Universitas Padjadjaran

FUNGSI

Mira Suryani, S.Pd., M.Kom

S-1 Teknik Informatika

Jatinangor, 25 Oktober 2018

TUJUAN INSTRUKSIONAL KHUSUS

- Mahasiswa mampu membuat program yang terstruktur dan modular menggunakan fungsi baik fungsi dengan pengembalian nilai ataupun fungsi void.
- Mahasiswa mampu menggunakan dan membedakan passing by value dan passing by reference

DESKRIPSI MATERI PERKULIAHAN

- Pada bagian ini akan dibahas topik-topik tentang :
  - Fungsi User Defined
  - Fungsi void
  - Fungsi Boolean
  - Passing by value dan passing by refernce
- Dalam pembahasan diberikan contoh-contoh program yang relevan dengan model pengembangan program yang modular.

PENGERTIAN FUNGSI

- Fungsi adalah objek (**bagian program/rutin**) yang mengerjakan suatu tugas tertentu dan digunakan untuk **memodularkan program** dengan suatu **ciri mengembalikan suatu nilai (return value)**
- Fungsi dapat digunakan untuk menghindari penulisan yang sama yang ditulis berulang-ulang.
- **Semua variables** yang dideklarasikan **dalam fungsi** merupakan **variable lokal**, yang hanya diketahui dalam fungsi bersangkutan
- Fungsi bisa memiliki **parameter** yang menyediakan **komunikasi antara fungsi dengan bagian yang memanggil fungsi** tersebut (**Parameter formal** dan **Parameter aktual).**
- **Parameters bersifat local**.

USER DEFINE FUNCTION

- Berbagai macam fungsi yang disediakan oleh C++ (fungsi pustaka ***/ library function***) seperti INT_MAX yang didefinisikan dalam <limits.h> atau pow(x,y) dalam <math.h> dan banyak yang lain, masih belum cukup untuk menjadi alat bantu dalam pengembangan program.
- Oleh sebab itu maka programmer bisa membuat fungsi mereka sendiri yang disebut dengan ***user defined function***

Penulisan Fungsi

Bentuk umum penulisan fungsi yang didefinisikan pengguna:

```
TipeHasil namaFungsi(tipe1 par1, tipe2 par2 …) {
   // kamus data local
        …….
   // Isi Fungsi
   return (hasil);   // pengembalian hasil ke pemanggil
}
```

Pemanggilan :

```
hsl = namaFungsi (akt1, akt2 …)
```

Atau

```
cout<< namaFungsi (akt1, akt2 …)
```

- Antara **parameter aktual** (**yang ada di pemanggil**) dengan **parameter formal** (**yang ada di fungsi**) harus bersesuaian yaitu **memiliki tipe yang sama dan banyaknya juga sama**.

DEKLARASI & PENDEFINISIAN FUNGSI

- Ada 2 model pendeklarasian dan pendefinisian fungsi yaitu :
  1. Definisi lengkap dari fungsi diletakkan **diatas** dari **main program**.
  2. Meletakkan **header dari fungsi diatas main program** (deklarasi / protoype) dan **definisi lengkapnya** diletakkan **di bawah main program**.

contoh : *Program untuk mencari kuadrat bilangan*

- **Model 1:**

```
// Definisi Fungsi lengkap di atas main program
int pangkat2(int x){
    return ( x * x);
}

main() {
    int  n;
     cin >> n;
     cout << pangkat2(n) << endl;     // Testing fungsi
}
```

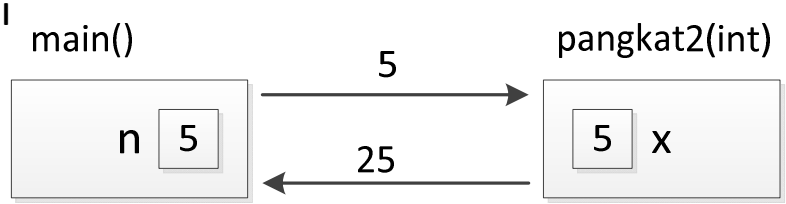
Model 2

```
int pangkat2(int x);          //Header / deklarasi fungsi

main() {
    int n;
    cin >> n;
    cout << pangkat2(n) << endl;  //Testing fungsi
}

int pangkat2(int x) {          // definisi fungsi di bawah main
    return ( x * x);
}
```

