

## **LAPORAN ARRAY & POINTER**



Oleh:

Nama : L HAFIDL ALKHAIR  
NIM : 2023903430060  
Kelas : TRKJ 1.C  
Jurusan : Teknologi Informasi dan Komputer  
Program Studi : Teknologi Rekayasa Komputer Jaringan  
Dosen Pembimbing : Indrawati, SST. MT



**JURUSAN TEKNOLOGI, KOMPUTER, DAN INFORMASI  
PRODI TEKNOLOGI REKAYASA KOMPUTER DAN JARINGAN  
POLITEKNIK NEGERI LHOKSEUMAWE  
TAHUN AJARAN 2023/2024**

### **LEMBAR PENGESAHAN**

No. Praktikum : 05 /TIK/TRKJ-1C/ Data Structure And Algorithms Practice

Judul : Laporan Array dan Pointer

Nama : L HAFIDL ALKHAIR

NIM : 2023903430060

Kelas : TRKJ-1C

Jurusan : Teknologi Informasi Dan Komputer

Prodi : Teknologi Rekayasa Komputer Jaringan

Tanggal Praktikum : 11 Oktober 2023

Tanggal Penyerahan : 16 Oktober 2023

Buketrata, 16 Oktober 2023

Dosen Pembimbing,

Indrawati, SST.MT

Nip. 19740815 200112 2 001

## A. TUJUAN

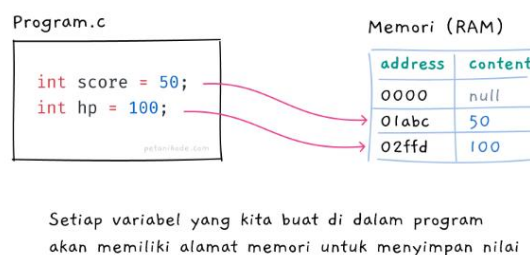
- ❖ Pengelolaan Memori yang Efisien
- ❖ Manipulasi Data yang Fleksibel
- ❖ Pemahaman yang Mendalam tentang Array dan String
- ❖ Penggunaan Fungsi yang Lebih Fleksibel
- ❖ Implementasi Struktur Data Dinamis
- ❖ Memahami konsep dasar pemrograman seperti variable, pengendalian alur, tipe data.
- ❖ Dengan menggunakan fungsi, anda dapat mengoptimalkan kinerja program.

## B. DASAR TEORI

Apa itu Pointer?

Setiap variabel yang kita buat pada program akan memiliki alamat memori. Alamat memori berfungsi untuk menentukan lokasi penyimpanan data pada memori (RAM). Kadang alamat memori ini disebut reference atau referensi.

Coba perhatikan gambar ini:



Pada gambar tersebut memiliki 2 variable “score” dan “hp” . Setiap variable yang kita buat didalam program akan memiliki Alamat memori untuk menyimpan nilai.

Pengertian Fungsi:

Fungsi adalah blok kode yang dapat dipanggil oleh program untuk menjalankan tugas tertentu. Fungsi memiliki nama, tipe kembalian, dan daftar parameter (jika ada). Mereka dapat digunakan untuk mengelompokkan dan mengatur kode, serta memungkinkan pemrogram untuk menjalankan tugas tertentu berulang kali.

Contoh sederhana fungsi dalam Bahasa C:

```
1 int tambah(int a, int b) {  
2     return a + b;  
3 }  
.
```

Fungsi `tambah` ini memiliki dua parameter `a` dan `b` serta mengembalikan hasil penjumlahan keduanya.

### C. ALAT DAN BAHAN

Di dalam pembuatan sebuah Bahasa pemrograman tentu saja memerlukan alat dan bahan-bahan supaya program tersebut dapat di jalankan dengan baik dan tidak mengalami yang namanya error saat program atau kode tersebut di jalankan. Adapun alat dan bahan sebagai berikut:

1. Laptop atau device



2. Tools atau aplikasi pemrograman (disini kita memakai aplikasi Dev-C++)



## D. PROGRAM ARRAY

### 1. Program pertama ( Program Pointer )

#### Source code

```
#include <stdio.h>
#include <conio.h>

int main(){
    int nilai [10];
    int indeks;

    nilai[0]=197;
    nilai[2]=-100;
    nilai[5]=350;
    nilai[3]=nilai[0] +nilai[5];
    nilai[9]=nilai [5]/10;
    --nilai [2];

    for (indeks=0; indeks<10; ++indeks)
        printf("nilai [%d] = %d\n", indeks, nilai[indeks]);

    getch();
}
```

