SUB PROGRAM & REKURSIF

ALGORITMA DAN STRUKTUR DATA

Institut Teknologi Sumatera

TUJUAN KULIAH

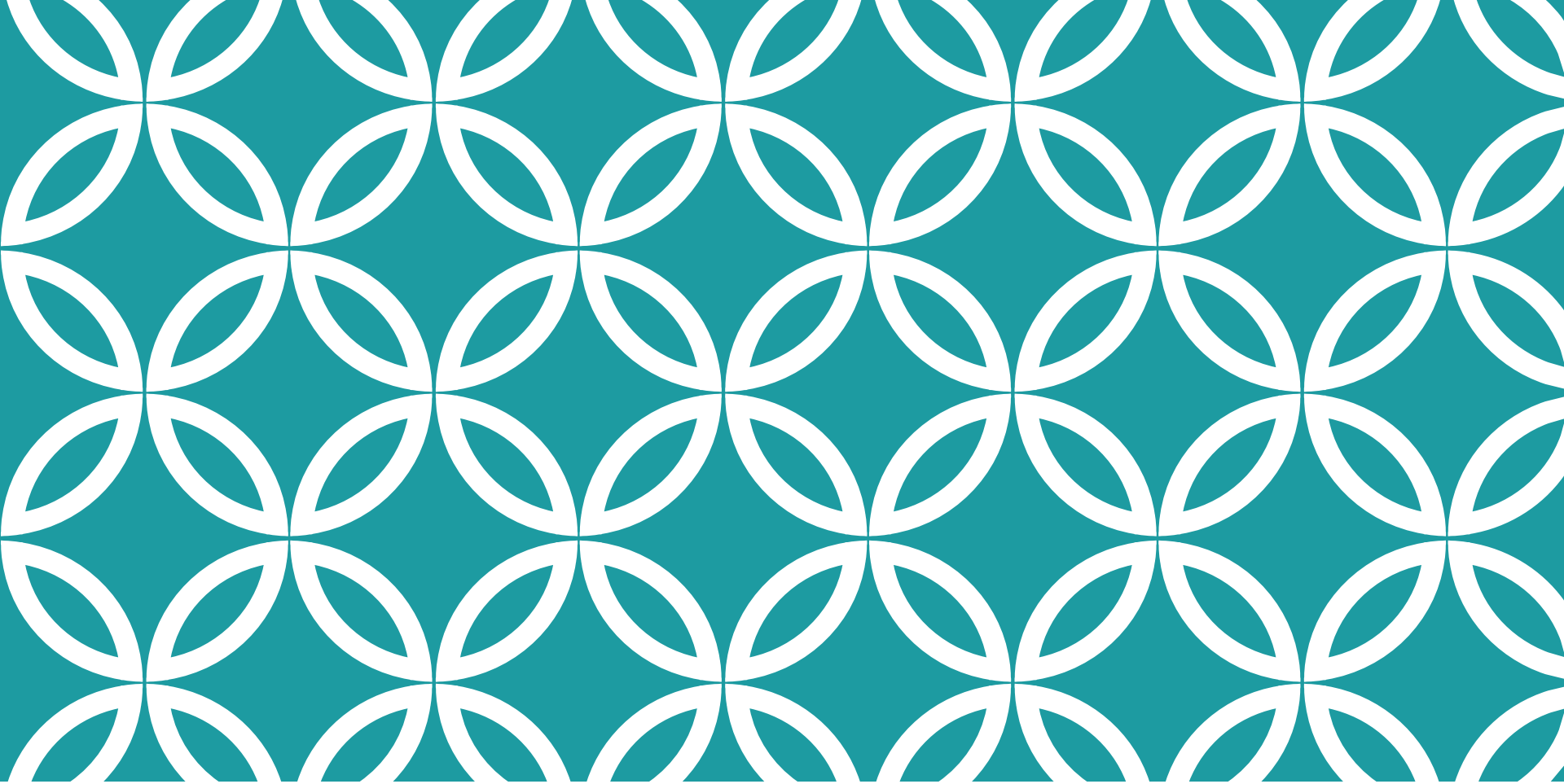
- Review sub-program
- Review passing by parameter
- Review prinsip rekursif dalam pemrograman

MATERI

- Sub Program
- Rekursif

PRE TEST

- Apakah yang dimaksud dengan sub-program?
- Ada berapa tipe pemanggilan sebuah sub-program? Sebutkan
- Apakah rekursif itu?
- Apakah perulangan dan rekursif sama?



SUB PROGRAM

SUB PROGRAM

Blok program yang merupakan sekumpulan instruksi yang bertujuan untuk menyelesaikan suatu tugas khusus. Sebuah sub progam dibuat untuk membantu mengerjakan tugas yang kompleks secara efektif dan efisien.

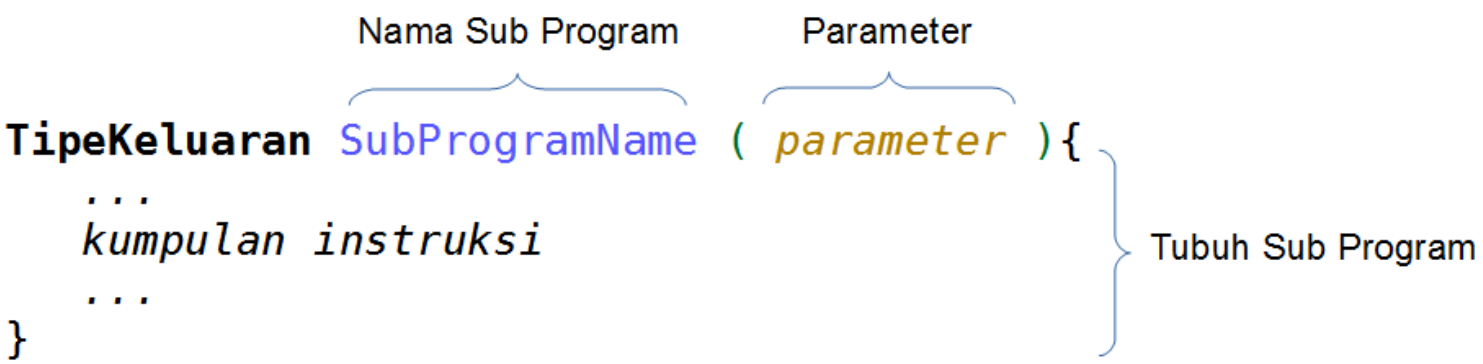
SUB PROGRAM

- Semakin besar program, akan semakin banyak bagian kode yang berulang
- Sangat tidak efisien jika bagian kode yang sama/serupa diketik berulang-ulang atau bahkan termasuk kalau di-copy paste
- Di samping itu, dalam banyak persoalan, ada berbagai rumus/formula yang berulang-ulang dipakai dalam satu program
- Bagaimana jika ada cara supaya bagian kode tersebut tidak perlu diketik berulang-ulang, tapi tetap dapat digunakan berkali-kali dalam program yang sama

TIPE SUB PROGRAM

- Dalam deklarasinya, sebuah sub program dibagi menjadi dua tipe, yaitu: fungsi dan prosedur.
- Fungsi void, atau disebut juga dengan prosedur.

