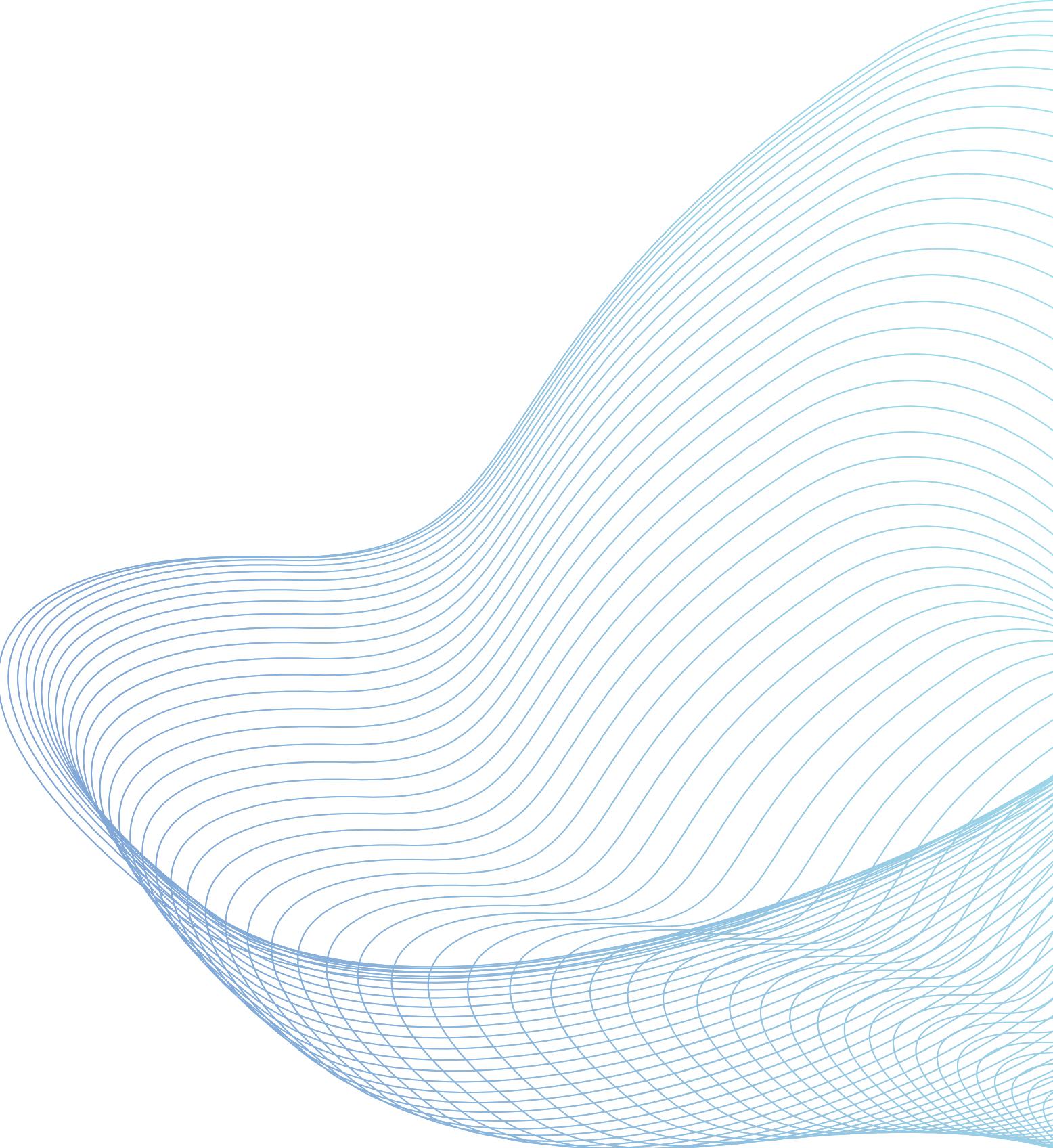




FUNGSI REKURSIF



TUJUAN BELAJAR

- Memahami konsep rekursi dalam pemrograman
- Mengetahui cara kerja fungsi rekursif
- mempelajari implementasi rekursif dalam berbagai kasus

Tapi apa itu rekursif?



PENGERTIAN

- Rekursi adalah metode pemrograman di mana suatu **fungsi memanggil dirinya sendiri**
- Digunakan untuk menyelesaikan masalah yang dapat **dipecah menjadi submasalah yang lebih kecil** [solusi yang diberikan oleh fungsi]
- **Harus memiliki kondisi dasar (base case)** untuk menghindari rekursi tak hingga



```
1 int fungsiRekursif(int input) {  
2     if (statement)  
3         return x;  
4     else  
5         return fungsiRekursif(x);  
6 }
```

CONTOH LAIN

Misal kita buat fungsi rekursif untuk Deret Penjumlahan, yang rumusnya seperti ini :

Jika input 5 maka jadinya --> $5 + 4 + 3 + 2 + 1$

Kalo kita analisis dulu, berarti kita input 5, terus berkurang jadi 4. Berarti $5 - 1$.

