Kota 3: 21.00
```

Program ini menghitung rata-rata suhu harian selama satu minggu untuk tiga kota menggunakan array dua dimensi dalam bahasa C. Pertama, program mendeklarasikan dan menginisialisasi array `temperatures[3][7]`, yang menyimpan suhu harian untuk tiga kota. Setiap baris mewakili satu kota, dan setiap kolom mewakili suhu pada hari tertentu. Program kemudian menggunakan loop bersarang untuk menghitung rata-rata suhu setiap kota. Loop luar iterasi melalui kota-kota, sedangkan loop dalam iterasi melalui hari-hari dalam seminggu. Dalam setiap iterasi, suhu harian dijumlahkan, dan rata-rata dihitung dengan membagi jumlah total dengan jumlah hari (7). Rata-rata suhu setiap kota disimpan dalam array `average`, yang kemudian dicetak dengan format dua angka di belakang koma. Program ini menyediakan cara yang efisien untuk mengelola dan menganalisis data suhu untuk beberapa lokasi.

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;
}
```

```
Value of numbers[0]: 10
Value of numbers[1]: 20
Value of *ptr: 10
Value of *(ptr + 1): 20
Value of *(ptr + 2): 30


=== Code Execution Successful ===
```

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 pertamaarray 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.

Contoh lain penerapan konsep pointer dengan array agar lebih memudahkan dalam memahaminya sebagai berikut:

```c
#include <stdio.h>

int main() {
    // Deklarasi dan inisialisasi array
    int numbers[] = {25, 47, 3, 19, 8, 18};
    int length = sizeof(numbers) / sizeof(numbers[0]); // Mendapatkan panjang array
    int *ptr; // Deklarasi pointer
    int max;

    // Menunjuk ke elemen pertama dari array
    ptr = numbers;

    // Inisialisasi nilai maksimum dengan elemen pertama
    max = *ptr;

    // Loop melalui array menggunakan pointer
    for (int i = 1; i < length; i++) {
        if (*(ptr + i) > max) {
            max = *(ptr + i); // Memperbarui nilai maksimum
        }
    }

    printf("The maximum value in the array is: %d\n", max);

    return 0;
}
```

Output:

```
The maximum value in the array is: 47
```

Program ini mencari nilai maksimum dalam array numbers yang berisi enam bilangan bulat. Array diinisialisasi dengan nilai {25, 47, 3, 19, 8, 18}. Variabel max digunakan untuk menyimpan nilai maksimum yang ditemukan dalam array. Pointer ptr digunakan untuk menunjuk ke elemen pertama dari array, dan panjang array dihitung dengan sizeof(numbers) / sizeof(numbers[0]).

Program memulai dengan menginisialisasi max dengan nilai elemen pertama dari array menggunakan pointer *ptr. Kemudian, program menggunakan loop for untuk menjelajahi array, dimulai dari indeks 1 hingga indeks terakhir. Dalam setiap iterasi, program membandingkan nilai elemen saat ini *(ptr + i) dengan max. Jika nilai elemen saat ini lebih besar dari max, maka max diperbarui dengan nilai tersebut.

Setelah loop selesai, program mencetak nilai maksimum yang ditemukan dalam array.

CODELAB 1

Deskripsi: Buatlah program untuk menyimpan dan memanipulasi informasi nilai ujian siswa dalam array 2D. Program juga harus mampu memperbarui nilai ujian dari salah satu siswa dan menampilkan daftar nilai sebelum dan setelah pembaruan.

Daftar Siswa dan Nilai Ujian:

1. Alice: 85
2. Bob: 90
3. Charlie: 78
4. David: 92
5. Eva: 88

Tugas:

- Simpan nama-nama siswa beserta nilai ujian mereka menggunakan array 2D.
- Tampilkan daftar nama dan nilai sebelum diubah.
- Ubah nilai ujian dari siswa "David" menjadi 95.
- Tampilkan daftar nama dan nilai setelah diubah.

