REVIEW ALGORITMA PEMROGRAMAN

“Prosedur dan Fungsi”

Indra Hermawan, S.Kom, M.Kom

indrah13@gmail.com / indra@nurulfikri.ac.id

No. 085217987034

Tujuan

- Mahasiswa memahami makna dan kegunaan subprogram dalam bentuk fungsi dan prosedur
- Mahasiswa dapat menggunakan notasi fungsi danprosedur dengan benar dan menggunakannya dlam program
- Mahasiswa dapat membuat program dengan menggunakan fungsi dan prosedur

Fakta (Kode yang Berulang)

- 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 dicopy 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 sam

Pengantar Prosedur dan Fungsi (contoh 1)

```
#include <iostream>
using namespace std;

int main() {
        string str1, str2;

        str1 = "Maya";
        cout << "Hello " << str1 << endl;


        cout << "Hello " << "Joko" << endl;

        cin >> str2;
        cout << "Hello " << str2 << endl;

        return 0;
}
```

Pengantar Prosedur dan Fungsi (contoh 1)

```
#include <iostream>
using namespace std;

int main() {
        string str1, str2;

        str1 = "Maya";
        CetakHello(str1);

        CetakHello("Joko");

        cin >> str2;
        CetakHello(str2);

        return 0;
}
```

DIGANTI DENGAN SUBPROGRAM (PROSEDUR)

Pengantar Prosedur dan Fungsi (contoh 2)

```
#include <iostream>
using namespace std;

int main() {
        const float PI = 3.14;
        float L, r, fx, x;

        r1 = 10;
        L = PI * r1 * r1;

        x = 10;
        fx = x * x;

        return 0;
}
```

Pengantar Prosedur dan Fungsi (contoh 2)

```
#include <iostream>
using namespace std;

int main() {
        const float PI = 3.14;
        float L, r, fx, x;

        r1 = 10;
        L = PI * FxKuadrat(r1);

        x = 10;
        fx = FxKuadrat(x);

        return 0;
}
```

DIGANTI DENGAN SUBPROGRAM (FUNGSI)

APA GUNANYA??

Subprogram

- " A set of instructions designed to perform a frequently used operation within a program "
- 2 (dua) jenis subprogram:
  - Fungsi
  - Prosedur

STT-NF

Bagian 1: Fungsi

Bagian 2: Prosedur

Definisi Fungsi

- Fungsi adalah sebuah transformasi akibat pemetaan suatu nilai (dari domain) ke nilai lain (dalam range) ➔ sama seperti di matematika
- Fungsi mempunyai nama dan sekelompok parameter formal (harga masukan yang diberi nama dan dijelaskan type-nya) serta memiliki hasil (dalam suatu type tertentu pula)
- Fungsi harus didefinisikan terlebih dahulu supaya dapat digunakan dalam bagian ALGORITMA program

Tahapan Penggunaan Fungsi

1. Mendefinisikan fungsi
   - Memberikan nama
   - Mendefinisikan parameter formal (parameter input)
   - Mendefinisikan type hasil
2. Merealisasikan fungsi
   - Membuat algoritma fungsi: memroses input hasil
3. Menggunakan fungsi dalam program utama
   - Memanggil fungsi dengan menggunakan parameter aktual

Contoh Fungsi

- Fungsi bernama f(x) memiliki satu parameter x didefinisikan sebagai $f(x) = x^2 + 3x - 5$
  - jika diberi harga x = 4 maka f(x) akan menghasilkan 23
  - jika diberi harga x = 1 maka f(x) akan menghasilkan -1
- Fungsi f(x,y) memiliki dua parameter x dan y, didefinisikan sebagai $f(x,y) = x^2 + 3xy - 5y - 1$
  - jika diberi harga x = 0 dan y = 0 maka f(x,y) akan menghasilkan -1
  - jika diberi harga x = 1 dan y = 0 maka f(x,y) akan menghasilkan 0