Terus hasilnya ditambahin, berarti $x = x + (5-1)$

Terus misal inputin angka 1, kan bilangan 1 selesai jadi langsung kembaliin 1.

Codenya jadi begini

```
● ● ●  
1 int sum(int n) {  
2     if (n == 1)  
3         return 1;  
4     else  
5         return n + sum(n - 1);  
6 }
```

```
● ● ●  
1 int main() {  
2     cout << "Deret Penjumlahan = " << endl;  
3     cout << "5 + 4 + 3 + 2 + 1 = " << sum(5) << endl;  
4  
5     return 0;  
6 }
```

Hasil

```
1+2+3+4+5 = 15  
15
```

BEBERAPA PENULISAN FUNGSI NYA

```
● ○ ●  
int sum(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n + sum(n - 1);  
}
```

```
● ○ ●  
int sum(int x){  
    return x + sum(x - 1);  
}
```

```
● ○ ●  
int sum(int y){  
    return (y == 1) ? 1 : y + sum(y - 1);  
}
```

```
● ○ ●  
int sum(int a) {  
    int hasil = 0;  
  
    if (a == 1) {  
        hasil = 1;  
    } else {  
        hasil = a + sum(a - 1);  
    }  
  
    return hasil;  
}
```

REKAP DIKIT

Yang perlu diperhatikan

- Ada sebuah yang Kondisi (if else) untuk menghentikan rekursif
- Ada perintah berhenti dan ada yang memanggil fungsinya lagi
- Bisa terjadi Stack Overflow kalo kondisinya ngawur (nanti muter muter terus)

Kelebihan Rekursif

- Lebih mudah dipahami dalam beberapa kasus.
- Berguna untuk masalah yang dapat dipecah menjadi submasalah kecil.

Kekurangan Rekursif

- Lebih lambat dibandingkan iterasi karena penggunaan memori stack.
- Risiko stack overflow jika tidak memiliki base case yang jelas.

Semoga paham, kita coba praktik

KASUS 1: DERET GEOMETRI

Buat program rekursif untuk menghitung jumlah deret geometri: $S=a+ar+ar^2+ar^3+\dots+ar^n$
Dengan **a** sebagai suku pertama, **r** sebagai rasio, dan **n** sebagai jumlah suku.

Berarti kita harus jumlahin per suku nya, terus nanti ditambah.

Kita perlu fungsi bawaan yaitu **pow(pangkat, perulangan)** buat menghitung 1 suku, terus nanti panggil suku lain tapi dikurangi **n-1**

Terus kalo udah sampe 0 berarti kita kembalikan nilai a nya. **[S = a]** + ar ...

Jika **[n = 0]** berarti kembalikan **[a]**,

Kalo tidak **[hasil suku] + [deretGeomter perulangan - 1]**



KASUS 1: DERET GEOMETRI

Buat program rekursif untuk menghitung jumlah deret geometri: $S=a+ar+ar^2+ar^3+\dots+ar^n$
Dengan **a** sebagai suku pertama, **r** sebagai rasio, dan **n** sebagai jumlah suku.

Berarti kita harus jumlahin per suku nya, terus nanti ditambah.

Kita perlu fungsi bawaan yaitu **pow(pangkat, perulangan)** buat menghitung 1 suku,

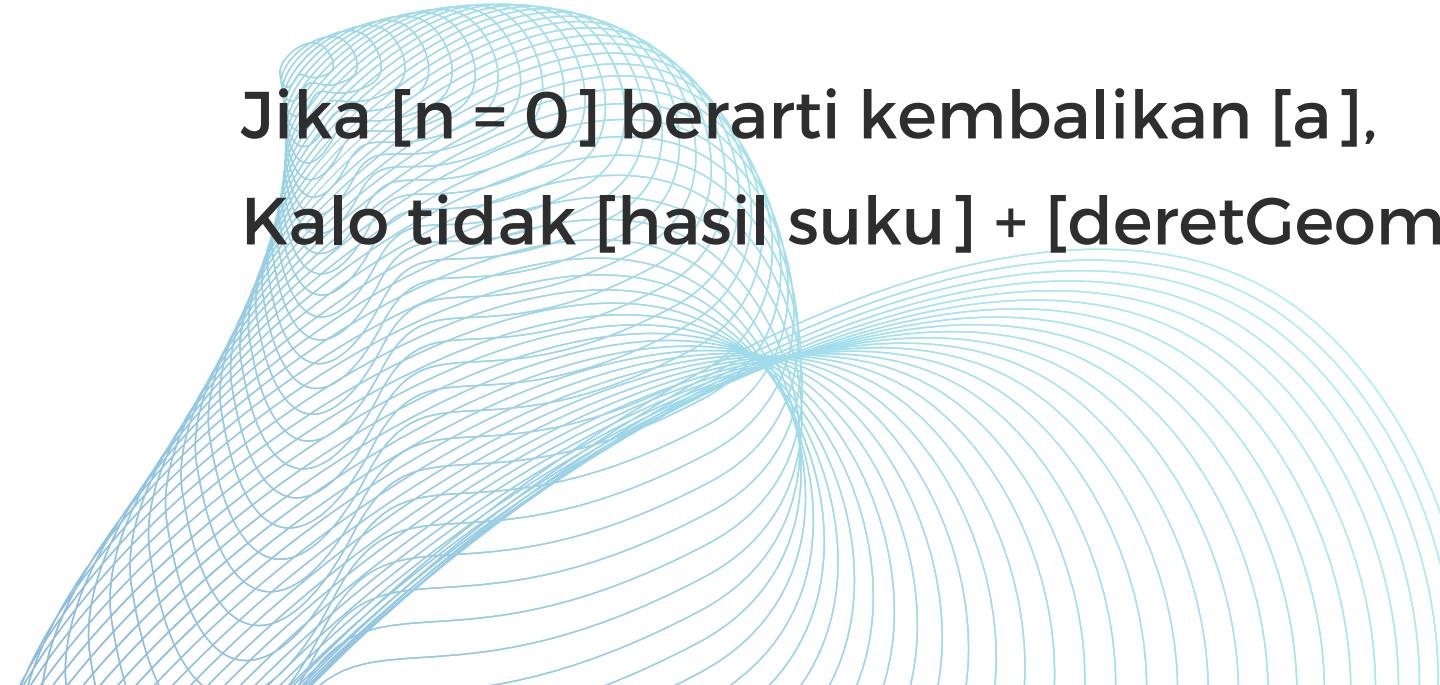
Note: Karena pake pow berarti kita perlu librarti cmath