DEKLARASI


Nama Sub Program
Parameter
TipeKeluaran SubProgramName (parameter){
...
kumpulan instruksi
...
}
Tubuh Sub Program

CONTOH

```
#include <iostream.h>

int mintaSeribu(){
    return 1000;
}

int main(){
    int i;
    cout << "Contoh pembuatan fungsi\n";
    i = mintaSeribu();

    cout << "Nilai dari variabel i = " << i;
    return 0;
}
```

PARAMETER

- Seringkali disebuah sebagai argumen atau nilai masukan dari sebuah fungsi.
- Jenis parameter:
  - Parameter formal, yaitu variabel yang dituliskan pada saat deklarasi fungsi.
  - Parameter aktual, yaitu variabel yang dituliskan pada saat pemanggilan sebuah fungsi.

LATIHAN

Buatlah prosedur untuk mengalikan dua buah bilangan

Initial State (I.S.) :

- diberikan nilai integer a dan b

Final State (F.S.) :

- hasil = a * b

CONTOH

```
#include <iostream>
using namespace std

int perkalian(int bil_1, int bil_2){
    int hasil;
    hasil = bil_1 * bil_2;
    return hasil;
}

int main(){
    int a,b,c;

    a = 5;
    b = 10;
    c = perkalian(a,b);

    cout << "Hasil perkalian antara a * b = " << c;

    return 0;
}
```

FUNGSI NON VOID

- Fungsi yang jika dipanggil akan mengembalikan sebuah nilai.
- Nilai-nilai yang dikembalikan dapat berupa int, float, char atau tipe data yang lainnya.

CONTOH

Tuliskan fungsi MAX2, yang menerima masukan dua buah bilangan integer dan menghasilkan bilangan terbesar

- Contoh: MAX2(1,2) → 2

JAWABAN

```
int Max2 (int a1, int b1) {
    // diberikan a1 dan b1, menghasilkan a1 jika
a1 >= b1,
    // dan b1 jika b1 > a1
    //Algoritma
    if (a1 >= b1) {
        return a1;
    } else { // a1 < b1
        return b1;
    }
}
```

LATIHAN

Tuliskan fungsi AVG, yang menerima masukan tiga buah bilangan integer dan menghasilkan nilai reratanya

- Contoh: AVG(2,3,4) → 3

LATIHAN

Tuliskan fungsi **Aneh** yang dapat menerima 2 buah masukan bilangan bulat, yaitu variabel A dan variabel B.

- Apabila variabel A lebih besar maka hitung hasil A * B
- Apabila variabel B lebih besar maka hitung A - B

PROSEDUR

- Fungsi yang jika dipanggil tidak mengembalikan nilai apapun.
- Seringkali disebut juga dengan istilah procedure.
- Ditandai dengan dituliskannya kata kunci void sebagai tipe keluaran fungsi.

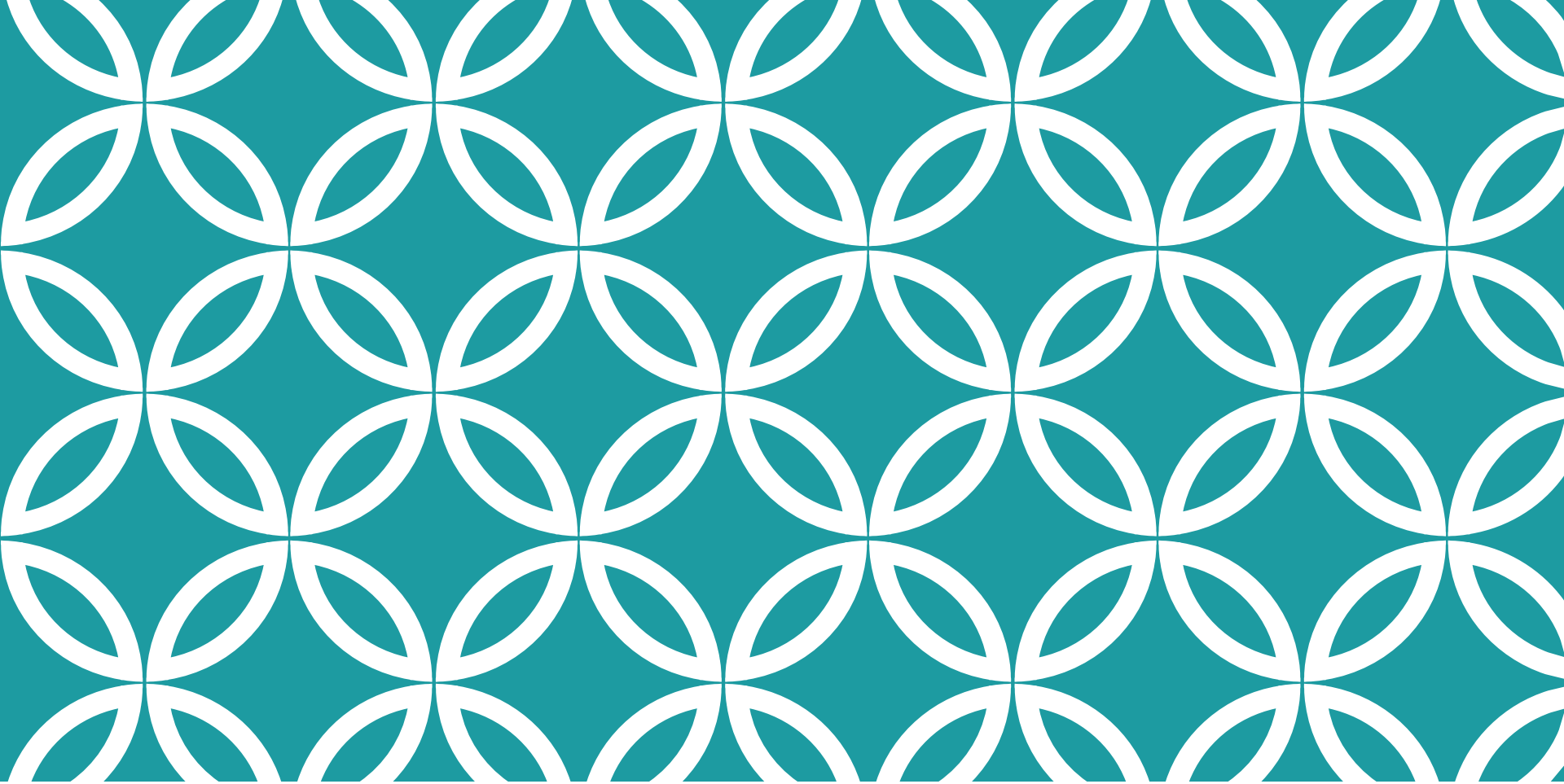
CONTOH

```
#include <iostream>
using namespace std

void cetakGaris(){
    cout << "--------------------------------------" << endl;
}

int main(){

    cetakGaris();
    cout << "Berikut ini adalah fungsi cetak garis" << endl;
    cetakGaris();

    return 0;
}
```

APA KEGUNAAN PROSEDUR?