Mendefinisikan Fungsi

List parameter **input** / parameter **formal**

Parameter input boleh ada, boleh kosong

```
type_hasil nama_fungsi ( [type_parameter1   nm_parameter1,
                          type_parameter2   nm_parameter2,
                          ...
                          type_parametern   nm_parametern]);

//Jelaskan spesifikasi fungsi
```

```
int fxkuadrat (int x);
//Menghasilkan x * x + 3 * x - 5
```

Nama fungsi : fxkuadrat
Parameter masukan : 1 buah, yaitu x dengan type int
Hasil : bertype int

Mendefinisikan Fungsi (2)

- Parameter input boleh tidak ada (kosong)
  - Fungsi tidak membutuhkan apa-apa dari pemakainya untuk menghasilkan harga
- Jika list parameter input (parameter FORMAL) ada (tidak kosong, minimal satu nama), maka merupakan satu atau beberapa nama beserta type-nya satu atau beberapa nama beserta type-nya
- Fungsi harus menghasilkan suatu harga
  - Harga yang dihasilkan oleh fungsi harus memiliki suatu type tertentu

Merealisasikan Fungsi

```
type-hasil nama_fungsi ( [type-parameter1  nm-parameter1,
                          type-parameter2  nm-parameter2,
                          ...
                          type-parametern  nm-parametern] );
// Jelaskan spesifikasi fungsi
{
    // KAMUS LOKAL
    // Deklarasikan semua NAMA yang dipakai dalam algoritma
    // fungsi

    // ALGORITMA
    // Deretan teks algoritma :
    // pemberian harga, analisa kasus, pengulangan, dll.

    // Pengiriman harga di akhir fungsi, harus sesuai dengan
    // type hasil, caranya adalah:
    return (hasil);

}
```

Contoh Realisasi Fungsi

```
int fxkuadrat (int x)
//Menghasilkan x * x + 3 * x – 5
{
    //KAMUS LOKAL
    //tidak ada nama lokal yang perlu dideklarasikan

    //ALGORITMA
    return (x * x + 3 * x - 5);

}
```

Contoh Realisasi Fungsi - Alternatif

```
int fxkuadrat (int x)
//Menghasilkan x * x + 3 * x - 5
{
    //KAMUS LOKAL
    int hasil;

    //ALGORITMA
    hasil = x * x + 3 * x - 5;
    return (hasil);

}
```

Variable **hasil** hanya dikenal di dalam fungsi fxkuadrat, tidak di bagian program yang lain

Kode Fungsi Dalam Program

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;

DEKLARASI FUNGSI

// PROGRAM UTAMA
int main () {
        PEMAKAIAN FUNGSI
        return 0;
}

REALISASI FUNGSI
```

Dalam REALISASI FUNGSI bisa terdapat pemakaian fungsi lan

Kode Fungsi Dalam Program

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI FUNGSI
int fxkuadrat (int x);
//Menghasilkan x * x + 3 * x - 5

// PROGRAM UTAMA
int main () {
        // KAMUS
        int x, p, hasil;
        // ALGORITMA
        x = 3;
        hasil = fxkuadrat(x);
        p = 10 + fxkuadrat(5);
        cout << hasil << " " << p << " " << fxkuadrat(10) << endl;
        return 0;
}
//REALISASI FUNGSI
int fxkuadrat (int x)
//Menghasilkan x * x + 3 * x - 5
{  //KAMUS LOKAL
   //ALGORITMA
    return (x * x + 3 * x - 5);
}
```