## Hasil Ouput

```
nilai [0] = 197
nilai [1] = 0
nilai [2] = -101
nilai [3] = 547
nilai [4] = 0
nilai [5] = 350
nilai [6] = 42
nilai [7] = 0
nilai [8] = 0
nilai [9] = 35
-----
```

## Analisa :

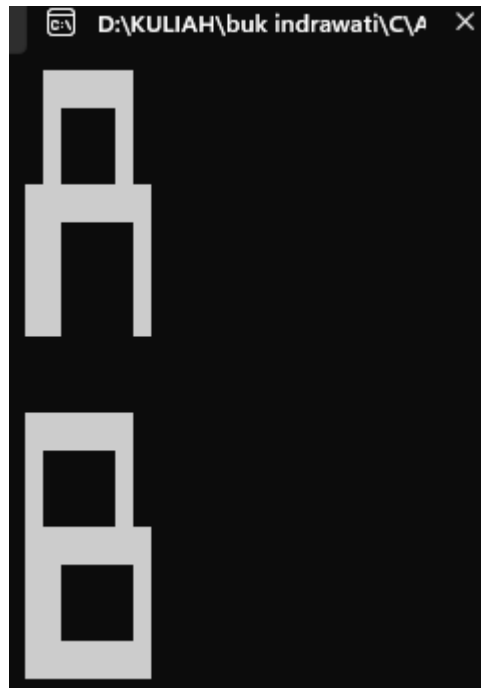
- Program mendeklarasikan array nilai dengan 10 elemen bertipe integer.
- Beberapa elemen dalam array diinisialisasi dengan nilai tertentu: nilai[0] dengan 197, nilai[2] dengan -100, dan nilai[5] dengan 350.
- nilai[3] diisi dengan hasil penjumlahan dari nilai[0] dan nilai[5] ( $197 + 350 = 547$ ).
- nilai[9] diisi dengan hasil pembagian nilai[5] dengan 10 ( $350 / 10 = 35$ ).
- Elemen nilai[2] dikurangi satu (nilai awal -100 menjadi -101).
- Program menggunakan loop for untuk mengakses dan mencetak nilai setiap elemen dalam array nilai.
- Setiap nilai elemen array bersama dengan indeksnya dicetak menggunakan fungsi printf.
- Setelah loop selesai dijalankan, program menunggu pengguna menekan tombol apapun sebelum menutup jendela konsol menggunakan fungsi getch() dari pustaka <conio.h>.
- Program ini menginisialisasi, melakukan operasi, dan mencetak nilai-nilai dalam sebuah array menggunakan loop for. Program ini memberikan gambaran tentang cara menggunakan array dan loop dalam bahasa pemrograman C.

## 2. Program kedua ( Array 2)

### Source code

```
3  int main()
4  {
5      int i, j, k;
6      static int data_huruf[2][8][8] = {
7          {
8              {0, 1, 1, 1, 1, 1, 0, 0},
9              {0, 1, 0, 0, 0, 1, 0, 0},
10             {0, 1, 0, 0, 0, 1, 0, 0},
11             {1, 1, 1, 1, 1, 1, 1, 0},
12             {1, 1, 0, 0, 0, 0, 1, 0},
13             {1, 1, 0, 0, 0, 0, 1, 0},
14             {1, 1, 0, 0, 0, 0, 1, 0},
15             {0, 0, 0, 0, 0, 0, 0, 0}
16         },
17         {
18             {1, 1, 1, 1, 1, 1, 0, 0},
19             {1, 0, 0, 0, 0, 1, 0, 0},
20             {1, 0, 0, 0, 0, 1, 0, 0},
21             {1, 1, 1, 1, 1, 1, 1, 0},
22             {1, 1, 0, 0, 0, 0, 1, 0},
23             {1, 1, 0, 0, 0, 0, 1, 0},
24             {1, 1, 1, 1, 1, 1, 1, 0},
25             {0, 0, 0, 0, 0, 0, 0, 0}
26         }
27     };
28
29     for (i = 0; i < 2; i++) {
30         for (j = 0; j < 8; j++) {
31             for (k = 0; k < 8; k++) {
32                 if (data_huruf[i][j][k])
33                     putchar('\xDB');
34                 else
35                     putchar(' ');
36             }
37             puts("");
38         }
39         puts("");
40     }
41     getchar();
42 }
43
```