PASSING PARAMETER

- Passing by value
- Passing by reference



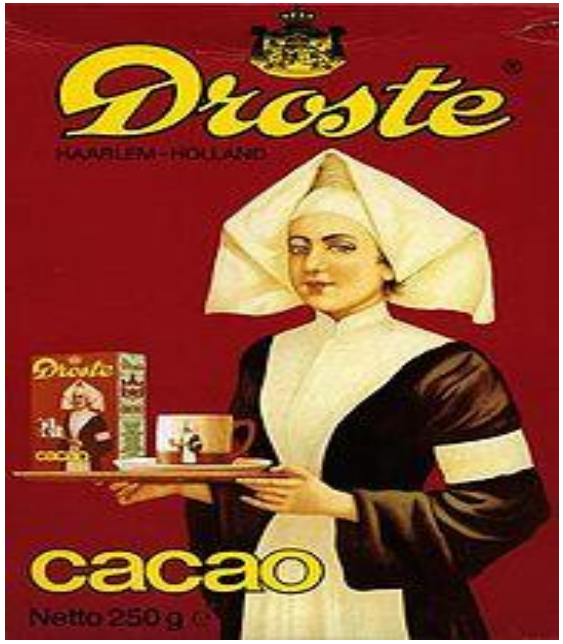
REKURSIF

Recursion is all about breaking a big problem into smaller occurrences of that same problem.

REKURSIF

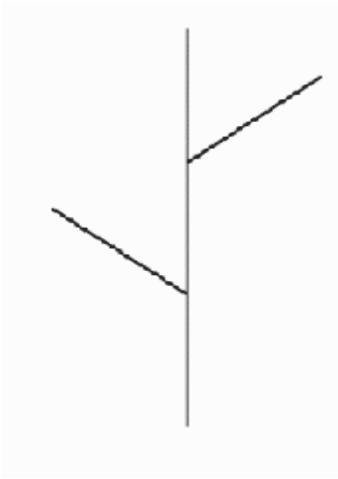
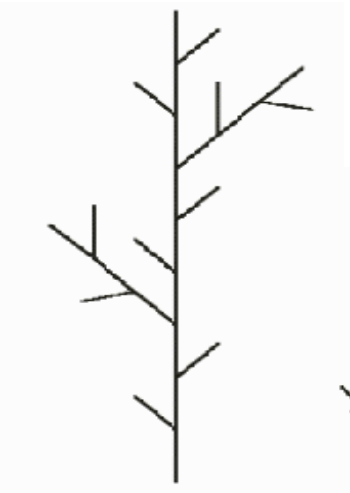
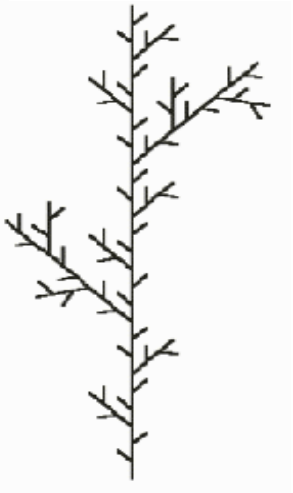
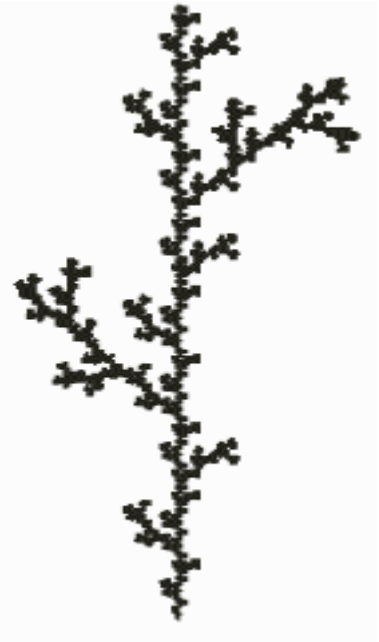
- Merupakan suatu fungsi atau prosedur → tidak bisa diimplementasikan sebagai program utama
- Proses yang memanggil dirinya sendiri