CONTOH

OUTPUT:
13 45 125

Kode Fungsi Dalam Program

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI FUNGSI
int fxkuadrat (int x);
//Menghasilkan x * x + 3 * x - 5

int main () {
        // KAMUS
        int x, p, hasil;
        // ALGORITMA
        x = 3;
        hasil = fxkuadrat(x);
        p = 10 + fxkuadrat(hasil);
        cout << hasil << p << fxkuadrat(10) << endl;
        return 0;
}
{  //KAMUS LOKAL
   //ALGORITMA
    return (x * x + 3 * x - 5);
}
```

Fungsi selalu dipanggil pada **ruas kanan** dari suatu ekspresi

Semua nilai (termasuk variable) yang digunakan dlam pemanggilan fungsi disebut sebagai parameter **AKTUAL**

Pemanggilan fungsi dapat menjadi bagian dari ekspresi dengan hasil yang **harus sesuai dengan type hasil** fungsi

Pemanggilan Fungsi (1)

- Fungsi hanya dapat dipakai sebagai bagian ekspresi bukan merupakan suatu instruksi yang dipanggil independen
- Dalam ekspresi, fungsi hanya dapat diletakkan di ruas kanan • Saat pemanggilan terjadi korespondensi antara parameter input (formal) dengan parameter aktual sesuai dengan urutan penulisan dalam list-nama parameter input penulisan dalam list-nama parameter input
- List parameter aktual harus sama jumlah, urutan, dan typenya dengan list parameter input pada pendefinisian fungsinya

Pemanggilan Fungsi (2)

- Harga yang dihasilkan oleh fungsi dapat didefinisikan domainnya dengan lebih rinci
- Pada akhir dari eksekusi fungsi, harga yang dihasilkan oleh fungsi dikirimkan ke pemakainya
- Fungsi boleh dipakai oleh program utama, prosedur, atau fungsi lain

Contoh 1 : Fungsi Konversi

- Persoalan:
  - Tuliskanlah sebuah fungsi, yang mengkonversikan harga karakter angka (nol sampai dengan 9) menjadi harga numerik sesuai dengan karakter yang tertulis. Contoh:
    - '0' → 0
    - '8' → 8
    - '8' → 8
  - Berikan contoh pemakaian
- Spesifikasi :
  - Fungsi KarakterToInteger :
  - Domain : x : character ['0'..'9'])
  - Range : integer [0..9]
  - Proses : analisis kasus terhadap x, untuk setiap harga x diasosiasikan integer yang sesuai.

Contoh 1 : Fungsi Konversi

```
int KarakterToInteger (char x) {
// diberikan x berupa karakter, menghasilkan harga integer yang
// sesuai dengan penulisan pada karakter

        //Algoritma
        switch(x) {
                case '0' : return 0;
                case '1' : return 1;
                case '2' : return 2;
                case '3' : return 3;
                case '4' : return 4;
                case '5' : return 5;
                case '6' : return 6;
                case '7' : return 7;
                case '8' : return 8;
                case '9' : return 9;
        }
}
```

Contoh 1 : Fungsi Konversi

```
#include <iostream>
using namespace std;
int KarakterToInteger (char x);
// diberikan x berupa karakter, menghasilkan harga integer yang
// sesuai dengan penulisan pada karakter
int main () {
        //KAMUS
        char x;
        int y;
        //ALGORITMA
        cin >> x;
        y = KarakterToInteger(x);
        cout << y+1 << endl;
        return 0;
}
// Realisasi KarakterToInteger
...
```

Contoh 2: MAX2 dan MAX3

- Tuliskan fungsi MAX2, yang menerima masukan dua buah bilangan integer dan menghasilkan bilangan terbesar
  - Contoh: MAX2(1,2) ➔2
- Tuliskan fungsi MAX3 yang memanfaatkan fungsi MAX2. Fungsi MAX3 menerima input 3 bilangan integer dan menghasilkan bilangan terbesar
  - Contoh: MAX3(10,2,3) ➔10

Contoh 2: MAX2 dan MAX3

