

Mata Kuliah	:	Dasar Pemrograman
Bobot Sks	:	2
Dosen Pengembang	:	Riad Sahara, S.Si, M.T Syahid Abdullah, S.Si, M.Kom
Tutor	:	Syahid Abdullah, S.Si, M.Kom
Capaian Pembelajaran Mata Kuliah	:	1. Mahasiswa memahami definisi fungsi 2. Mahasiswa memahami struktur fungsi 3. Mahasiswa memahami parameter fungsi
Kompetensi Akhir di Setiap Tahap (Sub- Cpmk)		1. Mahasiswa mampu memahami Konsep Fungsi
Minggu Perkuliahan Online Ke-		13

JUDUL TOPIK – Fungsi (Part 1)

Fungsi

Fungsi (Function) merupakan blok dari kode yang dirancang untuk melaksanakan tugas khusus. Kegunaan dari fungsi ini adalah untuk:

- Mengurangi pengulangan penulisan program yang berulang atau sama.
- Program menjadi lebih terstruktur, sehingga mudah dipahami dan dapat lebih dikembangkan.

Fungsi-fungsi yang sudah kita kenal sebelumnya adalah fungsi `main()`, yang bersifat mutlak, karena fungsi ini program akan dimulai, sebagai contoh yang lainnya fungsi `printf()`, `cout()` yang mempunyai tugas untuk menampilkan informasi atau data layar dan masih banyak lainnya.

Struktur Fungsi

Sebuah fungsi sederhana mempunyai bentuk penulisan sebagai berikut :

```
nama_fungsi(argumen)
{
... pernyataan / perintah;
... pernyataan / perintah;
... pernyataan / perintah;
}
```

Keterangan:

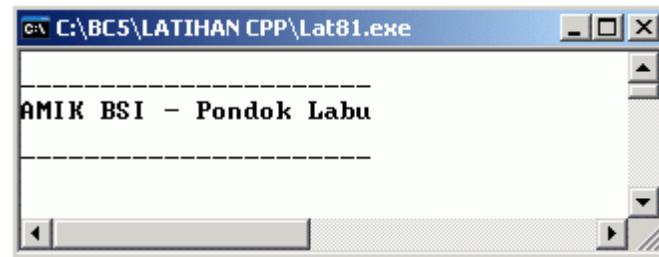
- Nama fungsi, boleh dituliskan secara bebas dengan ketentuan, tidak menggunakan spasi dan nama-nama fungsi yang mempunyai arti sendiri.
- Argumen, diletakan diantara tanda kurung “()” yang terletak dibelakang nama fungsi. Argumen boleh diisi dengan suatu data atau dibiarkan kosong.
- Pernyataan / perintah, diletakan diantara tanda kurung ‘{ }’.
- Pada pemanggilan sebuah fungsi, cukup dengan menuliskan nama fungsinya.

Contoh pembuatan fungsi sederhana dengan bahasa C++

```
/* pembuatan fungsi garis() */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
garis()
{
printf("\n-----\n");
}
/* program utama */
main()
{
clrscr();
garis(); //memanggil fungsi garis
cout<<"AMIK BSI - Pondok Labu"<<endl;;
garis(); //memanggil fungsi garis
getche();
}
```

}

Output yang akan dihasilkan, dari program contoh diatas adalah :



Prototipe Fungsi

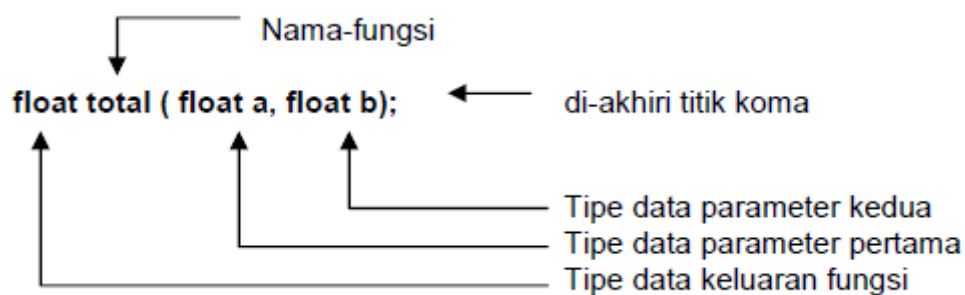
Prototipe fungsi digunakan untuk mendeklarasikan ke kompiler mengenai:

- Tipe data keluaran dari fungsi.
- Jumlah parameter yang digunakan
- Tipe data dari masing-masing parameter yang digunakan.