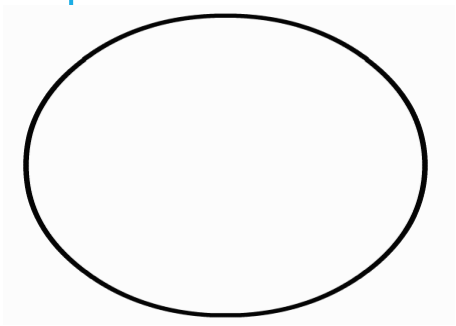
REKURSIF

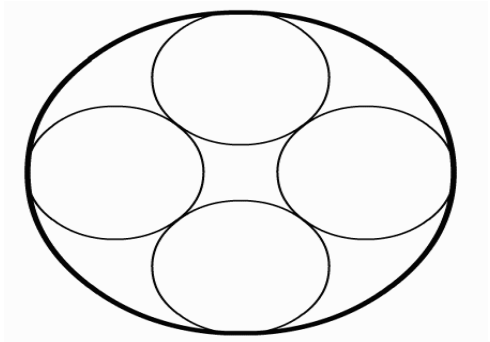

Droste
HAARLEM - HOLLAND
cacao
Netto 250 g

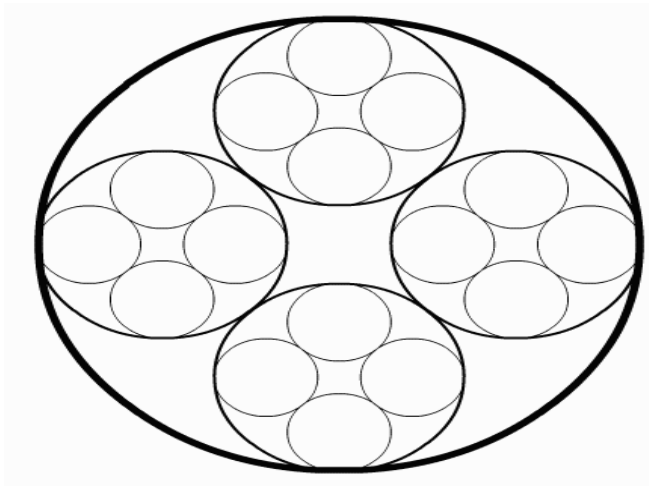

16
PORTIONS/PORTIES
La vache qui rit
Calcium
280g

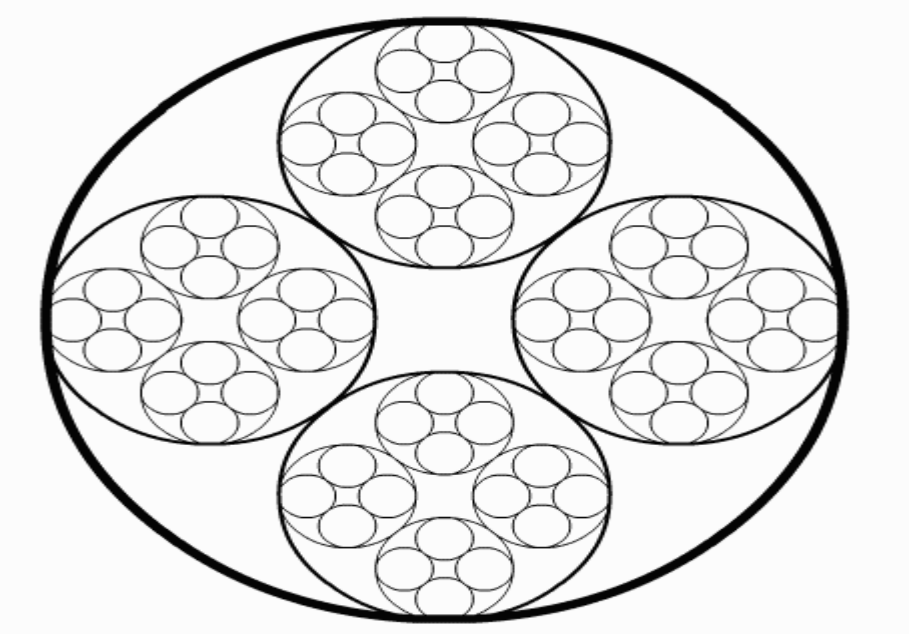
CONTOH REKURSIF





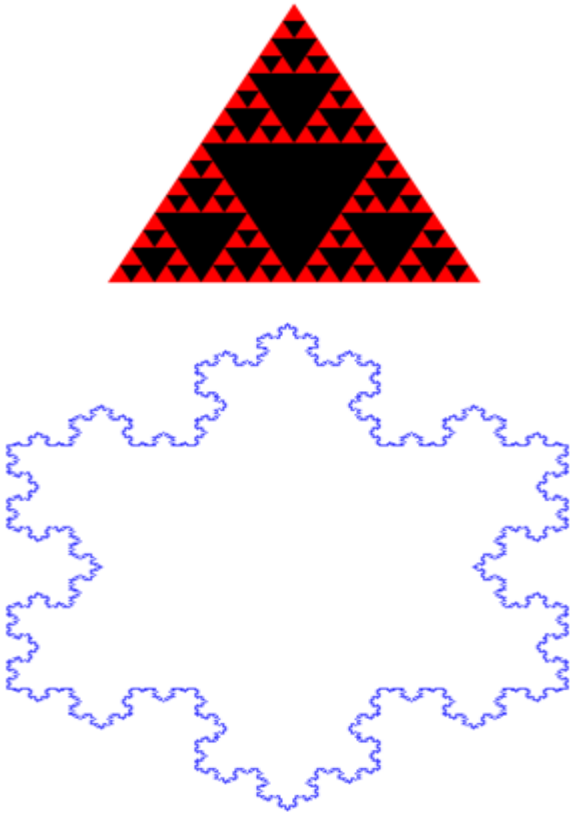




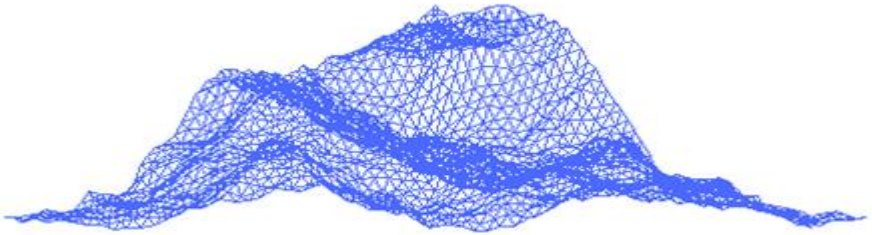
APLIKASI REKURSIF DALAM BIDANG INFORMATIKA

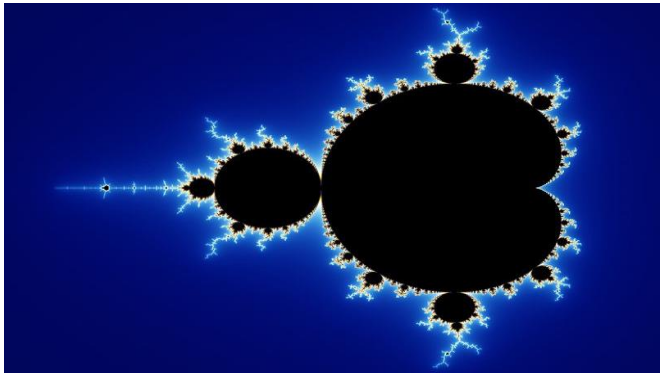
Fraktal:

- bentuk-bentuk geometris yang terdiri atas bagian-bagian kecil, tiap bagian adalah kopi (dalam ukuran yang lebih kecil) dari bentuk keseluruhan
- Biasanya diimplementasikan dari fungsi matematis yang bersifat rekursif



CONTOH-CONTOH GAMBAR FRAKTAL





KOMPONEN REKURSIF

- Kondisi berhenti **(Basis/Base Case)**
- Kondisi rekursif yaitu memanggil dirinya sendiri dengan parameter tertentu **(Rekurens)**

ANALISIS REKURENS

Solusi rekursif terdiri dari dua bagian:

- **Basis**, kasus yang menyebabkan fungsi berhenti, karena jawaban sudah bisa diperoleh
- **Rekurens** : mengandung call terhadap fungsi tersebut, dengan parameter bernilai mengecil (biasanya) atau menuju basis.

Solusi rekursif memiliki sekurang-kurangnya satu kasus basis dan satu kasus rekursif.

- Dimungkinkan ada lebih dari satu kasus basis maupun rekursif.