Output program yang diharapkan:

```
Daftar siswa dan nilai sebelum diubah:
Alice: 85
Bob: 90
Charlie: 78
David: 92
Eva: 88

Daftar siswa dan nilai setelah diubah:
Alice: 85
Bob: 90
Charlie: 78
David: 95
Eva: 88


=== Code Execution Successful ===
```

CODELAB 2

Deskripsi: Buatlah program untuk menghitung dan menampilkan total pendapatan dari penjualan produk dalam dua toko yang berbeda. Pengguna menentukan jumlah produk yang terjual di setiap toko, dan program akan menjumlahkan total penjualan dan menampilkan hasilnya.

Jumlah Produk yang Terjual di Setiap Toko:

1. Toko 1: [5, 8, 6] (Produk A, Produk B, Produk C)
2. Toko 2: [3, 7, 9] (Produk A, Produk B, Produk C)

Tugas:

- Simpan jumlah produk yang terjual di dua toko menggunakan array 2D.
- Hitung total penjualan untuk setiap produk.
- Tampilkan hasil total penjualan dari kedua toko.

Output program yang diharapkan:

```
Total penjualan untuk setiap produk:
Produk A: 8
Produk B: 15
Produk C: 15


=== Code Execution Successful ===
```

KEGIATAN 1

Deskripsi: Buatlah program yang meminta inputan dari pengguna untuk memasukkan sejumlah nama buah ke dalam array. Kemudian, program akan menampilkan buah-buah tersebut dalam urutan yang diinputkan dan juga dalam urutan abjad.

Tugas:

1. Minta pengguna untuk memasukkan sejumlah nama buah.
2. Simpan nama-nama buah tersebut dalam array.
3. Tampilkan buah-buah tersebut dalam urutan yang diinputkan.
4. Urutkan nama buah dalam array secara abjad.
5. Tampilkan buah-buah tersebut dalam urutan abjad.

Contoh output yang diharapkan:

```
Masukkan jumlah buah (maksimal 10): 4
Masukkan nama buah ke-1: mangga
Masukkan nama buah ke-2: apel
Masukkan nama buah ke-3: pisang
Masukkan nama buah ke-4: jeruk

Nama buah dalam urutan yang diinputkan:
mangga
apel
pisang
jeruk

Nama buah dalam urutan abjad:
apel
jeruk
mangga
pisang


=== Code Execution Successful ===
```

KEGIATAN 2

Deskripsi: Buatlah program yang meminta inputan dari pengguna untuk memasukkan data nilai ujian dari beberapa mahasiswa untuk tiga mata kuliah yang berbeda. Program kemudian akan menghitung rata-rata nilai untuk setiap mata kuliah dan rata-rata nilai keseluruhan, serta menampilkan data tersebut dalam bentuk tabel.

Tugas:

1. Minta pengguna untuk memasukkan jumlah mahasiswa.
2. Simpan data nilai ujian untuk tiga mata kuliah (misalnya, Matematika, Fisika, Kimia) untuk setiap mahasiswa dalam array multidimensi.
3. Hitung rata-rata nilai untuk setiap mata kuliah.
4. Hitung rata-rata nilai keseluruhan.
5. Tampilkan data nilai ujian dan rata-rata dalam bentuk tabel.

Contoh output program yang diharapkan:

```
Masukkan jumlah mahasiswa: 4
Masukkan nilai untuk mahasiswa ke-1 (Matematika, Fisika, Kimia):
80 90 100
Masukkan nilai untuk mahasiswa ke-2 (Matematika, Fisika, Kimia):
70 90 90
Masukkan nilai untuk mahasiswa ke-3 (Matematika, Fisika, Kimia):
60 80 80
Masukkan nilai untuk mahasiswa ke-4 (Matematika, Fisika, Kimia):
80 90 95

Data Nilai Ujian:
Mahasiswa   Matematika  Fisika      Kimia
Mhs 1       80.00       90.00       100.00
Mhs 2       70.00       90.00       90.00
Mhs 3       60.00       80.00       80.00
Mhs 4       80.00       90.00       95.00

Rata-rata Nilai:
Matematika: 72.50
Fisika: 87.50
Kimia: 91.25
Rata-rata Keseluruhan: 83.75


=== Code Execution Successful ===
```

KRITERIA & DETAIL PENILAIAN

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