

Algoritma & Pemrograman

PROSEDUR/SUB RUTIN

Pengampu : Agus Priyanto, M.Kom



INSTITUT TEKNOLOGI TELKOM
Smart, Trustworthy, And Teamwork

Ilustrasi

```
#include <iostream>

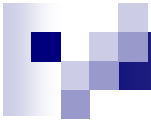
using namespace std;

int main()
{
    int tigaExpEmpat = 1;
    for (int i = 0; i < 4; i = i + 1) {
        tigaExpEmpat = tigaExpEmpat * 3;
    }
    cout << "3^4 adalah " << tigaExpEmpat << endl;

    return 0;
}
```



Copy-
paste
Coding



```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int tigaExpEmpat = 1;
```

```
    for (int i = 0; i < 4; i = i + 1) {
```

```
        tigaExpEmpat = tigaExpEmpat * 3;
```

```
    }
```

```
    cout << "3^4 adalah " << tigaExpEmpat << endl;
```

```
    int enamExpLima = 1;
```

```
    for (int i = 0; i < 5; i = i + 1) {
```

```
        enamExpLima = enamExpLima * 6;
```

```
    }
```

```
    cout << "6^5 adalah " << enamExpLima << endl;
```

```
    return 0;
```

```
}
```



Copy-
paste
Coding



Copy-
paste
Coding



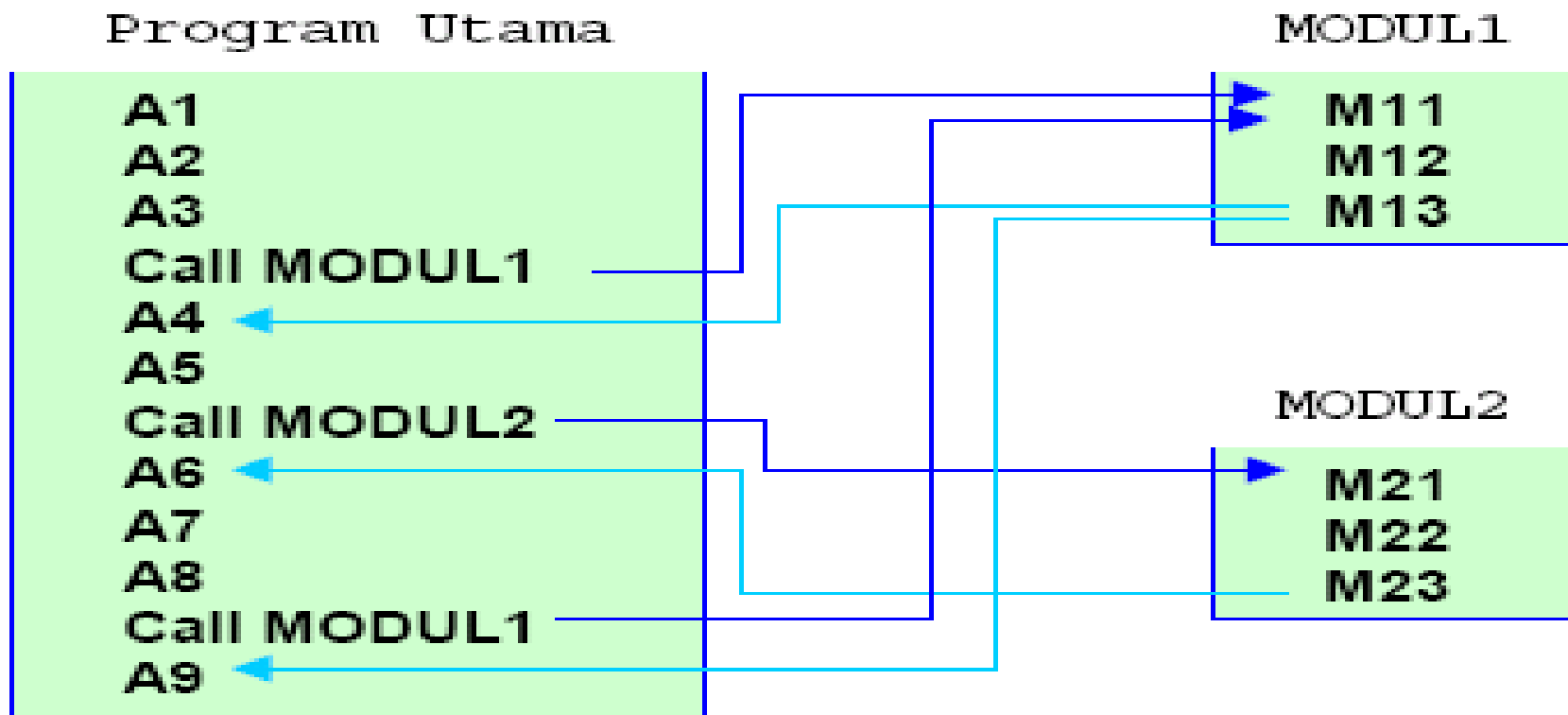
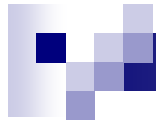
Pemrograman Modular