FAKTORIAL

- Fungsi faktorial dari bilangan bulat positif n didefinisikan sebagai berikut:

$$n! = 1 \quad \text{jika } n = 0 \quad \text{(basis)}$$

$$n! = n * (n-1)! \quad \text{jika } n > 0 \quad \text{(rekurens)}$$

- Contoh :

$$3! = 3 * 2!$$

$$3! = 3 * 2 * 1!$$

$$3! = 3 * 2 * 1 * 0!$$

$$3! = 3 * 2 * 1 * 1$$

$$3! = 6$$

FAKTORIAL

- Konsep Faktorial

  $n! = n(n-1)(n-2)\ldots1$

- Dapat diselesaikan dengan
  - Cara Biasa (secara iteratif, dengan pengulangan)
  - Rekursif

FAKTORIAL : CARA ITERATIF (DENGAN PENGULANGAN)

```
int Faktorial(int n)
// Menghasilkan faktorial n (n!). Asumsi n >= 0
{
	int S, i;
	S = 1;
	i = 1;
	while (i <= n) {
		S = S * i;
		i++;
	} // i > N
	return S;
}
```

FAKTORIAL : DENGAN REKURSIF

Basis → jika n = 0, maka n! = 1

Rekurens → jika n > 0, maka n! = n * (n-1) * (n-2) * … * 1 = n * (n-1)!

(n – 1) !

Contoh:

0! = 1

1! = 1 * 0! = 1 * 1 = 1

5! = 5 * 4! = 120

FAKTORIAL : DENGAN REKURSIF

```
int Faktorial(int n)
// Menghasilkan faktorial n, asumsi n >= 0, dengan cara rekursif
{
	if (n == 0) { // Basis
		return 1;
	} else {	// n > 0; Rekurens
		return (n * Faktorial(n-1));
	}
}
```

DERET FIBONACCI

- Diciptakan oleh Leonardo Fibonacci berasal dari Italia 1170-1250
- Deret Fibonacci f_1, f_2,... didefinisikan secara rekursif sebagai berikut :

$$f_1 = 1$$

$$f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2} \quad \text{untuk } n > 2$$

- Deret: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597,...

CONTOH

Untuk n = 4, proses perhitungan Fibonacci dapat dilakukan sebagai berikut:

$$f_4 = f_3+f_2$$

$$f_4 = (f_2+f_1) + f_2$$

$$f_4 = (1+1) +1$$

$$f_4 = 3$$

DERET FIBONACCI

Basis → $n = 1, f(n) = 1$

$n = 2, f(n) = 1$

Rekurens → $n > 2, f(n) = f(n-1) + f(n-2)$

DERET FIBONACCI (VERSI FUNGSI)

```
int Fib (int n)
// Menghasilkan deret fibonacci pada suku ke-n
{
        if (n == 1) { // Basis-1
                return 1;
        } else if (n == 2) { // Basis-2
                return 1;
        } else {  // n > 2, rekurens
                return (Fib(n-1) + Fib(n-2));
        }
}
```

DERET FIBONACCI (VERSI PROSEDUR)

```
void PFib (int n, int *F)
// I.S. n adalah suku deret fibonacci, terdefinisi
// F.S. F berisi nilai deret fibonacci pada suku ke-n
{       int F1, F2;

        if (n == 1) {           // Basis-1
                *F = 1;
        } else if (n == 2) { // Basis-2
                *F = 1;
        } else {                // n > 2, rekurens
                PFib(n-1, &F1);
                PFib(n-2, &F2);
                *F = F1 + F2;
        }
}
```

KESIMPULAN

Fungsi rekursif merupakan fungsi yang memanggil dirinya sendiri. Terdapat dua komponen penting dalam fungsi rekursif, yaitu kondisi kapan berhentinya fungsi (BASIS) dan pengurangan atau pembagian data ketika fungsi memanggil dirinya sendiri (REKURENS).

LATIHAN 1

Buat program yang menerima masukan sebuah integer n dan menuliskan ke layar: 1+2+3+4+5+...+n jika n > 0.

Jika n <= 0, tuliskan: “Tidak dapat dihitung”.

Program ini harus mengandung fungsi yang menerima masukan n (asumsi n > 0) dan menghasilkan: 1+2+3+4+5+...+n dengan menggunakan cara rekursif.

LATIHAN 2

Menggunakan cara rekursif, buat fungsi yang menerima masukan n (integer > 0) dan mencetak ke layar hasil jumlahannya. Contoh: 2+4+6+8+10+...+(2*n)

LATIHAN 3

Buatlah fungsi **DeretSegitiga**, yang merupakan fungsi untuk mencari nilai bilangan ke-n pada deret segitiga. Fungsi harus diselesaikan dengan cara rekursif. Contoh:
Deret segitiga: 1, 3, 6, 10, 15, …

LATIHAN 4

Buatlah fungsi **IsGanjil** yang menerima masukan bilangan integer n $\geq$0 dan menghasilkan true jika n adalah bilangan ganjil dan false jika tidak. Fungsi harus diselesaikan dengan cara rekursif.

Contoh: n = 0 → false

n = 1 → true

n = 4 → false

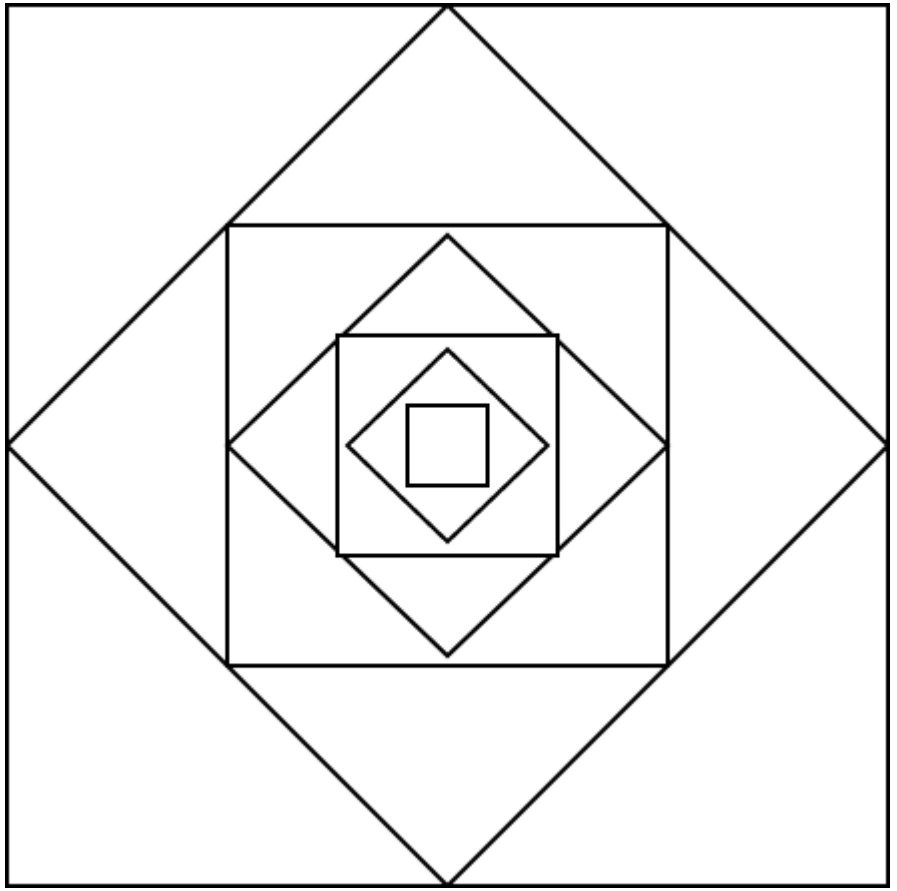
n = 11 → true

LATIHAN 5

Buatlah fungsi **LuasBS** yang menerima masukan nilai sisi sebuah bujur sangkar dengan nilai > 0 dan menghasilkan luas bujur sangkar dengan panjang sisi tersebut. Fungsi harus dikerjakan secara rekursif.

