Subprogram (dalam Bahasa C++)

Tim Penyusun Materi PTI-B

KU1072/Pengenalan Teknologi Informasi B
Tahap Tahun Pertama Bersama
Institut Teknologi Bandung

Tujuan

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

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;
}
```

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)

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;
}
```

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)

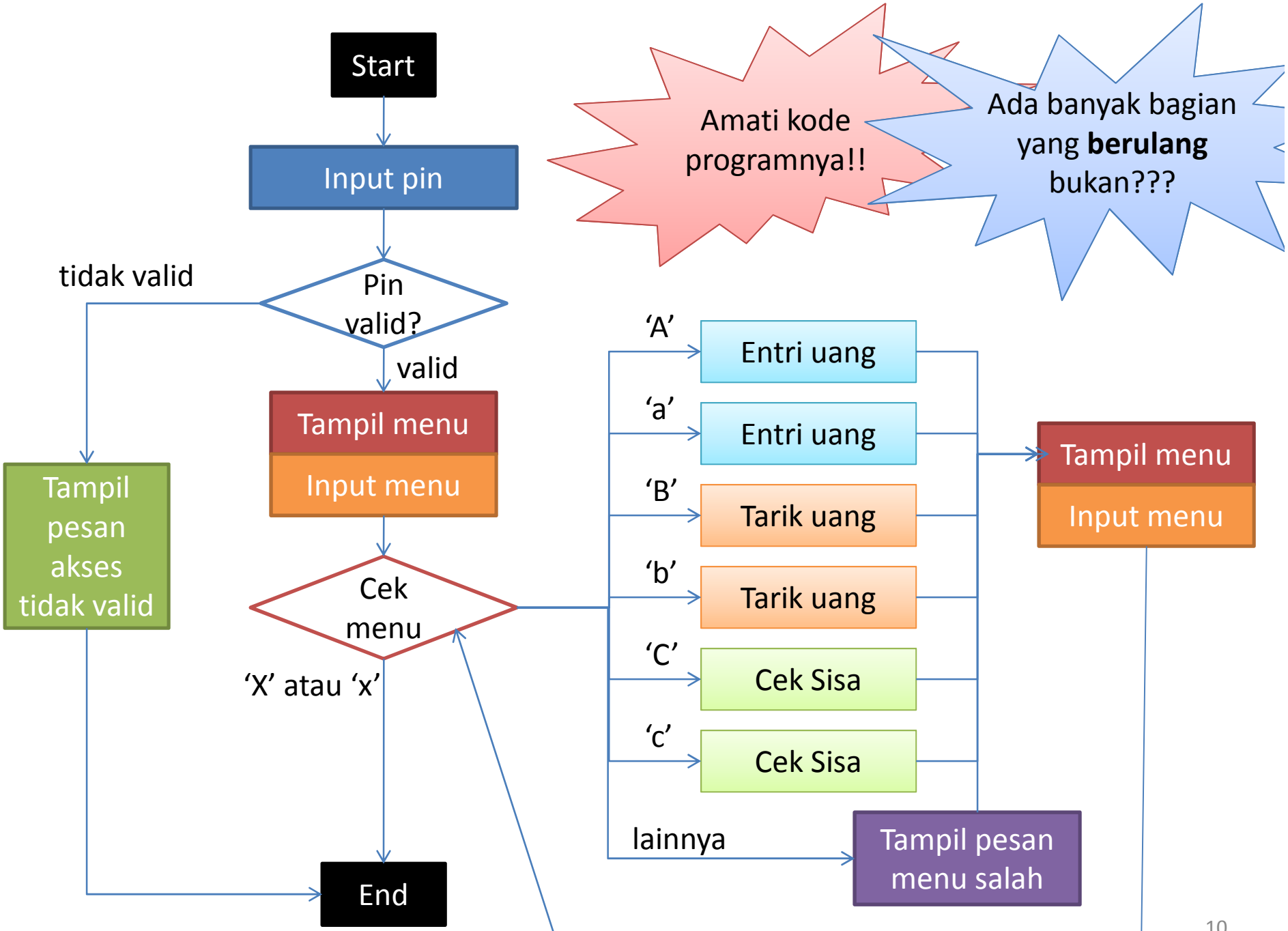
APA GUNANYA??

ATM-SS

- Sebuah mesin ATM Super Sederhana (ATM-SS) milik BSSJ (Bank Super Sederhana Juga) memiliki cash dispenser yang digunakan untuk menampung uang
- ATM-SS hanya dapat digunakan untuk memasukkan uang ke dalam cash dispenser, menarik uang dari cash dispenser tersebut, dan menampilkan sisa uang yang ada di cash dispenser
- Untuk mengakses ATM-SS, pengguna harus memasukkan pin. Pin yang valid adalah nilai integer di antara 101 dan 1000. Jika pin tidak valid, maka program langsung selesai.