```
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;
        }
}

int Max3 (int a, int b, int c) {
// diberikan a, b, dan c, menghasilkan a jika a>=b dan a>=c,
// menghasilkan b jika b>=a dan b>=c, menghasilkan c jika c>=a dan
// c>=b
        //Algoritma
        return Max2(Max2(a,b),c);
}
```

Bagian 1: Fungsi

Bagian 2: Prosedur

Definisi Prosedur

- Prosedur adalah sederetan instruksi algoritmik yang diberi nama, dan akan menghasilkan efek netto yang terdefinisi
- Mendefinisikan prosedur berarti:
  - menentukan nama prosedur serta parameternya (jika ada)
  - menentukan nama prosedur serta parameternya (jika ada)
  - Mendefinisikan keadaan awal (initial state/IS) dan keadaan akhir (final state/FS)
- Cara penulisan spesifikasi
  - prosedur diberi nama dan
  - parameter formal (jika ada), yang diberi nama dan dijelaskan typenya

Tahapan Pemanfaatan Prosedur

1. Mendefinisikan prosedur
   - Memberikan nama
   - Mendefinisikan parameter formal (parameter input, output, input/output)
   - Mendefinisikan initial state (I.S.) dan final state (F.S.) – Mendefinisikan initial state (I.S.) dan final state (F.S.)
2. Merealisasikan prosedur
   - Membuat algoritma prosedur memroses agar I.S. dapat berubah menjadi F.S.
3. Menggunakan prosedur dalam program utama
   - Memanggil prosedur dengan menggunakan parameter aktual

Mendefinisikan Prosedur

Pada dasarnya adalah fungsi yang menghasilkan **void**

parameter **formal**

```
void nama_procedure ( [type_parameter1   nm_parameter1,
                       type_parameter2   nm_parameter2,
                          ...
                       type_parametern   nm_parametern]);

//Jelaskan spesifikasi prosedur
```

CONTOH DEFINISI

```
void HITUNG_V (int R1, int A1, int * V1);
// Prosedur untuk memproses tahanan & arus menjadi
// tegangan
// I.S: R1 dan A1 telah terdefinisi
// F.S: V1 terdefinisi dengan rumus V1=R1*A1
```

Merealisasikan Prosedur

```
void nama_prosedur ( [type-parameter1  nm-parameter1,
                      type-parameter2  nm-parameter2,
                           ...
                      type-parametern  nm-parametern] );
// Jelaskan spesifikasi prosedur
{
    // KAMUS LOKAL
    // Deklarasikan semua NAMA yang dipakai dalam algoritma
    // prosedur

    // ALGORITMA
    // deretan instruksi pemberian harga, input, output,
    // analisa kasus, pengulangan, pemanggilan fungsi dan
    // prosedur

}
```

TIDAK ADA RETURN DALAM PROSEDUR

Contoh Realisasi Prosedur

```
void HITUNG_V (int R1, int A1, int * V1) {
// Prosedur untuk memproses tahanan & arus menjadi tegangan
// I.S: R1 dan A1 telah terdefinisi
// F.S: V1 terdefinisi dengan rumus V1=R1*A1

      // Algoritma
      *V1 = R1 * A1;

}
```

Pemanggilan Prosedur

- Sebuah prosedur yang terdefinisi “disimpan” di tempat lain, dan ketika “dipanggil” dengan menyebutkan namanya “seakan-akan” teks yang tersimpan di tempat lain itu menggantikan teks tersimpan di tempat lain itu menggantikan teks pemanggilan
- Dengan konsep ini, maka I.S. dan F.S. dari prosedurlah yang menjamin bahwa eksekusi program akan menghasilkan efek netto yang diharapkan

Kode Prosedur dalam Program (1)

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;

DEKLARASI PROSEDUR

// PROGRAM UTAMA
int main () {
        PEMAKAIAN PROSEDUR
        return 0;
}

REALISASI PROSEDUR
```

Kode Prosedur dalam Program (2)

CONTOH

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1);
// Prosedur untuk memproses tahanan & arus menjadi tegangan
// I.S: R1 dan A1 telah terdefinisi
// F.S: V1 terdefinisi dengan rumus V1=R1*A1