Teknik pemrograman modular :

- Program dipecah menjadi beberapa **subprogram** yang lebih kecil.
- Subprogram (**modul**, ***routine***) kadang independen dari program utama sehingga dapat dirancang tanpa mempertimbangkan konteks tempat ia digunakan, bahkan dapat dirancang orang lain.

Modularisasi memberikan dua keuntungan :

- Untuk aktivitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama secara berulang kali.
- Kemudahan dalam menulis dan menemukan kesalahan (*debug*) program.





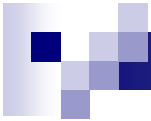
Definisi Prosedur

- **Prosedur** (***subroutine***) adalah modul program yang mengerjakan tugas/aktifitas yang spesifik dan menghasilkan suatu efek netto.
- **Efek netto** diketahui dengan membandingkan keadaan awal (sebelum) dan keadaan akhir (sesudah) pelaksanaan sebuah prosedur.
→ Pada setiap prosedur harus didefinisikan keadaan awal (K.Awal) dan keadaan akhir (K.Akhir).



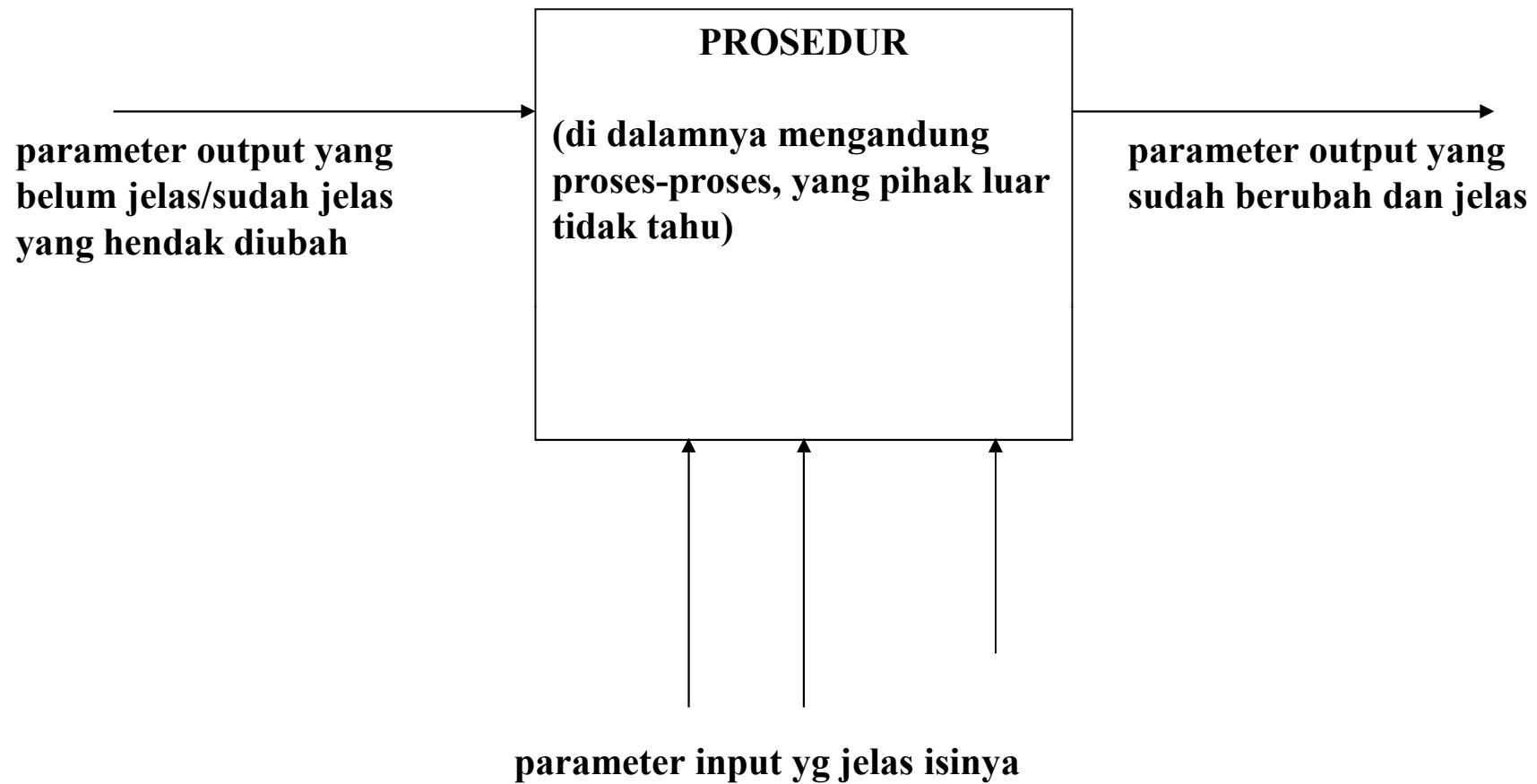
Definisi Prosedur

- **Prosedur** (***subroutine***) adalah modul program yang mengerjakan tugas/aktifitas yang spesifik dan menghasilkan suatu efek netto.
- Pada dasarnya *prosedur membutuhkan input dan output*.
- Input yang standar berasal dari piranti input standar (*standard input device*), yaitu **keyboard**.
- Output yang standar berasal dari piranti output standar (*standard output device*), yaitu **monitor**.
- Kedua piranti I/O standar ini dijembatani penggunaannya dalam C oleh library **stdio.h** (standard I/O), yang memperkenalkan **printf** untuk keperluan output dan **scanf** untuk keperluan input.

- 
- Berdasarkan darimana datangnya input dan output, prosedur dikategorikan ke dalam 4 macam:

input	output	prosedur
Std dev	Std dev	Naïve
Parameter	Std dev	Semi Naïve
Std dev	Parameter	Semi Naïve
Parameter	Parameter	Nett Effect

Hanya ada satu jenis prosedur yang direkomendasikan, yaitu prosedur yang menghasilkan efek netto (*nett effect procedure*)





Mendefinisikan Prosedur

- Struktur prosedur sama dengan struktur algoritma, yaitu : bagian judul, bagian deklarasi, dan badan prosedur.
- Setiap prosedur memiliki nama unik (sebaiknya diawali dengan kata kerja, mis : TUKAR, HITUNG_LUAS, CARI_MAKS, dll)



Mendefinisikan Prosedur (2)

Prosedur Naif:

```
void Hitung_Luas
```

```
{
```

```
    /*prosedur untuk menghitung luas segitiga dengan input t dan a */
```

```
}
```

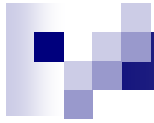
Prosedur Berparameter:

```
void Hitung_Luas(int *Luas, int tinggi, int alas)
```

```
{
```

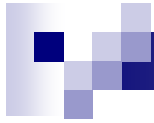
```
    /*prosedur untuk menghitung luas segitiga dengan input t dan a */
```

```
}
```



Pemanggilan Prosedur

- Prosedur diakses dengan memanggil namanya dari program pemanggil (program utama atau modul program lain).
- Dalam program pemanggil, *prototype* prosedur harus dideklarasikan supaya program pemanggil mengenal nama prosedur tersebut serta cara mengaksesnya.



Nama Lokal dan Nama Global

Nama Lokal :

- Nama-nama di bagian deklarasi prosedur.
- Bersifat lokal, hanya dapat digunakan di dalam prosedur yang melingkupinya.

Nama Global :

- Nama-nama yang dideklarasikan di program utama.
- Bersifat global, dapat digunakan di bagian manapun dalam program, baik di program utama maupun di prosedur.



Nama Lokal dan Nama Global

Menggunakan Nama Lokal atau Nama Global ?