Hasil output



Analisa Program :

- i, j, dan k digunakan sebagai variabel iterasi untuk loop.
- data\_huruf adalah array tiga dimensi yang menyimpan dua pola huruf. Dimensi pertama (indeks 0) adalah huruf "L", dan dimensi kedua (indeks 1) adalah huruf "U".
- Array data\_huruf diinisialisasi dengan dua matriks 8x8 yang merepresentasikan pola huruf "L" dan "U".
- Program menggunakan tiga tingkat nested loops untuk mengiterasi melalui setiap elemen dalam array tiga dimensi.
- Variabel i digunakan untuk mengakses dimensi pertama (huruf "L" atau "U").
- Variabel j digunakan untuk mengakses baris dalam matriks (0 hingga 7).
- Variabel k digunakan untuk mengakses kolom dalam matriks (0 hingga 7).



- Setiap elemen dalam matriks diuji. Jika elemen tersebut aktif (nilai 1), program mencetak karakter '■'; jika tidak (nilai 0), mencetak spasi.
- Setelah mencetak satu baris matriks, program menggunakan puts("") untuk pindah ke baris berikutnya dalam output.
- Program menggunakan getch() untuk menunggu masukan pengguna sebelum program selesai berjalan.

### 3. Program ketiga ( Array 3)

#### Source code

```

1  #include <stdio.h>
2
3  int findmax(int[], int);
4
5  int main() {
6      static int data1[] = {5, 34, 56, -12, 3, 19};
7      static int data2[] = {1, -2, 34, 207, 93, -12};
8
9      printf("Nilai maksimum dari data1[] adalah %d\n", findmax(data1, 6));
10     printf("Nilai maksimum dari data2[] adalah %d\n", findmax(data2, 6));
11
12     return 0;
13 }
14
15 int findmax(int nilai[], int jml_data) {
16     int terbesar = nilai[0];
17     for (int i = 1; i < jml_data; ++i) {
18         if (nilai[i] > terbesar) {
19             terbesar = nilai[i];
20         }
21     }
22     return terbesar;
23 }

```

#### Hasil output

```

Nilai maksimum dari data1[] adalah 56
Nilai maksimum dari data2[] adalah 207
-----

```

Analisa program :

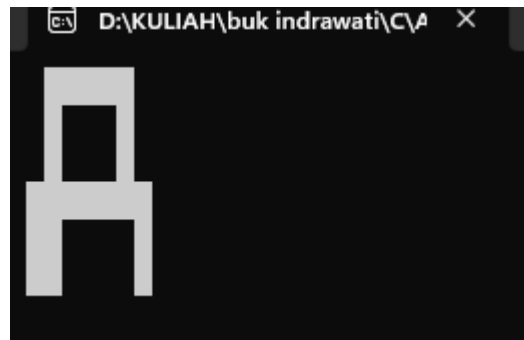
- Fungsi `int findmax(int nilai[], int jml_data)` didefinisikan untuk mencari nilai maksimum dari sebuah array.
- Fungsi ini menerima dua parameter: array integer (`nilai[]`) dan jumlah elemen dalam array (`jml_data`).
- Mengembalikan nilai integer yang merupakan nilai maksimum dalam array.
- Dua array statis, `data1` dan `data2`, didefinisikan dan diinisialisasi dengan nilai-nilai integer.
- Fungsi `findmax()` dipanggil dua kali dengan dua array yang berbeda untuk mencari nilai maksimum dari masing-masing array.
- Hasilnya dicetak ke layar menggunakan `printf()`.
- Program dimulai dengan fungsi `main()`.
- Variabel `i`, `j`, dan `k` digunakan sebagai variabel iterasi.
- Fungsi `findmax()` diimplementasikan di bawah `main()`.
- Program mengembalikan nilai 0 (`return 0;`) untuk menunjukkan bahwa program berjalan dengan sukses.
- Fungsi ini menggunakan pendekatan pencarian linear untuk menemukan nilai maksimum dalam array.
- Dimulai dengan menginisialisasi variabel terbesar dengan nilai pertama dalam array.
- Melalui loop `for`, fungsi membandingkan nilai setiap elemen dengan terbesar. Jika nilai elemen lebih besar, terbesar diperbarui.
- Setelah loop selesai, nilai terbesar dikembalikan.