Keuntungan didalam pemakai prototipe yaitu :

- Kompiler akan melakukan konversi antara tipe parameter dalam definisi dan parameter fungsi.
- Jika jumlah parameter yang digunakan dalam definisi fungsi dan pada saat pemanggilan fungsi berbeda atau tidak sama, maka akan menunjukkan kesalahan,

Contoh prototipe fungsi :

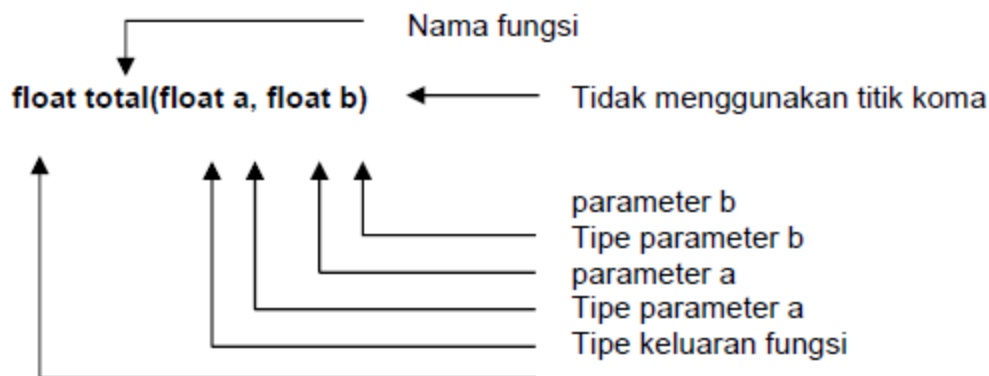


Bentuk definisi dalam penggunaan fungsi yang dideklarasikan dengan menggunakan prototipe, harus diubah. Sebagai contoh pada pendefinisian berikut :

```
float total(a, b)
```

```
float a, b;
```

Bentuk pendefinisian diatas harus diubah menjadi bentuk modern pendefinisian fungsi :

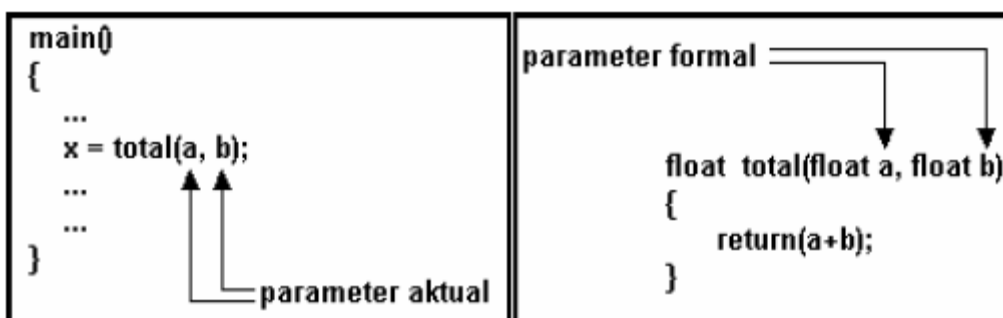


Parameter Fungsi

Terdapat dua macam parameter fungsi, yaitu :

- Parameter formal adalah variabel yang terdapat pada daftar parameter yang berada didalam definisi fungsi.
- Parameter Aktual adalah variabel yang digunakan pada pemanggilan suatu fungsi.

Bentuk penulisan Parameter Formal dan Parameter Aktual.