- Gambaran Pemanggilan Fungsi



FUNGSI VOID

- **Procedure** atau **subroutine** pada bahasa lain diimplementasikan pada C++ dengan menjadikannya sebagai fungsi dengan menempatkan kata kunci void di depan nama procedure yang dibuat.
- Tidak diperlukan adanya pengembalian nilai dari fungsi, sehingga perintah return tidak dimasukkan.
- **Penulisan Fungsi void**

```
void NamaFungsi(tipe1 par1, tipe2 par2 …) {
    // kamus data local
        …….
    // Isi Fungsi void
        …….
}
```

Pemanggilan

```
NamaFungsi(akt1, akt2, …..)
```

PENGIRIMAN SECARA NILAI (PASSING BY VALUE)

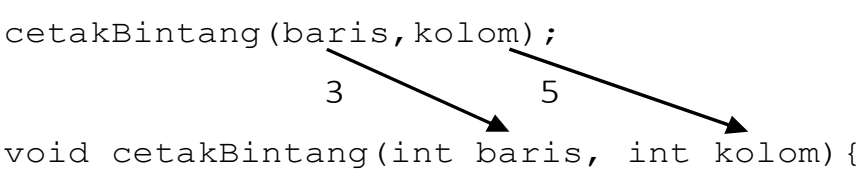
- Penulisan parameter di fungsi (parameter formal) dengan bentuk :
  - *tipe parameter*
- Yang dikirimkan ke fungsi adalah **nilai dari datanya**, **bukan alamat memori letak** dari datanya sehingga fungsi hanya bisa membaca nilai dari yang memanggil (*read only*).
- Fungsi yang menerima kiriman nilai ini akan menyimpannya di alamat yang terpisah dari nilai aslinya (membuat salinan)
- Perubahan nilai di fungsi tidak akan merubah nilai asli di bagian program yang memanggil walaupun namanya sama ataupun berbeda
- Pengiriman nilai oleh pemanggil fungsi (Parameter aktual) bisa berupa konstanta, variable, dan ekspresi

Contoh : Fungsi cetak bintang sebanyak m baris dan n kolom

```
void cetakBintang(int baris, int kolom){
    cout << ”Pencetakan bintang : ” << endl;
    for (int i=1; i<=baris; i++) {
            for (int j=1; j<=kolom; j++) {
                 cout << " * ";
            }
            cout << endl;
    }
}

main() {
     int baris,kolom;
    cout << ”Banyak baris : ”; cin >> baris;
    cout << ”Banyak kolom : ”; cin >> kolom;
    cetakBintang(baris,kolom);    // pemanggilan
}
```

- Illustrasi pemanggilan



PENGIRIMAN SECARA ACUAN (PASSING BY REFERENCE)

- Penulisan parameter di fungsi (parameter formal) dengan bentuk:

  tipe& parameter

- Yang dikirimkan ke fungsi adalah **alamat letak dari datanya sekaligus dengan nilai dari datanya** sehingga fungsi selain bisa membaca nilai dari yang memanggil juga bisa menulis ke yang memanggil (read/write)
- Fungsi yang menerima kiriman nilai ini akan menyimpannya di alamat yang sama dari nilai aslinya (membuat alias / sinonim)

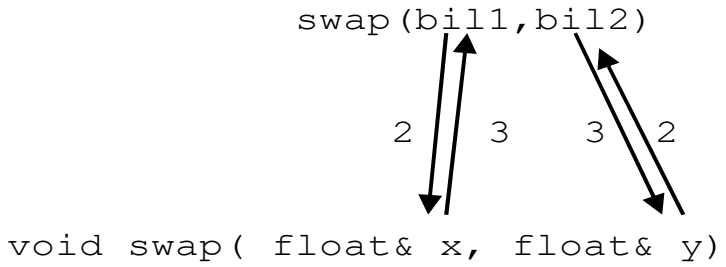
- Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil walaupun namanya sama ataupun berbeda
- Pengiriman nilai oleh pemanggil fungsi (Parameter aktual) hanya bisa berupa variable