#### 4. Program keempat ( Array 4)

Source code

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  int main() {
5      int i, j;
6
7      static int A[8][8] = {
8          {0, 1, 1, 1, 1, 1, 0, 0},
9          {0, 1, 0, 0, 0, 1, 0, 0},
10         {0, 1, 0, 0, 0, 1, 0, 0},
11         {1, 1, 1, 1, 1, 1, 1, 0},
12         {1, 1, 0, 0, 0, 0, 1, 0},
13         {1, 1, 0, 0, 0, 0, 1, 0},
14         {0, 0, 0, 0, 0, 0, 0, 0}
15     };
16
17     for (i = 0; i < 7; i++) {
18         for (j = 0; j < 8; j++) {
19             if (A[i][j]) {
20                 putchar('\xDB');
21             } else {
22                 putchar(' ');
23             }
24         }
25         puts("");
26     }
27     getch();
28 }
```

Hasil output



Analisa Program :

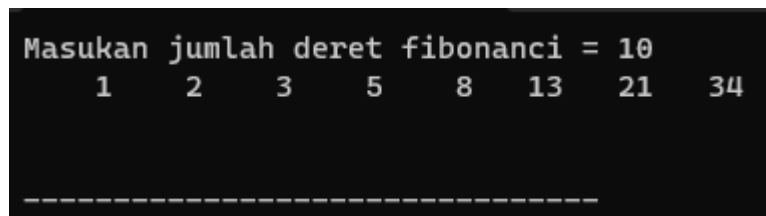
- Program mencetak matriks 8x8 ke layar terminal.
- Karakter 'A' (kode ASCII 219) dicetak untuk elemen matriks yang bernilai 1, dan spasi dicetak untuk elemen matriks yang bernilai 0.
- Program menggunakan dua loop bersarang: loop pertama mengiterasi baris, dan loop kedua mengiterasi kolom dalam matriks.
- Program menggunakan puts("") untuk pindah ke baris berikutnya setelah mencetak satu baris matriks.
- Program akan menunggu hingga pengguna menekan tombol apapun setelah menampilkan matriks.

## 5. Program kelima (array 5 )

### Source code

```
1  #include <stdio.h>
2
3  int main(){
4      int fibo[100],i , jumlah;
5
6      fibo[0]= 0;
7      fibo[1]= 1;
8
9      printf("Masukan jumlah deret fibonanci = ");
10     scanf("%d", &jumlah);
11
12     for(i=2; i< jumlah ; ++i){
13         fibo[i] = fibo[i-1] + fibo[i-2];
14         printf("%5d", fibo[i]);
15     }
16     printf("\n\n");
17 }
```

### Hasil ouput



```
Masukan jumlah deret fibonanci = 10
      1      2      3      5      8     13     21     34
-----
```

### Analisa :

- int fibo[100]: Array untuk menyimpan deret Fibonacci.
- int i, jumlah: Variabel untuk penggunaan dalam perulangan dan menyimpan jumlah elemen deret Fibonacci yang diminta.
- Elemen pertama dan kedua dari deret Fibonacci diinisialisasi dengan nilai 0 dan 1, sesuai dengan aturan dasar deret Fibonacci.
- Program meminta pengguna untuk memasukkan jumlah elemen deret Fibonacci yang ingin dihasilkan.

- Program menggunakan loop for untuk menghitung dan menyimpan nilai-nilai deret Fibonacci dalam array fibo. Perhitungan dilakukan dengan menjumlahkan dua elemen sebelumnya (fibo[i-1] dan fibo[i-2]).
- Program mencetak elemen-elemen deret Fibonacci ke layar dengan format penggantian baris setiap lima elemen (%5d).

## E. PROGRAM POINTER

### 1. Program pertama (pointer 1)

Source code

```

1  #include <stdio.h>
2  #include <conio.h>
3
4  int main(){
5      int y,x =2002;
6      int *px;
7
8      px = &x;
9      y = *px;
10
11     printf("Alamat x      = %p\n",&x);
12     printf("Isi Px        = %p\n",px);
13     printf("Isi X          = %d\n", x);
14     printf("Nilai *px      = %d\n", *px);
15     printf("Nilai Y        = %d\n", y);
16
17     getch();
18 }
```

Hasil Ouput :