Ada dua cara untuk melewatkan parameter ke dalam fungsi, yaitu berupa :

1. Pemanggilan dengan nilai (Call by Value)

Pada pemanggilan dengan nilai yaitu nilai dari parameter aktual akan dimasukkan keparameter formal. Dengan cara ini nilai parameter aktual tidak bisa berubah, walaupun nilai dari parameter formal berubah. Berikut contoh pemanggilan dengan nilai dapat dilihat pada contoh berikut ;

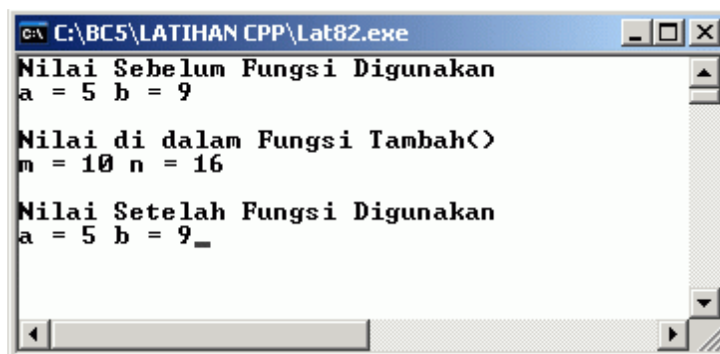
```
/* ----- */
/* Penggunaan Call By Value */
/* Program Tambah Nilai */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
tambah(int m, int n);
main()
{
    int a, b;
    a = 5;
    b = 9;
    clrscr();
    cout<<"Nilai Sebelum Fungsi Digunakan ";
    cout<<"\na = "<<a<<" b = "<<b;
    tambah(a,b);
    cout<<"\nNilai Setelah Fungsi Digunakan";
    cout<<"\na = "<<a<<" b = "<<b;
    getch();
}
tambah(int m, int n)
{
```

```

m+=5;
n+=7;
cout<<"\n\nNilai di dalam Fungsi Tambah()";
cout<<"\nm = "<<m<<" n = "<<n;
cout<<endl;
}

```

Output yang akan dihasilkan, dari program contoh diatas adalah :



```

C:\BC5\LATIHAN CPP\Lat82.exe
Nilai Sebelum Fungsi Digunakan
a = 5 b = 9
Nilai di dalam Fungsi Tambah()
m = 10 n = 16
Nilai Setelah Fungsi Digunakan
a = 5 b = 9

```

2. Pemanggilan dengan Referensi (Call by Reference)

Pemanggilan dengan reference merupakan pemanggilan alamat suatu variabel didalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel yang diluar dari fungsi dengan melaksanakan pengubahan nilai dari suatu variabel dilakukan didalam fungsi.

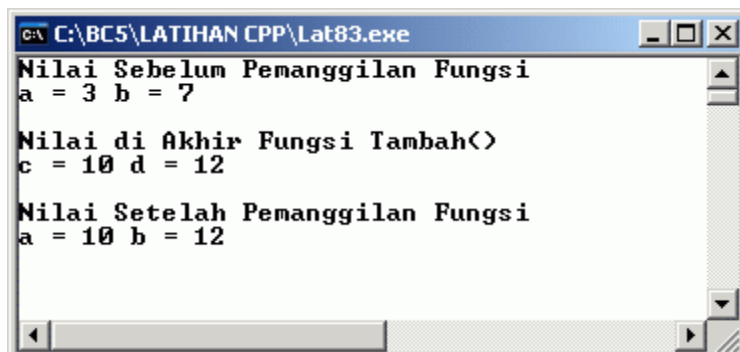
```

/* ----- */
/* Penggunaan Call By Reference */
/* Program Tambah Nilai */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
tambah(int *c, int *d);
main()
{

```

```
int a, b;
a = 4;
b = 6;
clrscr();
cout<<"Nilai Sebelum Pemanggilan Fungsi";
cout<<"\na = "<<a<<" b = "<<b;
tambah(&a,&b);
cout<<endl;
cout<<"\nNilai Setelah Pemanggilan Fungsi";
cout<<"\na = "<<a<<" b = "<<b;
getch();
}
tambah(int *c, int *d)
{
    *c+=7;
    *d+=5;
    cout<<endl;
    cout<<"\nNilai di Akhir Fungsi Tambah()";
    cout<<"\nc = "<<*c<<" d = "<<*d;
}
```

Output yang akan dihasilkan, dari program contoh diatas adalah :



```
C:\BC5\LATIHAN CPP\Lat83.exe
Nilai Sebelum Pemanggilan Fungsi
a = 3 b = 7
Nilai di Akhir Fungsi Tambah()
c = 10 d = 12
Nilai Setelah Pemanggilan Fungsi
a = 10 b = 12
```