Contoh : Fungsi untuk swap (tukar bilangan)

```
void swap( float& x, float& y){
        float temp = x;
        x = y;
        y = temp;
}

main() {
        float bil1 = 2, bil2 = 3;
        cout << bil1 << " , " << bil2 << endl;    // 2   3
        swap(bil1,bil2);                          // pemanggilan
        cout << "Hasil Swap / Tukar : ";
        cout << bil1 << " , " << bil2 << endl;    // 3   2
}
```

- Illustrasi



FUNGSI BOOLEAN

- Dalam beberapa situasi terkadang diperlukan suatu fungsi untuk mengevaluasi suatu kondisi apakah bernilai benar atau salah.
- Jika fungsi bernilai true (benar) bernilai 1 dan jika bernilai false (salah) bernilai 0.
- Bentuk umumnya :

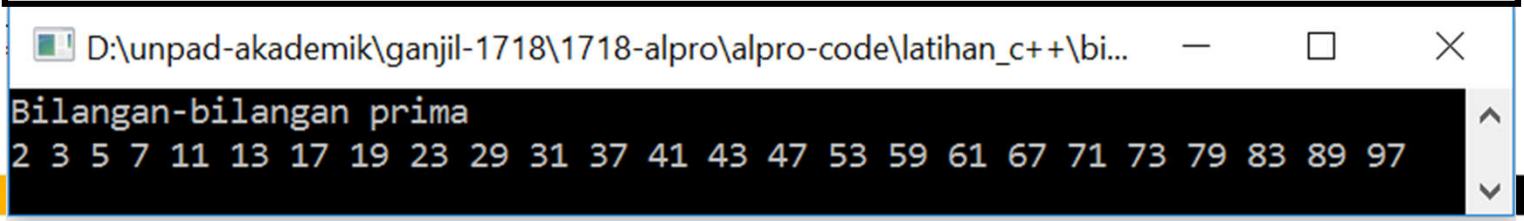
```
int  namaFungsiBoolean (parameter..) {
        . . .
        return ( 1 / 0);
}
```

Contoh : Fungsi untuk menentukan bilangan prima.

```
if (isPrima(bilangan))
    AKSI
```

```
#include <iostream.h>
#include <math.h>

int isPrima(int bil){
   float sqrtBil = sqrt(float(bil));
   if (bil < 2) return 0;                    // false
   if (bil == 2) return 1;                   // true
   if (bil % 2 == 0) return 0;               // false
   for (int gjl=3; gjl <= sqrtBil; gjl+=2) {
        if (bil % gjl == 0) return 0;        //  false
   }
   return 1;                                 // true
}
main() {
   cout << "Bilangan-bilangan prima" << endl;
   for (int n=1; n < 100; n++) {
        if (isPrima(n))
            cout << n << " " ;
   }
   cout << endl;
}
```



Studi Kasus : Mencari luas Persegi Panjang

```
void inputData(float& pjg, float& lbr);
float hitungLuas(float pjg,float lbr);
void cariLuas(float pjg,float lbr, float& hsl);
void cetakData(float pjg, float lbr, float hsl);
main() {
     float panjang,lebar,luas;
     inputData(panjang,lebar);
     luas=hitungLuas(panjang,lebar);            // panggil fungsi
     cetakData(panjang,lebar,luas);

     cariLuas(panjang,lebar,luas);              // panggil void
     cetakData(panjang,lebar,luas);
}
void inputData(float& pjg, float& lbr){
     cout<<"Input Panjang = "; cin>>pjg;
     cout<<"Input Lebar   = "; cin>>lbr;
}
float hitungLuas(float pjg,float lbr){                  // fungsi
     return (pjg*lbr);
}
void cariLuas(float pjg,float lbr, float& hsl){         // void
     hsl=pjg*lbr;
}
void cetakData(float pjg, float lbr, float hsl){
     cout<<"Panjang = " << pjg <<endl;
     cout<<"Lebar   = " << lbr <<endl;
     cout<<"Luas    = " << hsl <<endl;
}
```


ANY
QUESTIONS?

Sesi Berakhir
TERIMA KASIH