// PROGRAM UTAMA
int main () {
        // KAMUS
        int r, a, vhasil;
        // ALGORITMA
        cin >>  r; cin >> a;
        HITUNG_V(r, a, &vhasil);
        cout << vhasil << endl;
        return 0;
}
// REALISASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1) {
        // Algoritma
        *V1 = R1 * A1;
}
```

Kode Prosedur dalam Program (3)

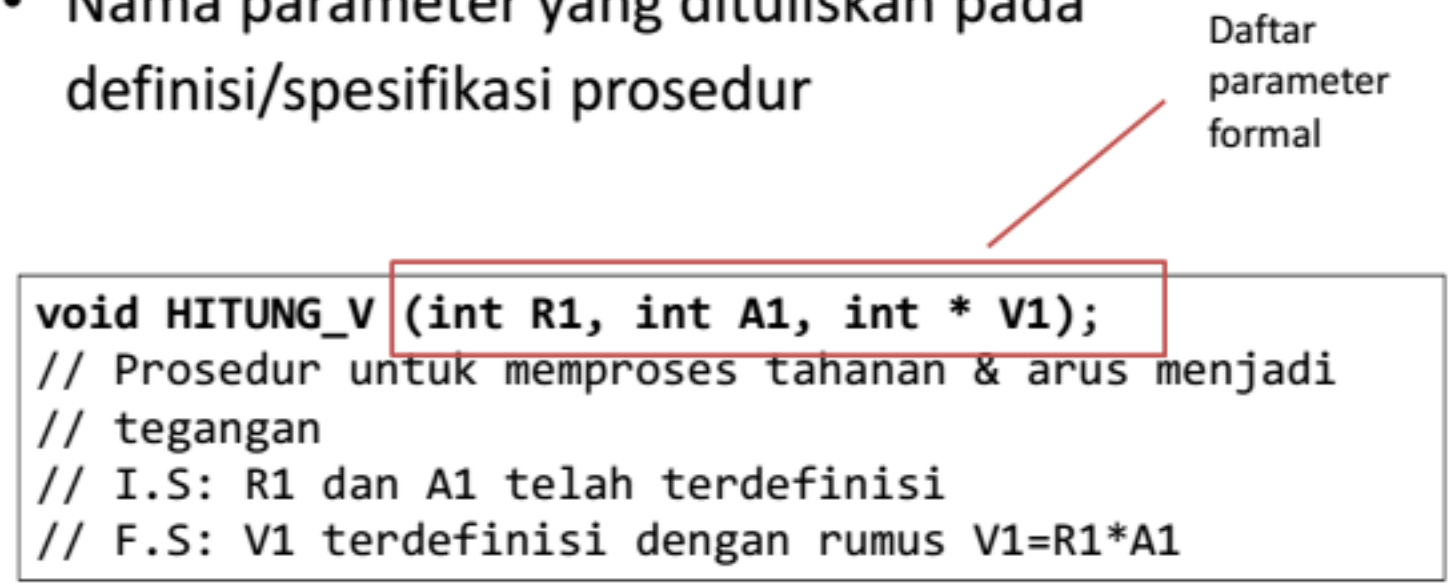
```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1)
// Prosedur untuk memproses tahanan &
// I.S: R1 dan A1 telah terdefinisi
// F.S: V1 terdefinisi dengan rumus V1=R1*A1

int main () {
        // KAMUS
        int r, a, vhasil;
        // ALGORITMA
        cin >>  r; cin >> a;
        HITUNG_V(r, a, &vhasil);
        cout << vhasil << endl;
        return 0;
}

