

# KAPAN PERLU MENGGUNAKAN POINTER

## A. Perlu Menggunakan Pointer

### 1. Alokasi Memori Dinamis:

- **Situasi:** Ketika Anda perlu mengalokasikan memori secara dinamis pada waktu eksekusi.
- **Mengapa:** Memori yang dialokasikan secara dinamis di heap bisa bertahan setelah fungsi yang mengalokasikannya selesai.
- **Contoh:** Menggunakan ``malloc``, ``calloc``, atau ``realloc`` untuk alokasi memori.

```
// Membuat variable player1 dg ukuran dinamis
struct Player *player1 = malloc(sizeof(struct Player));
```

### 2. Mengembalikan Alamat Variabel dari Fungsi:

- **Situasi:** Ketika Anda ingin fungsi mengembalikan lebih dari satu nilai atau mengembalikan struktur data kompleks(tipe bentukan).
- **Mengapa:** Mengembalikan pointer ke memori yang dialokasikan di heap memungkinkan data tersebut diakses di luar fungsi.
- **Contoh:** Fungsi yang membuat node baru dalam linked list.

```
c Copy code

struct Node *createNode(int data) {
    struct Node *newNode = malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
```

### 3. Mengakses dan Memodifikasi Variabel di Luar Fungsi:

- **Situasi:** Ketika Anda perlu memodifikasi variabel yang didefinisikan di luar cakupan fungsi.
- **Mengapa:** Memberikan pointer ke variabel memungkinkan fungsi untuk memodifikasi variabel tersebut.
- **Contoh:** Mengubah nilai variabel dalam fungsi.

```
c Copy code

void updateValue(int *ptr) {
    *ptr = 10;
}

int main() {
    int value = 5;
    updateValue(&value); // Mengirim alamat variabel ke fungsi
    printf("%d", value); // Akan mencetak 10
    return 0;
}
```

#### 4. Menggunakan Struktur Data yang Berhubungan:

- **Situasi:** Ketika Anda menggunakan struktur data seperti linked list, tree, graph, dll.
- **Mengapa:** Struktur data ini sering memerlukan pointer untuk menghubungkan elemen-elemen.
- **Contoh:** Node dalam linked list.

```
c Copy code  
  
struct Node {  
    int data;  
    struct Node *next;  
};
```

### B. Tidak Perlu Menggunakan Pointer

#### 2. Mengembalikan Nilai Tunggal dari Fungsi:

- **Situasi:** Ketika Anda hanya perlu mengembalikan nilai tunggal dari fungsi.
- **Mengapa:** Mengembalikan nilai secara langsung lebih sederhana dan lebih mudah dipahami.
- **Contoh:** Fungsi yang mengembalikan integer.

```
c Copy code  
  
int add(int a, int b) {  
    return a + b;  
}
```

#### 3. Penggunaan Struktur Data Statis:


- **Situasi:** Ketika Anda menggunakan struktur data yang memiliki ukuran tetap.
- **Mengapa:** Alokasi statis lebih sederhana dan efisien jika ukuran struktur data sudah diketahui.
- **Contoh:** Array statis.

```
c Copy code  
  
int array[10]; // Array statis dengan 10 elemen
```

## 1. Variabel Lokal dalam Fungsi:

- **Situasi:** Ketika Anda hanya perlu variabel sementara dalam fungsi.
- **Mengapa:** Variabel lokal otomatis dikelola oleh stack dan akan dihapus setelah fungsi selesai.
- **Contoh:** Variabel biasa dalam fungsi.

c

 Copy code

```
void printValue() {  
    int value = 10;  
    printf("%d", value);  
}
```