Contoh: sisi = 1 → luas = 1
sisi = 2 → luas = 1 + 3 = 4
sisi = 3 → luas = 4 + 5 = 9
sisi = 4 → luas = 9 + 7 = 16
sisi = 5 → luas = 16 + 9 = 25
sisi = 8 → luas = 49 + 15 = 64

LATIHAN 6



$P = 100 * 4 + 50\sqrt{2} * 4 + 50 * 4 \ldots\ldots + n * 4$

LATIHAN 7

```
int Aneh(int z){
    if(z > 0){
        if(z % 2 == 0){
            return z - Aneh(z-3) * Aneh(z-2);
        } else {
            return z + Aneh(z-2) * Aneh(z-1);
        }
    } else {
        return z;
    }
}
```

- Tuliskan luaran dari fungsi tersebut diatas apabila diberikan Aneh(6)
- Gambarkan setiap tahapan yang dilalui menggunakan pohon eksekusi!

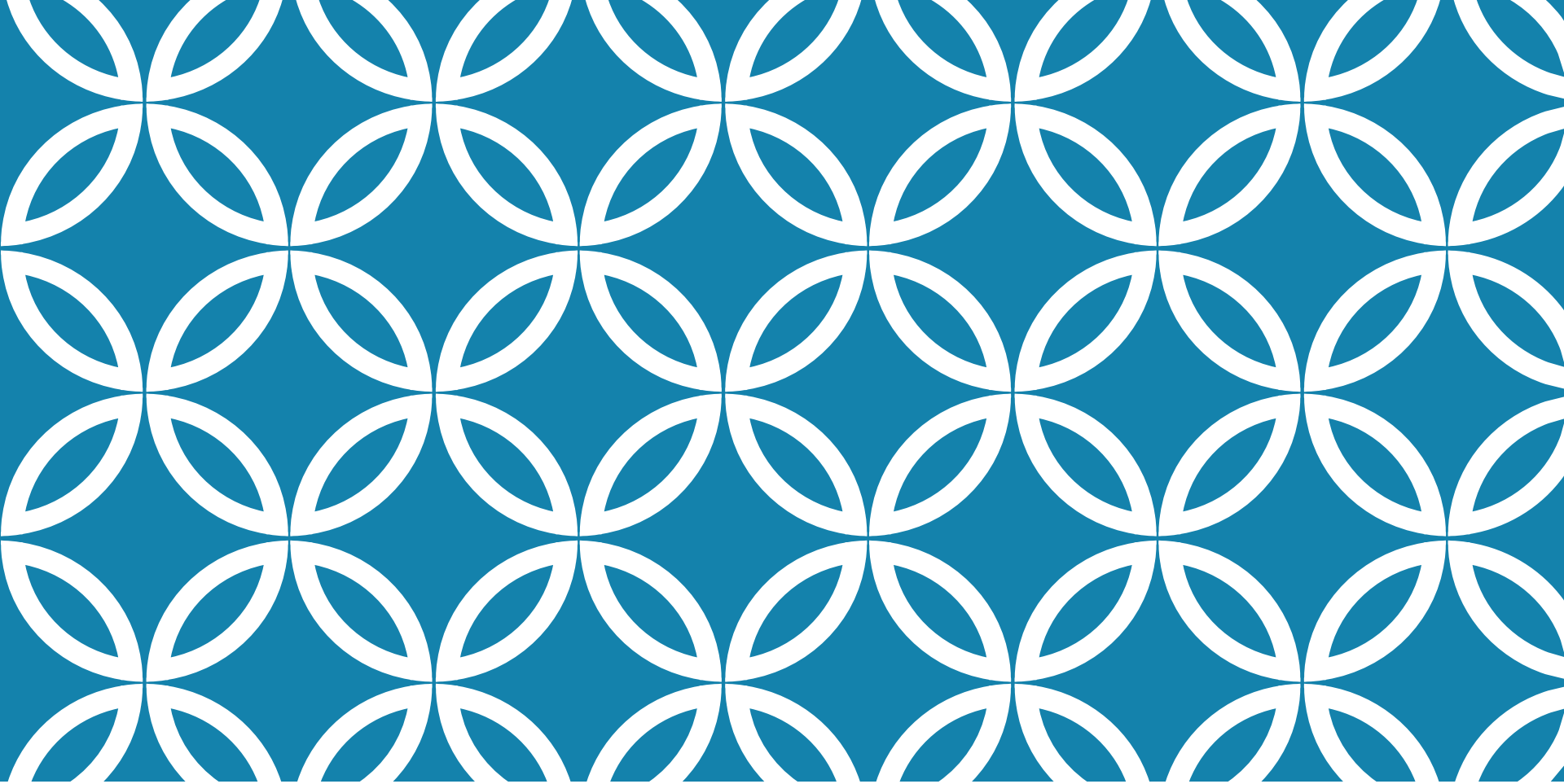
POST TEST

- Apa perbedaan mendasar passing parameter by value dengan passing parameter by reference?
- Kapan kita perlu menggunakan passing paramter by reference?
- Apakah permasalahan yang dapat diselesaikan dengan perulangan dapat diselesaikan dengan rekursif? Jelaskan!

PR

Buatlah sebuah program dengan menggunakan prinsip rekursif untuk mencari faktor persekutuan terbesar (FPB) dari dua buah bilangan A dan B! Bilangan A dan B didapat dari masukan oleh pengguna.

Bagaimana jika program dibuat menjadi versi dengan menggunakan perulangan?



MATERI SUPLEMEN REKURSIF

PENYELESAIAN PROBLEM REKURSIF DENGAN POHON REKURSIF

Oleh:
I Wayan Wiprayoga W.
Teknik Informatika
ITERA

PROGRAM BILANGAN FIBONACI KE-N

```
int fibo(int n) {
	if (n == 1) {
		return 1;
	} else if (n == 2) {
		return 1;
	}
	else { return fibo(n-1) + fibo(n-2) };
}
```

Apa output program jika dipanggil dengan menggunakan ekspresi **fibo(4)**?