```

Alamat x      = 000000000062FE10
Isi Px        = 000000000062FE10
Isi X          = 2002
Nilai *px      = 2002
Nilai Y        = 2002
```

### Analisa Program :

- Program ini menunjukkan cara menggunakan pointer untuk merujuk ke alamat memori variabel dan mengakses nilainya.
- Alamat variabel x dan alamat yang ditunjuk oleh pointer px sama, menunjukkan bahwa pointer px berhasil diinisialisasi dengan alamat variabel x.
- Nilai dari x dan nilai yang ditunjuk oleh pointer (\*px) adalah sama, menunjukkan bahwa pointer px dapat digunakan untuk mengakses nilai variabel x.
- Nilai variabel y setara dengan nilai yang ditunjuk oleh pointer (\*px), menunjukkan bahwa nilai dapat disalin dari pointer ke variabel lain.

### 2. Program kedua (pointer 2)

#### Source code

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  int main(){
5
6      int a,*b, **c;
7
8      a = 1975;
9      b = &a;
10     c = &b;
11
12     printf("Nilai a = %d atau %d atau %d\n",a, *b , **c);
13     printf("b = %p = alamat a di memori \n",b);
14     printf("c = %p = alamat b di memori \n",c);
15     printf("alamat c dimemori = %p \n",&c);
16
17     getch();
18 }
```

## Hasil Program

```
Nilai a = 1975 atau 1975 atau 1975  
b = 000000000062FE1C = alamat a di memori  
c = 000000000062FE10 = alamat b di memori  
alamat c di memori = 000000000062FE08
```

## Analisis Program

- **Pointer dan Indirection:** Program ini menggambarkan konsep pointer dan indirection di C. b adalah pointer yang menunjuk ke a, dan c adalah pointer yang menunjuk ke b. Dengan menggunakan \* dan \*\*, nilai dari a dapat diakses melalui b dan c.
- **Alamat Memori:** Program ini memperlihatkan alamat memori dari variabel a, pointer b, dan pointer c, serta alamat dari pointer c sendiri. Ini menggambarkan bagaimana variabel dan pointer disimpan di memori komputer.
- **Kepastian Output:** Output dari program ini tergantung pada alamat memori yang dialokasikan oleh sistem saat program dijalankan. Oleh karena itu, alamat memori yang sebenarnya akan berbeda setiap kali program dijalankan.
- **Header <conio.h>:** Penggunaan <conio.h> menunjukkan bahwa program ini ditulis untuk lingkungan DOS atau Windows. Harap diingat bahwa ini adalah header tidak standar dan mungkin tidak didukung di beberapa kompilator C modern atau di sistem operasi lainnya.



### 3. Program Ketiga ( pointer 3)

#### Source code

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  void naikan_nilai(int *x, int *y);
5
6  main(){
7      int a = 3, b = 7;
8
9      printf("SEMUA      : a = %d b = %d\n",a , b);
10     naikan_nilai(&a, &b);
11     printf("SEKARANG   : a = %d b = %d\n",a , b);
12     getch();
13 }
14 void naikan_nilai(int *x , int *y){
15     *x = *x + 2 ;
16     *y = *y + 3 ;
17 }
```

#### Hasil Program

```
SEMUA      : a = 3 b = 7
SEKARANG   : a = 5 b = 10
-----
```

#### Analisa program :

- Program ini mengilustrasikan penggunaan pointer untuk mengubah nilai variabel di luar fungsi yang mengirim variabel tersebut sebagai argumen.
- Variabel a dan b diubah nilainya di dalam fungsi naikan\_nilai melalui penggunaan pointer.
- Program menunjukkan konsep pengiriman alamat memori variabel melalui pointer dan modifikasi nilai variabel menggunakan pointer tersebut.

#### 4. Program keempat ( pointer 4)

##### Source code

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  char *nama_bulan(int n){
5      static char *bulan[] = {
6          "Kode bulan salah",
7          "Januari", "Februari", "Maret",
8          "April", "Mei", "Juni", "Juli",
9          "Agustus", "September", "Oktober",
10         "November", "Desember"
11     };
12     return((n<1 || n>12)?bulan[0]: bulan[n]);
13 }
14 int main(){
15     int b1;
16
17     printf("Masukan kode bulan [1..12] : ");
18     scanf("%d",&b1);
19     printf("Bulan ke-%d adalah %s\n", b1, nama_bulan(b1));
20
21     getch();
22 }
```

##### Hasil output

```
Masukan kode bulan [1..12] : 7
Bulan ke-7 adalah Juli
```

##### Analisa Program

- Program ini menggambarkan penggunaan fungsi, array, dan input/output dasar dalam bahasa C.
- Fungsi nama\_bulan menggunakan konsep array dan conditional operator (?) untuk mengembalikan nama bulan sesuai dengan kode bulan yang dimasukkan pengguna.
- Program memberikan respons yang sesuai jika pengguna memasukkan kode bulan yang tidak valid.
- Program ini mengajarkan konsep dasar manipulasi string dan penggunaan fungsi dalam bahasa C.