- Jika nama (peubah, tipe, atau tetapan) digunakan di seluruh bagian program, maka harus dideklarasikan global.
- Jika nama tersebut hanya digunakan di dalam prosedur, maka sebaiknya dideklarasikan lokal.
- Usahakan menggunakan nama global sedikit mungkin!
→ menyulitkan dalam pencarian kesalahan (*debugging*)



Contoh : Prosedur tanpa parameter

```
#include <iostream>

using namespace std;

// Membuat prosedur tanpa parameter
void Tulis10Kali() {
    for (int C=0; C<10; C++) {
        cout<<"Aku sangat menyukai C++"<<endl;
    }
}

int main() {
    // Memanggil fungsi Tulis10Kali()
    // untuk dieksekusi
    Tulis10Kali();

    return 0;
}
```

Contoh : Prosedur dengan parameter

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

```
int N;  
float rata2;
```

Variabel global

```
void Hit_Rata2();
```

Deklarasi prosedur
(prototype)

```
Int main()
```

```
{
```

```
    printf("Banyak data: %d"); scanf("%d", &N);
```

```
    Hit_Rata2();
```

Pemanggilan prosedur

```
    printf("Rata-rata : %f\n", rata2);
```

```
}
```


Contoh

```
void Hit_Rata2()
{
    int i, bil, jumlah;
    jumlah=0;
    for(i=0;i<=N;i++)
    {
        printf("Masukkan angka ke-%d: ", i); scanf("%d", &bil);
        jumlah=jumlah+bil;
    }
    rata2=(float)jumlah/N;
}
```

Variabel lokal

Variabel Global

The diagram illustrates variable scope in the provided C code. Red circles are drawn around the following elements: the local variable declarations 'int i, bil, jumlah;', the global variable 'N' used in the for loop, and the global variable 'rata2' used in the calculation. A red arrow points from the first circle to the text 'Variabel lokal' (Local Variable). Three red arrows point from the other two circles to the text 'Variabel Global' (Global Variable).



Parameter

- Kebanyakan program memerlukan pertukaran informasi antara prosedur (atau fungsi) dan titik di mana ia dipanggil.
 - *Parameter berfungsi sebagai media komunikasi antara modul dengan program pemanggil.*
- Tiap item data ditransferkan antara parameter aktual dan parameter formal.
 - **Parameter aktual** : parameter yang disertakan pada waktu pemanggilan.
 - **Parameter formal** : parameter yang dideklarasikan di bagian *header* prosedur itu sendiri.



Parameter

- Saat prosedur dipanggil, parameter aktual menggantikan parameter formal.
- Tiap parameter aktual berpasangan dengan parameter formal yang bersesuaian.
- Aturan penting yang harus diperhatikan:
 - Jumlah parameter aktual pada pemanggilan prosedur harus sama dengan jumlah parameter formal pada deklarasi prosedurnya.
 - Tiap parameter aktual harus bertipe sama dengan tipe parameter formal yang bersesuaian.
 - Tiap parameter aktual harus diekspresikan dalam cara yang sesuai dengan parameter formal yang bersesuaian, bergantung pada jenis parameter formal.



Parameter

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

```
void HitungLuasLingkaran(float *hasil, float jejari);
```

```
void main()
```

```
{
```

```
    float Luas, radius;
```

```
    radius=9.8;
```

```
        HitungLuasLingkaran(&Luas, radius);
```

```
    printf("%f\n",Luas);
```

```
}
```

```
void HitungLuasLingkaran(float *hasil, float jejari)
```

```
{
```

```
    *hasil = 3.14*jejari*jejari;
```

```
}
```



Parameter

- Jenis parameter formal yang disertakan dalam prosedur:
 - **Parameter masukan (input parameter)**
→ nilainya berlaku sebagai masukan untuk prosedur.
 - **Parameter keluaran (output parameter)**
→ menampung nilai keluaran yang dihasilkan prosedur.
 - **Parameter masukan/keluaran (input/output parameter)**
→ berfungsi sebagai masukan sekaligus keluaran dari prosedur.



Parameter

Pada bahasa pemrograman :

- Parameter masukan
→ *value parameter, parameter by value*
- Parameter keluaran & parameter masukan/keluaran
→ *reference parameter, parameter by reference*



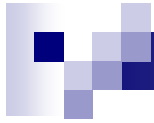
Parameter Masukan

- Nilai parameter aktual diisikan (*assign*) ke dalam parameter formal yang bersesuaian untuk digunakan dalam badan prosedur yang bersangkutan.
- Nilai yang dinyatakan oleh parameter masukan tidak dapat dikirim ke arah sebaliknya.
- Perubahan nilai parameter di dalam prosedur tidak mengubah nilai parameter aktual.
- Nama parameter aktual boleh berbeda dengan nama parameter formal.