PEMANGGILAN FIBO(4)

Pemanggilan pertama, fibo(4), diagram pohon nya adalah sebagai berikut:

fibo(4)

PROGRAM BILANGAN FIBONACI KE-N

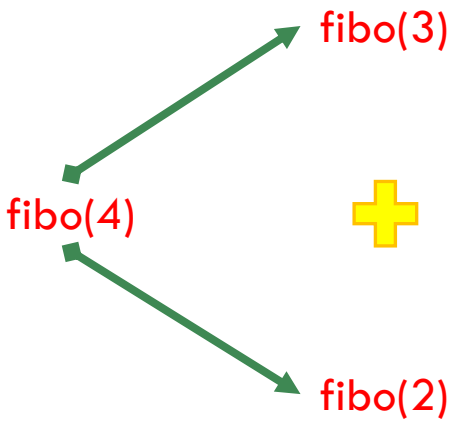
```
int fibo(int n) {
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 1;
        }
        else { return fibo(n-1) + fibo(n-2) };
}
```

n = 4

fibo(3) fibo(2)

PEMANGGILAN FIBO(3) + FIBO(2)

Pemanggilan fibo(4) mengakibatkan pemanggilan fibo(3) dan fibo(2), diagram pohonnya menjadi:

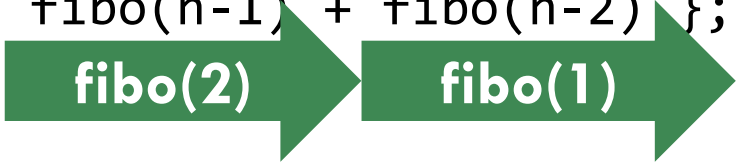


Hasilnya fibo(3) + fibo(2)

PROGRAM BILANGAN FIBONACI KE-N

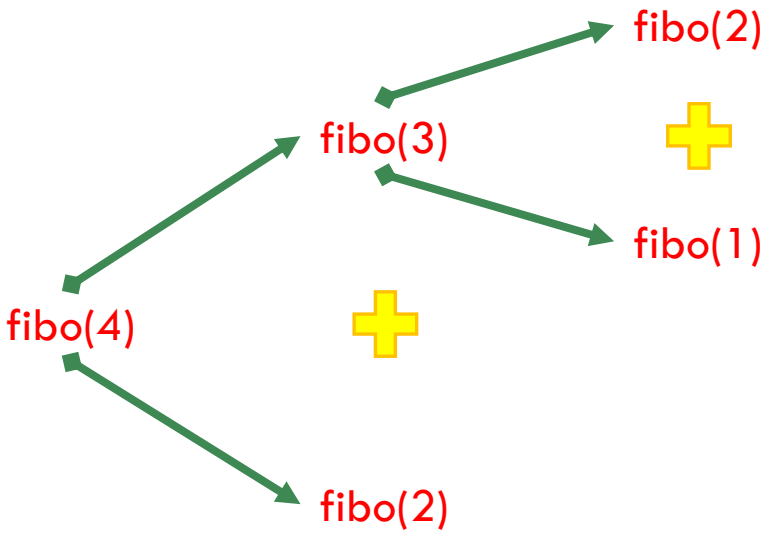
n = 3

```
int fibo(int n) {
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 1;
        }
        else { return fibo(n-1) + fibo(n-2) };
}
```



PEMANGGILAN FIBO(3)

Pemanggilan fibo(3) mengakibatkan pemanggilan fibo(2) dan fibo(1), seperti berikut:



PROGRAM BILANGAN FIBONACI KE-N

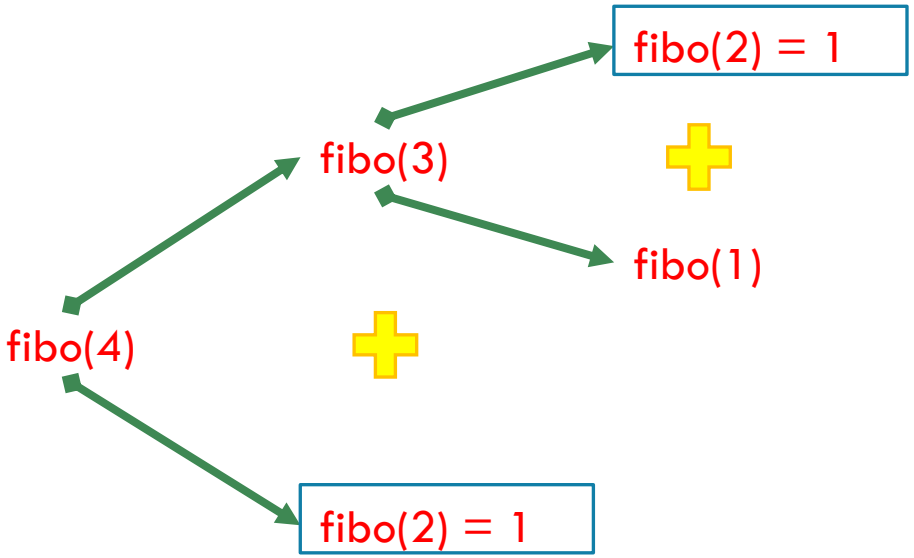
```
int fibo(int n) {
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 1;
        }
        else { return fibo(n-1) + fibo(n-2) };
}
```

n = 2

return 1

PEMANGGILAN FIBO(2)

fibo(2) >> return 1; nilai fibo(2) = 1



PROGRAM BILANGAN FIBONACI KE-N

```
int fibo(int n) {
        if (n == 1) {
                return 1;
        } else if (n == 2) {
                return 1;
        }
        else { return fibo(n-1) + fibo(n-2) };
}
```

n = 1

PEMANGGILAN FIBO(1)