ATM-SS

- Jika pin valid, pengguna dapat memilih beberapa menu sbb:
  - Jika memasukkan menu huruf ‘A’ atau ‘a’ : memasukkan sejumlah uang ke cash dispenser
  - Jika memasukkan menu huruf ‘B’ atau ‘b’ : menarik sejumlah uang dari cash dispenser
  - Jika memasukkan menu huruf ‘C’ atau ‘c’ : menampilkan jumlah uang yang ada di cash dispenser
  - Jika memasukkan menu huruf ‘X’ atau ‘x’ : keluar dari program
  - Jika memasukkan huruf lain, program akan menampilkan pesan “Bukan pilihan menu yang benar”


Start
Input pin
Amati kode programnya!!
Ada banyak bagian yang **berulang** bukan???
tidak valid
Pin valid?
valid
Tampil menu
Input menu
Tampil pesan akses tidak valid
Cek menu
'X' atau 'x'
End
'A'
Entri uang
'a'
Entri uang
'B'
Tarik uang
'b'
Tarik uang
'C'
Cek Sisa
'c'
Cek Sisa
lainnya
Tampil pesan menu salah
Tampil menu
Input menu

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 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

Subprogram

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

FUNGSI

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 Memanfaatkan 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 (1)

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
- 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

CONTOH

```
//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);
}
```

OUTPUT:
13 45 125

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

- **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
- List parameter aktual harus sama jumlah, urutan, dan type-nya dengan list parameter input pada pendefinisian fungsinya
- 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' $\rightarrow$ 0
    - '8' $\rightarrow$ 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.

```
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;
        }
}
```

```
#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

```
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);
}
```

PROSEDUR

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

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)
  - 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 Memanfaatkan Prosedur

1. Mendefinisikan prosedur
   - Memberikan nama
   - Mendefinisikan parameter formal (parameter input, output, input/output)
   - 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 (1)

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 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

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

DEKLARASI PROSEDUR

```
// PROGRAM UTAMA
int main () {
```

PEMAKAIAN PROSEDUR

```
        return 0;
}
```

REALISASI PROSEDUR

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;
}
```

```
//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

Daftar parameter formal

```
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
```

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

Passing parameter by value

R1 dan A1 adalah parameter **input**; di-**pass by value**

Nilai R1 dan A1 sebelum dan sesudah prosedur dijalankan adalah tetap

```
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;

}
```

Penanda parameter yang di-pass by value: antara type dan nama parameter tidak ada simbol yang lain

Parameter input
Passing parameter by value

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

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

V1 adalah parameter **output**; di-**pass by reference**

Nilai V1 sebelum dan sesudah prosedur dijalankan dapat berubah

```
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;

}
```

Penanda parameter yang di-pass by reference: antara type dan nama parameter ada simbol asterisk *; dalam algoritma tetap dipakai

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
  - 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 parameter

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.

Contoh 1: Hello, X

- Buatlah sebuah prosedur yang menerima sebuah string, misalnya str, dan menghasilkan tampilan:

  Hello, **str**

- Buatlah contoh pemakaiannya di program utama

```
void CetakHello(string s) {
//Menerima s dan
//mencetak "Hello, s" ke layar
//I.S.: s terdefinisi
//F.S.: tercetak "Hello, s" di
//layar

   //Algoritma
   cout << "Hello " << s << endl;
}
```

s adalah parameter input, passed by value

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

void CetakHello(string s);
...

int main() {
        string str1, str2, str3;

        str1 = "Maya";
        CetakHello(str1);
        CetakHello("Joko");
        cin >> str3;
        CetakHello(str3);

        return 0;
}
//Realisasi Prosedur
void CetakHello(string s) {
        ...
}
```

Contoh 2: Prosedur TUKAR

Buatlah prosedur untuk menukar dua harga yang disimpan dalam dua nama a dan b

Initial State (I.S.) :

diberikan nilai integer a = A dan b = B

Final State (F.S.) :

a = B dan b = A

```
void Tukar(int * a, int * b) {
//I.S.: a, b terdefinisi
//F.S.: nilai a dan b ditukar

    //Kamus Lokal
    int temp;

    //Algoritma
    temp = *a;
    *a = *b;
    *b = temp;
}
```

a dan **b** adalah contoh parameter **input / output**; di-**pass by reference**
Nilai **a** dan **b** harus terdefinisi terlebih dahulu, tetapi diubah dalam program