Terus nanti panggil suku lain tapi dikurangi **n-1**

Terus kalo udah sampe 0 berarti kita kembalikan nilai a nya. **[S = a] + ar ...**

Jika **[n = 0]** berarti kembalikan **[a]**,

Kalo tidak **[hasil suku] + [deretGeomter perulangan - 1]**



HASIL KASUS 1



```
double deretGeometri(int a, int r, int n) {
    if (n == 0) return a; // Base case
    return a * pow(r, n) + deretGeometri(a, r, n - 1);
}

int main() {
    int a, r, n;
    cout << "Masukkan suku pertama (a): ";
    cin >> a;
    cout << "Masukkan rasio (r): ";
    cin >> r;
    cout << "Masukkan jumlah suku (n): ";
    cin >> n;

    cout << "Jumlah deret geometri: " << deretGeometri(a, r, n - 1) << endl;
    return 0;
}
```

Input

a = 3 ; r = 2 ;n = 3

Hasil

Masukkan suku pertama (a): 3
Masukkan rasio (r): 2
Masukkan jumlah suku (n): 3
Jumlah deret geometri: 21

KASUS 2 : REKUSI PADA ARRAY

Buat fungsi rekursif yang menghitung jumlah elemen dalam array tanpa menggunakan loop.

Tips parameternya array.

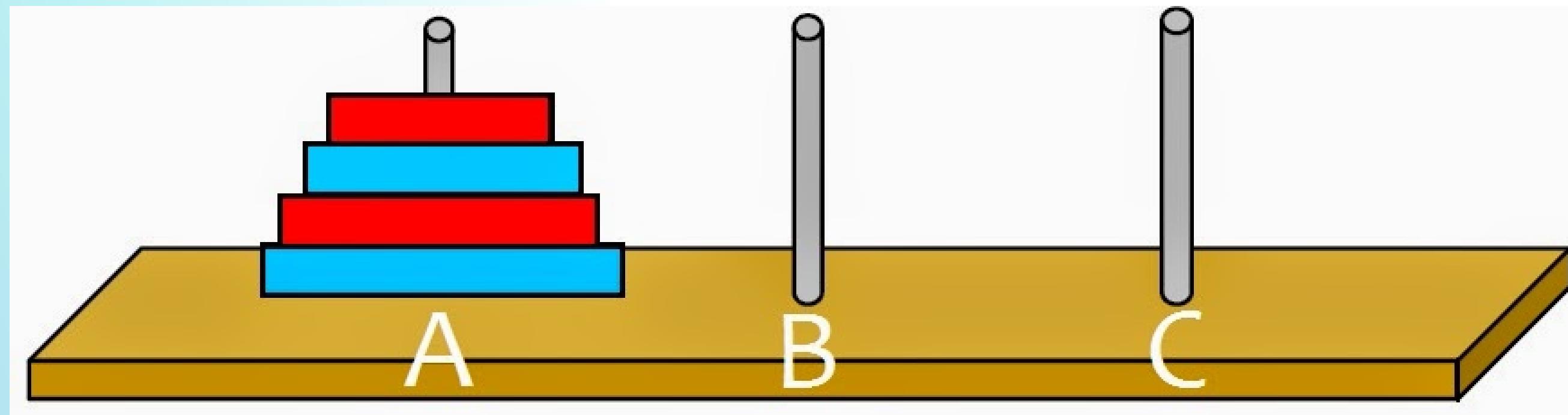


```
int sumArray(int arr[], int n) {  
    // isikan kode Le  
}
```

Kurang greget?

KASUS 3 : MENARA HANOI

Buat program rekursif untuk menyelesaikan Menara Hanoi dengan tiga tiang dan sejumlah cakram.



**KERJAKAN
DULU LE**