        *V1 = R1 * A1;
}
```

Semua nilai (termasuk variable) yang digunakan dlam pemanggilan fungsi disebut sebagai parameter **AKTUAL**

Prosedur selalu dipanggil sebagai sebuah **instruksi** tersendiri

Parameter Formal

- Nama parameter yang dituliskan pada definisi/spesifikasi prosedur



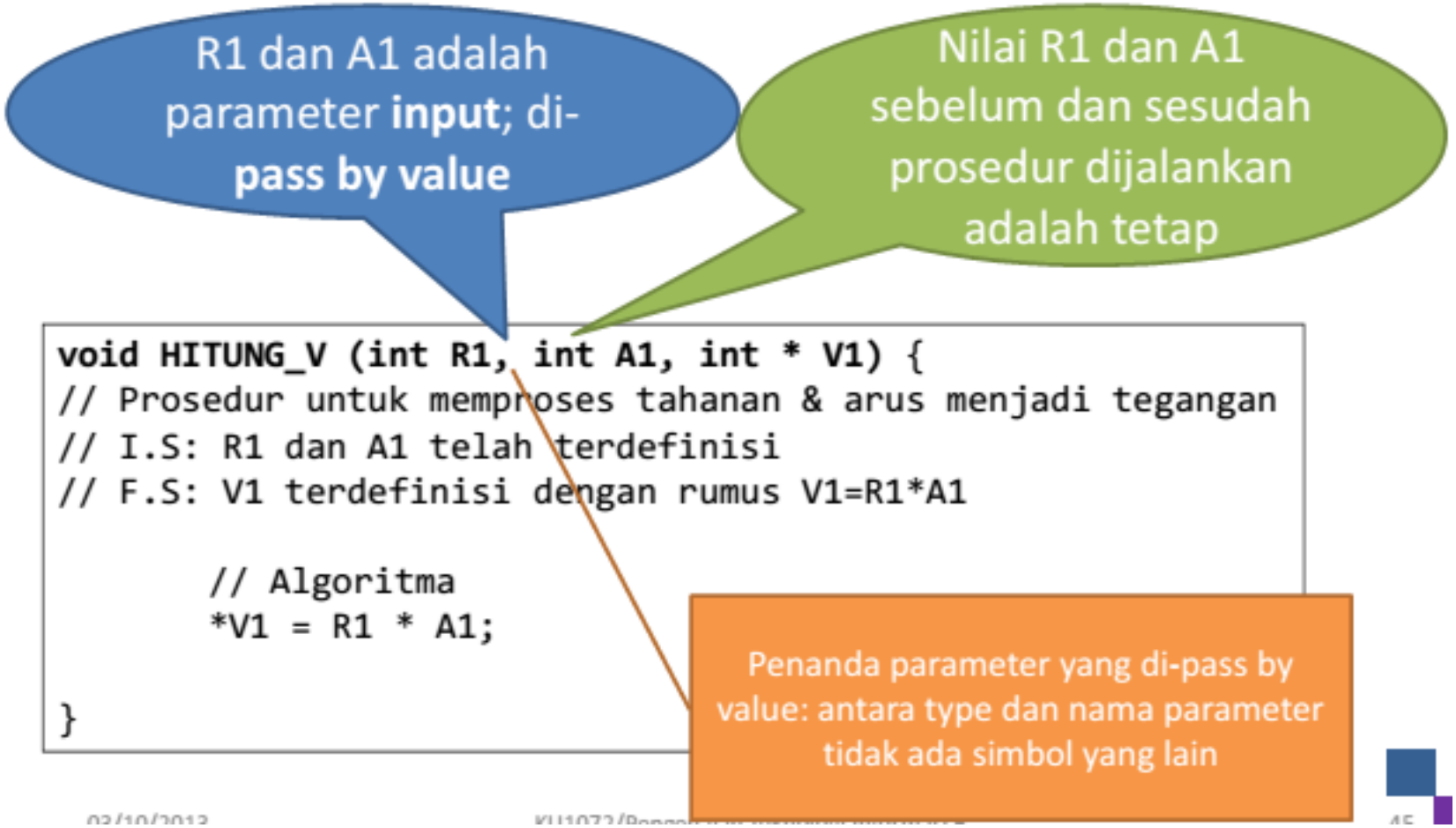
Parameter Formal

- Jenis-jenis parameter formal:
  - Parameter Input: parameter yang diperlukan prosedur sebagai masukan untuk melakukan aksi yang efektif ➔ **passing parameter by value**
  - Parameter Output: parameter yang nilainya akan dihasilkan oleh prosedur ➔ **passing parameter by reference**
  - Parameter Input/Output: parameter yang nilainya diperlukan prosedur sebagai masukan untuk melakukan aksi, dan pada akhir prosedur akan dihasilkan nilai yang baru ➔ **passing parameter by reference**

Parameter Aktual

- Parameter Aktual: nama-nama informasi yang dipakai ketika prosedur itu dipakai (“dipanggil”).
- Parameter aktual dapat berupa nama atau harga, tetapi harus berupa nama jika parameter tersebut adalah parameter output (karena hasilnya akan disimpan dalam nama tersebut)
- Pada saat pemanggilan prosedur terjadi asosiasi antara parameter formal dengan parameter aktual