Contoh pemakaian nama-nama secara lokal
temp hanya dikenal di prosedur Tukar

Contoh 3: Putar 3 Bilangan

- Dengan memanfaatkan prosedur Tukar di contoh 2, buatlah sebuah program utama yang digunakan untuk memutar 3 buah bilangan, misalnya A=a, B=b, C=c sehingga A=c, B=a, C=b

  Ketiga bilangan dibaca dari keyboard

- Contoh:

  A = 3; B = 2; C = 1

  ➔ ditukar menjadi :

  A = 1; B = 3; C = 2

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

void Tukar(int * a, int * b);
...

int main () {
//Kamus
   int a1, b1, c1;
//Algoritma
   cin >> a1 >> b1 >> c1;
   cout << "Sebelum ditukar : " << endl;
   cout << "a = " << a1 << "; b = " << b1 << "; c = " << c1 << endl;
   Tukar(&a1, &b1);
   Tukar(&a1, &c1);
   cout << "Sesudah ditukar : " << endl;
   cout << "a = " << a1 << "; b = " << b1 << "; c = " << c1 << endl;

   return 0;
}
// Realisasi Prosedur ...
```

LATIHAN SOAL

Soal-1a:

- Diketahui fungsi CiriBilangan seperti di samping:

```
int CiriBilangan(int bil) {
// Menghasilkan:
// 0 : jika bil adalah 0
// 1 : jika bil adalah positif
// -1 : jika bil adalah negatif
        //Kamus Lokal
        int hasil;
        //Algoritma
        if (a == 0) {
                hasil = 0;
        } else if (bil > 0) {
                hasil = 1;
        } else { // bil < 0
                hasil = -1;
        }
        return hasil;
}
```

Soal-1b:

- Apakah hasil dari potongan program berikut?

```
int X = 0;
cout << CiriBilangan(X) << endl;
```

0

```
cout << CiriBilangan(1000) << endl;
```

1

```
int bil, c;

bil = -222;
c = CiriBilangan(bil);
cout << c << endl;
```

-1

Soal 2

- Buatlah sebuah fungsi yang digunakan untuk menerima sebuah bilangan riil (float) yang merupakan jari-jari sebuah lingkaran dan menghasilkan luas lingkaran berdasarkan jari-jari tersebut

Alternatif Solusi Soal-2:

```
float LuasLingkaran (float r) {
// Menghasilkan luas lingkaran berdasarkan r (radius)

        //Kamus Lokal
        const float PI = 3.14;

        //Algoritma
        return PI * r * r;
}
```

Konstanta **PI** hanya dikenal di dalam prosedur LuasLingkaran

Soal-3

- Bagaimana jika Anda harus membuat 2 buah fungsi dari masukan jari-jari:
  - Menghitung keliling lingkaran
  - Menghitung luas lingkaran
- Bagaimana agar konstanta PI tidak perlu didefinisikan berulang-ulang di setiap fungsi?

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

//Konstanta Global
const float PI = 3.14;

//Spesifikasi Fungsi dan Prosedur
float KelLingkaran(float r); //Menghitung keliling lingkaran
float LuasLingkaran(float r); //Menghitung luas lingkaran

//PROGRAM UTAMA
int main () {
//Kamus
   ...
//Algoritma
   ...
   return 0;
}

// Realisasi Fungsi dan Prosedur
float KelLingkaran(float r) { ... }
float LuasLingkaran(float r) { ... }
```

Konstanta dideklarasikan **global,** sehingga bisa digunakan oleh blok program manapun termasuk fungsi/prosedur

Soal-4

- Implementasikan sebuah prosedur yang digunakan untuk:
  - Membaca sejumlah N bilangan integer dari keyboard ➔ N merupakan parameter input untuk prosedur diasumsikan N > 0
  - Menghasilkan penjumlahan seluruh bilangan yang dimasukkan ➔ menjadi parameter output
- Definisi dari prosedur tersebut adalah sebagai berikut:

```
void BacaDanJumlahBil (int N, int * Sum);
// Membaca N bilangan integer dari keyboard dan menghasilkan
// Sum yang merupakan jumlah dari seluruh bilangan yang
// dimasukkan
// I.S. : N terdefinisi, N > 0
// F.S. : Sum merupakan jumlah N bilangan
```

Alternatif Solusi Soal 4

```
void BacaDanJumlahBil (int N, int * Sum) {
// Membaca N bilangan integer dari keyboard dan menghasilkan Sum
// yang merupakan jumlah dari seluruh bilangan yang dimasukkan
// I.S. : N terdefinisi, N > 0
// F.S. : Sum merupakan jumlah N bilangan
        // KAMUS LOKAL
        int x, i;

        // ALGORITMA
        *Sum = 0;
        for (i = 1; i <= N; i++) {
                cin >> x;
                *Sum = *Sum + x;
        }
}
```