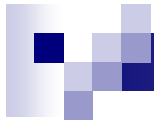
Parameter Keluaran

- Digunakan untuk menampung nilai keluaran dari prosedur yang akan digunakan oleh program pemanggil.
- Nama parameter aktual di program pemanggil akan menggantikan (*substitute*) nama parameter formal yang bersesuaian di prosedur.
- Setelah pemanggilan, parameter aktual akan berisi nilai keluaran dari prosedur yang dipanggil.



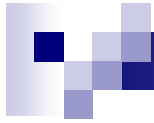
Parameter Masukan/Keluaran

- Nilai parameter aktual akan diisikan ke parameter formal yang bersesuaian untuk digunakan dalam prosedur, dan nama parameter aktual juga digunakan dalam prosedur (untuk menampung keluaran → nilainya berubah).



Perhatikan :

- Cara pendefinisian prosedur
- Penggunaan parameter masukan, keluaran, dan masukan/keluaran
- Parsing parameter
- Tipe data parameter aktual dan formal
- Nama lokal dan nama global



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

```
void Tukar(int *A, int *B)
```

```
void main()
```

```
{
```

```
    int nilai1, nilai2;
```

```
    scanf("%d", &nilai1);
```

```
    scanf("%d", &nilai2);
```

```
    printf("Nilai 1 sebelum pertukaran = %d", nilai1);
```

```
    printf("Nilai 2 sebelum pertukaran = %d", nilai2);
```

```
    Tukar(&nilai1, &nilai2);
```

```
    printf("Nilai 1 sebelum pertukaran = %d", nilai1);
```

```
    printf("Nilai 2 sebelum pertukaran = %d", nilai2);
```

```
}
```

```
void Tukar(int *A, int *B)
```

```
{
```

```
    int temp;
```

```
    temp = *A;
```

```
    *A = *B;
```

```
    *B = temp;
```

```
}
```



```
#include <iostream>
```

```
using namespace std;
```

```
void infoPTS ()
```

```
{  
    cout<<"ST3 Telkom ";  
}
```

```
int main()
```

```
{  
    infoPTS();  
    return 0;  
}
```

```
#include <iostream>
```

```
using namespace std;
```

```
double hitung_keliling_kotak (double panjang,  
    double lebar)
```

```
{  
    double keliling;  
    keliling = 2 * (panjang + lebar);  
    return keliling;  
}
```


```
int main()
```

```
{  
    double keliling, panjang, lebar;  
    panjang = 56.5;  
    lebar = 2;  
    keliling = hitung_keliling_kotak(panjang, lebar);  
    cout << "Keliling = " << keliling << "\n";  
    return 0;  
}
```



LATIHAN

1. Buat *prosedur* untuk menghitung nilai akhir seorang mahasiswa dengan parameter masukan berupa nilai_UTS, nilai_UAS, dan rata_kuis dengan rumus:
$$\text{nilai_akhir} = \text{nilai_UTS} * 35\% + \text{nilai_UAS} * 35\% + \text{rata_kuis} * 30\%$$
2. Buat prosedur untuk menghitung biaya SPP yang harus dibayar seorang mahasiswa. SPP dihitung dari jumlah SKS * biaya per SKS. Biaya per SKS untuk tiap angkatan berbeda:
 1. Angkatan 2010 keatas, biaya per SKS Rp. 100.000,-
 2. Angkatan 2005 – 2009, biaya per SKS Rp. 80.000,-
 3. Angkatan 2000 – 2004, biaya per SKS Rp. 60.000,-
 4. Angkatan 1999 kebawah, biaya per SKS Rp. 40.000,-Tentukan parameter-parameter prosedurnya



```
#include <iostream>

using namespace std;
double hitung_spp (int jml_sks, int angkatan)
{
    double biaya_spp;
    Int biaya_sks, sks;
    If (angkatan >= 2010) {
        biaya_sks = 100000;
    }
    else if((angkatan >= 2005) && (angkatan < 2010)){
        biaya_sks = 80000;
    }
    else if((angkatan >= 2000) && (angkatan < 2005)){
        biaya_sks = 60000;
    }
    Else{
        biaya_sks = 40000;
    }
    biaya_spp : sks * biaya_sks
    return biaya_spp;
}
```