fibo(1) >> return 1; nilai fibo(1) = 1

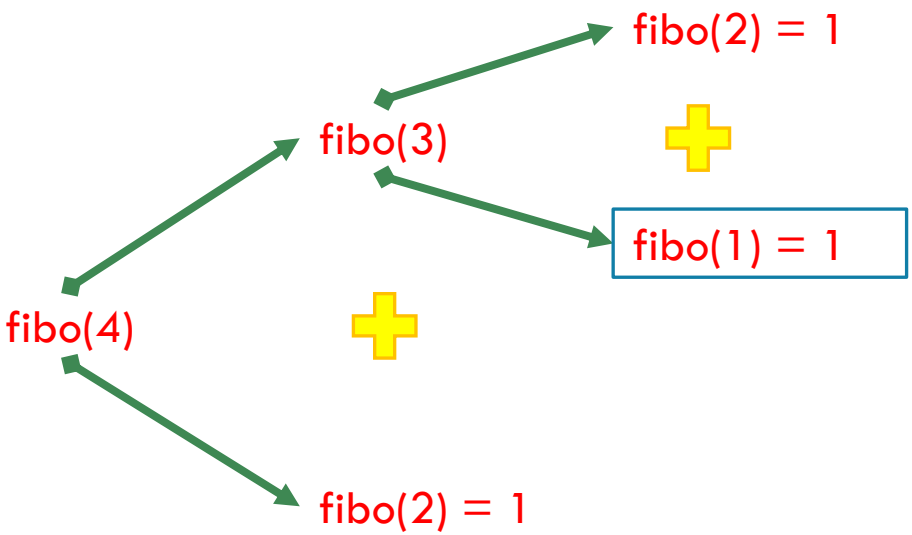
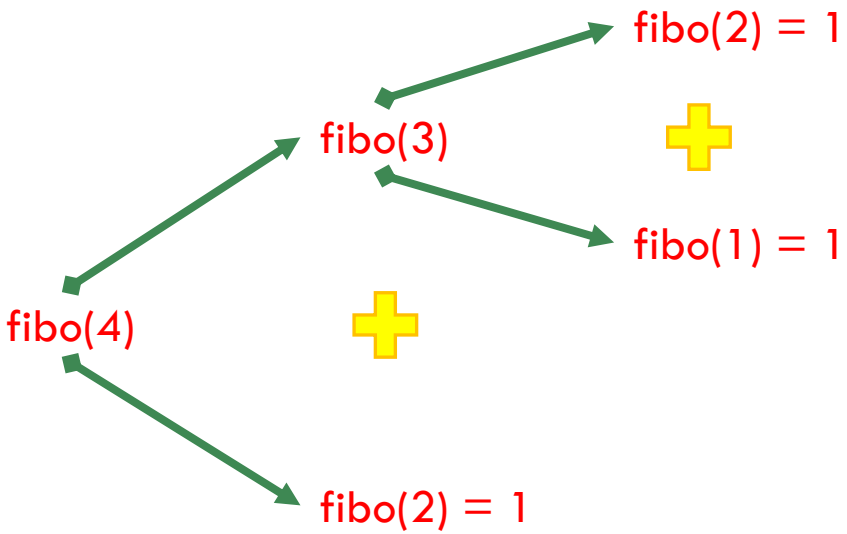


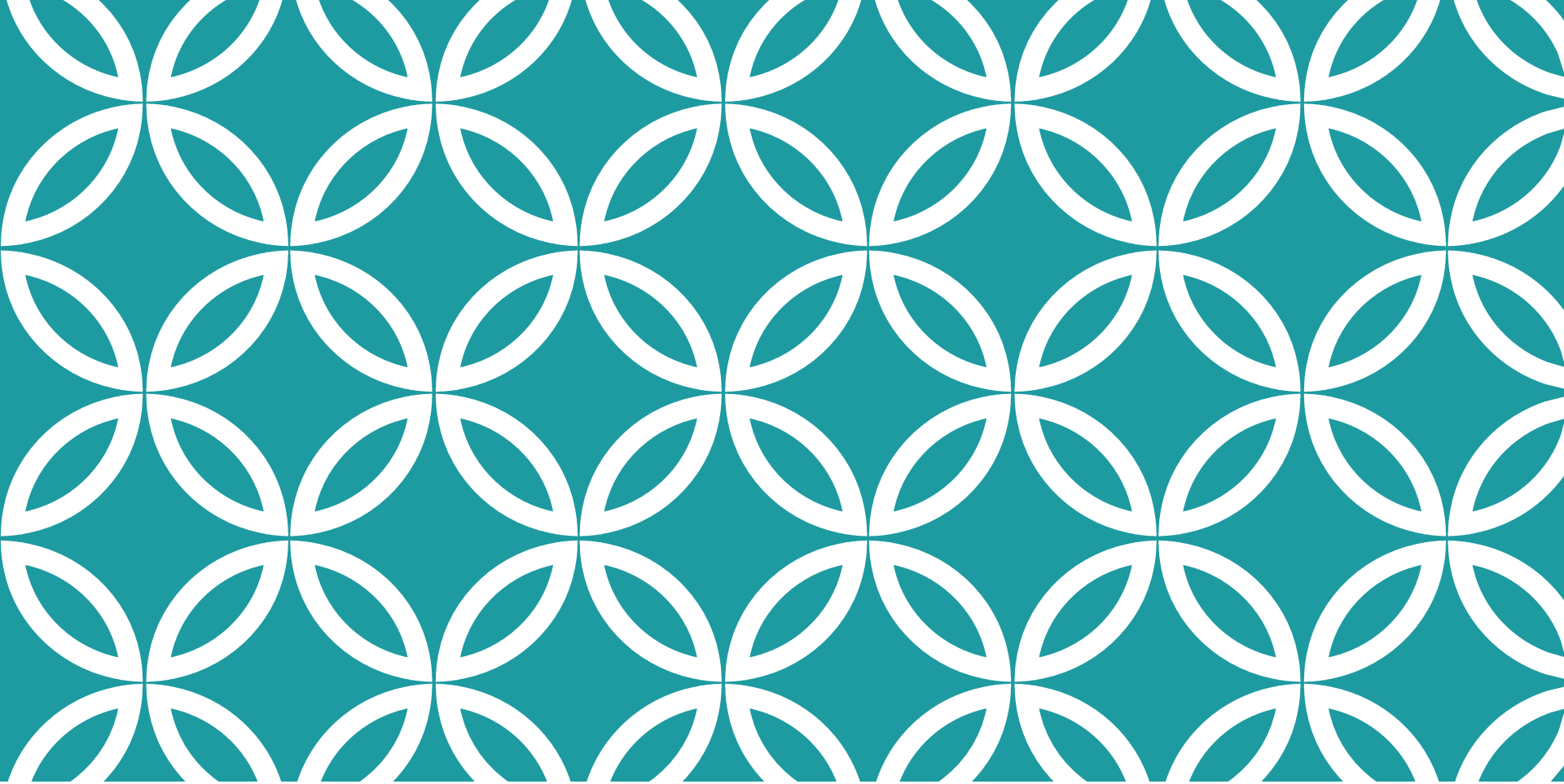
DIAGRAM POHON FIBO(4)



Nilai fibo(3) = fibo(2) + fibo(1) = 1 + 1 = 2

Nilai fibo(4) = fibo(3) + fibo(2) = 2 + 1 = 3

fibo(4) = 3



LATIHAN

BILANGAN FIBONACI KE-N (MODIFIKASI)

```
int fibo2(int n) {
	if (n == 1) {
		cout << "basis 1" << endl;
		return 1;
	} else if (n == 2) {
		cout << "basis 2" << endl;
		return 1;
	}
	else return fibo(n-1) + fibo(n-2);
}
```

Apa output program (yang tercetak ke layar) saat fungsi **fibo2** dipanggil dengan cara berikut? Gunakan diagram pohon rekursif untuk menentukan output program!

1. fibo2(4)
2. fibo2(5)

Silakan diskusikan di grup LINE jika ada kesulitan!

TERIMA KASIH