- Asosiasi dilakukan dengan cara “by position”, urutan nama parameter aktual akan diasosiasikan sesuai dengan urutan parameter formal. Karena itu, type harus kompatibel.

Parameter Input
Parameter Passing By Value



Parameter Input
Parameter Passing By Value

- Parameter yang di-pass by value:
- Parameter formal: antara type dan nama parameter diberi tidak ada simbol apa pun tidak ada simbol apa pun
- Parameter aktual:
  - Harus sudah terdefinisi nilainya, sebelum dipanggil dengan prosedur
  - Nilainya tidak berubah sebelum dan sesudah digunakan dalam prosedur

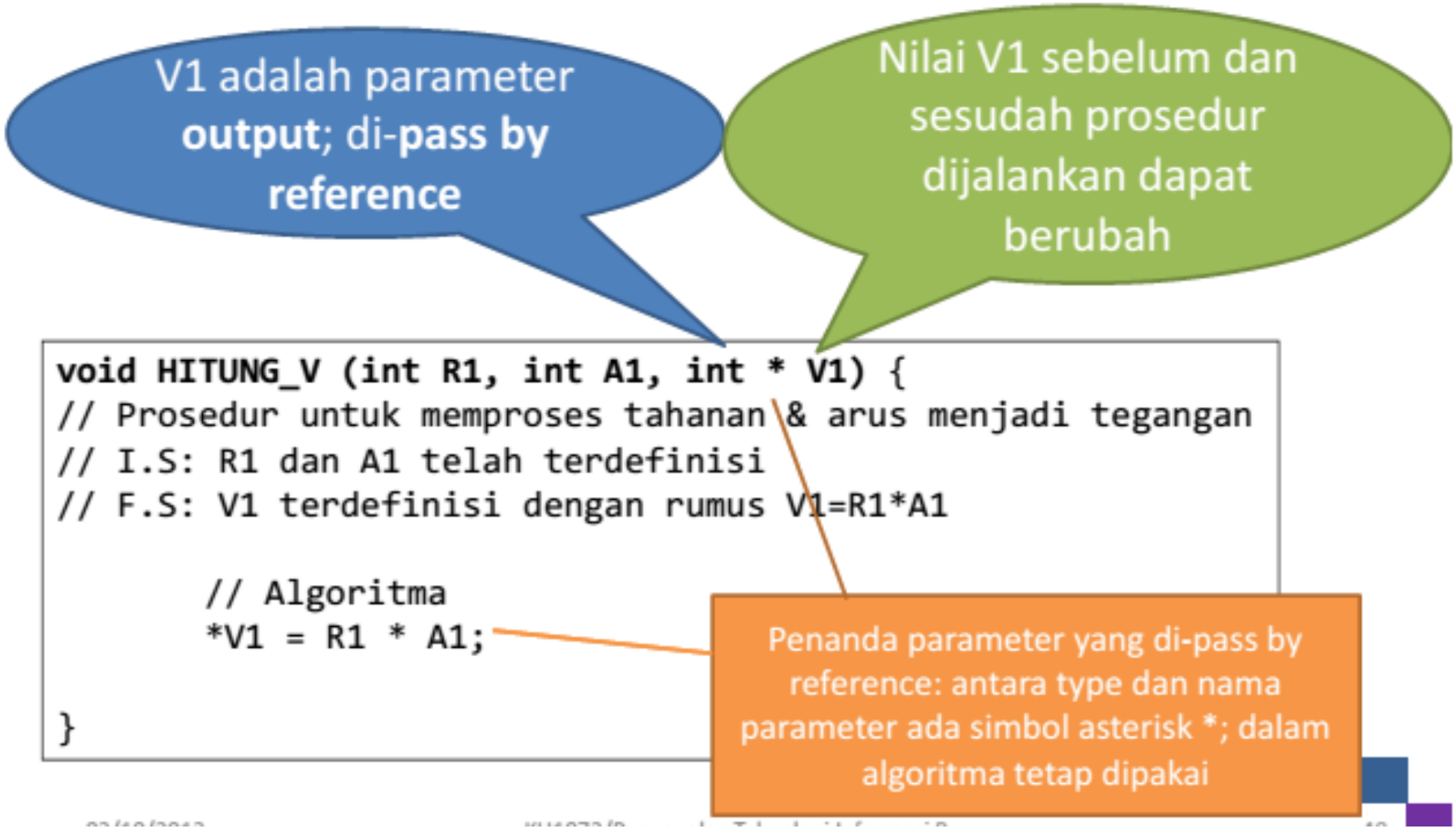
Parameter Input
Parameter Passing By Value

CONTOH

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1);
// ...

// PROGRAM UTAMA
int main () {
        // KAMUS
        int r, a, vhasil;
        // ALGORITMA
        cin >>  r; cin >> a;
        cout << r << " " << a << endl; // 1. sebelum pemanggilan
        HITUNG_V(r, a, &vhasil);
        cout << r << " " << a << endl; // 2. sesudah pemanggilan
        cout << hasil << endl;
        return 0;
}
// REALISASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1) {
        ...
}
```