## 5. Program kelima (pointer 5)

### Source code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main() {
6      char str1[80], str2[80], *ptrx;
7
8      strcpy(str1, "INI ADALAH HURUF BESAR SEMUA!!!");
9      strcpy(str2, "ini adalah huruf kecil semua!!!");
10
11     ptrx = (char *)calloc(80, sizeof(char));
12     strcpy(ptrx, str1);
13
14     printf("Isi str1 = "); puts(str1);
15     printf("Isi str2 = "); puts(str2);
16     printf("Isi ptrx = "); puts(ptrx);
17
18     ptrx = strupr(str2);
19     printf("\nSetelah 'str2' diproses dengan strupr()\n");
20     printf("Isi str2 = "); puts(str2);
21     printf("Isi ptrx = "); puts(ptrx);
22
23     free(ptrx); // Dealokasi memori yang dialokasikan dengan calloc
24
25     return 0;
26 }
```

### Hasil output

```
Isi str1 = ini adalah huruf besar semua!!!
Isi str2 = ini adalah huruf kecil semua!!!
Isi ptrx = ini adalah huruf besar semua!!!

Setelah 'str2' diproses dengan strupr()
Isi str2 = INI ADALAH HURUF KECIL SEMUA!!!
Isi ptrx = INI ADALAH HURUF KECIL SEMUA!!!

-----
```

## Analisa

- Fungsi main dideklarasikan sebagai int main() dan blok program dibungkus dengan kurung kurawal { }.
- Penggunaan strcpy dan strlwr diperbaiki.
- Fungsi calloc digunakan untuk mengalokasikan memori dinamis untuk ptrx.
- Penggunaan printf digunakan untuk menampilkan teks ke layar.
- Memori yang dialokasikan dengan calloc dilepaskan menggunakan free pada akhir program.
- Program ini akan menyalin dua string ke variabel str1 dan str2, kemudian mengubah str1 menjadi huruf kecil dengan strlwr dan str2 menjadi huruf besar dengan strupr. Nilai-nilai ini kemudian dicetak ke layar menggunakan printf dan puts.

## PENUTUP

### KESIMPULAN

- Dalam bahasa pemrograman C, array adalah kumpulan variabel yang memiliki tipe data yang sama. Mereka menyimpan data secara berurutan di dalam memori. Dalam hal ini, setiap elemen dalam array memiliki indeks numerik yang dimulai dari 0.
- Pointer, di sisi lain, adalah variabel yang menyimpan alamat memori dari variabel lain. Pointer dapat merujuk ke alamat memori variabel atau objek lain, termasuk elemen-elemen dalam array.
- Dalam konteks array, nama array itu sendiri dapat dianggap sebagai pointer ke elemen pertama array. Dengan kata lain, jika Anda memiliki array `int arr[5]`, variabel `arr` adalah alamat dari `arr[0]`. Anda dapat menggunakan pointer untuk berinteraksi dengan array ini, misalnya untuk mengakses, memodifikasi, atau mengiterasi melalui elemen-elemen array.
- Contoh penggunaan pointer dengan array adalah dengan menggunakan aritmetika pointer. Misalnya, jika Anda memiliki pointer `int *ptr` yang merujuk ke elemen pertama dalam array, Anda dapat menggunakan `ptr++` untuk bergerak ke elemen berikutnya dalam array. Ini memungkinkan Anda mengakses dan memanipulasi elemen-elemen array menggunakan pointer.
- Dalam inti, pointer dan array berkaitan erat karena pointer dapat digunakan untuk mengakses dan memanipulasi elemen-elemen dalam array. Penggunaan pointer memungkinkan fleksibilitas dan kontrol tambahan dalam pemrograman C, memungkinkan manipulasi yang lebih dinamis dari data dalam array.