Nilai r dan a sebelum dan sesudah pemanggilan prosedur adalah sama

Parameter output atau input/output
Passing parameter by reference



Parameter output atau input/output
Passing parameter by reference

- Parameter yang di-pass by reference:
  - Parameter formal: antara type dan nama parameter diberi tanda asterisk * digunakan pada algoritma tanda asterisk * digunakan pada algoritma
  - Parameter aktual:
    - Jika dipakai hanya sebagai output: nilai parameter aktual tidak harus terdefinisi
    - Jika dipakai sebagai input dan sekaligus output: nilai parameter aktual harus didefinisikan terlebih dahulu
  - Pemanggilan parameter aktual : menggunakan tanda & sebelum nama paramete

Parameter output atau input/output
Passing parameter by reference

CONTOH

```
//Judul dan spesifikasi program
#include <iostream>
using namespace std;
// DEKLARASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1);
// ...

// PROGRAM UTAMA
int main () {
        // KAMUS
        int r, a, vhasil;

        // ALGORITMA
        cin >>  r; cin >> a;
        // cout << vhasil << endl; // 1. sebelum pemanggilan, illegal!!
        HITUNG_V(r, a, &vhasil);
        cout << hasil << endl; // 2. sesudah pemanggilan

        return 0;
}
// REALISASI PROSEDUR
void HITUNG_V (int R1, int A1, int * V1) {
        ...
}
```

Sebelum pemanggilan prosedur, **vhasil** tidak terdefinisi. Sesudah pemanggilan prosedur, **vhasil** terdefinisi nilainya.

Prosedur VS Fungsi

- Menghitung Tegangan (V) dengan rumus R * A

```
int HITUNG_V (int R1, int A1) {
// menghasilkan tegangan sebagai hasil kali R dan A
        // Algoritma
        return (R1 * A1);
}
```

FUNGSI

```
void HITUNG_V (int R1, int A1, int * V1) {
// Prosedur untuk memproses tahanan & arus menjadi tegangan
// I.S: R1 dan A1 telah terdefinisi
// F.S: V1 terdefinisi dengan rumus V1=R1*A1
        // Algoritma
        *V1 = R1 * A1;
}
```

PROSEDUR

Latihan