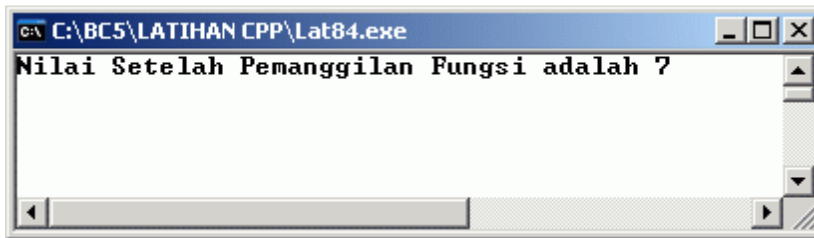
Pernyataan return().

Digunakan untuk mengirimkan nilai atau nilai dari suatu fungsi kepada fungsi yang lain yang memanggilnya. Pernyataan return() diikuti oleh argumen yang berupa nilai yang akan dikirimkan. Contoh pemakaian pernyataan return() dapat dilihat pada contoh berikut ;

```
/* ----- */
/* Penggunaan Fungsi return() */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
tambah(int *c); //prototype fungsi tambah
main()
{
    int a, b = 5;
    clrscr();
    a = tambah(&b);
    cout<<"Nilai Setelah Pemanggilan Fungsi adalah "<<a;
    getch();
}

tambah(int *c) //fungsi tambah
{
    return(*c+=2);
}
```

Output yang akan dihasilkan, dari program contoh-4 diatas adalah :



Pengiriman Data Ke Fungsi

1. Pengiriman Data Konstanta Ke Fungsi.

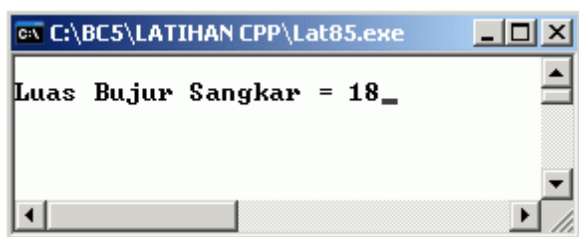
Mengirimkan suatu nilai data konstanta ke suatu fungsi yang lain dapat dilakukan dengan cara yang mudah, dapat dilihat dari program berikut :

```
/* ----- */
/* Pengiriman data Konstanta */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
luas(float sisi);
main()
{
    float luas_bs;
    clrscr();
    luas_bs = luas(4.25);
    cout<<"\nLuas Bujur Sangkar = "<<luas_bs;
    getch();
}
luas(float sisi)
{
    return(sisi*sisi);
}
```

Keterangan :

Dalam struktur program diatas dilihat bahwa, pernyataan `luas_bs=luas(4.25)`, akan dikirimkan nilai kepada fungsi `luas()`, untuk diolah lebih lanjut, yang nilai tersebut akan ditampung pada variabel `sisi`. Selanjutnya didalam fungsi `return` terjadi perkalian `sisi` dengan `sisi`, setelah itu hasil perkalian tersebut dikirim balik ke variabel `luas_bs` yang memanggil fungsi.

Output yang akan dihasilkan, dari program contoh-5 diatas adalah :



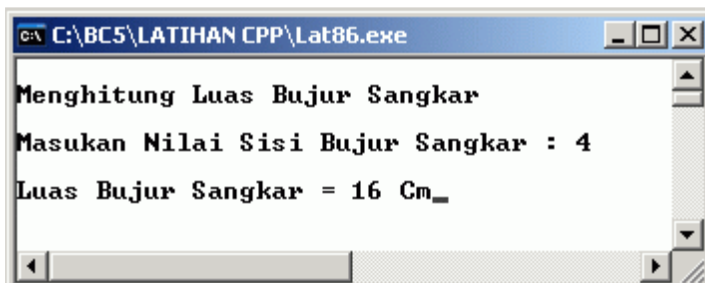
2. Pengiriman Data Variabel Ke Fungsi

Bentuk pengiriman data Variabel, sama seperti halnya pengiriman suatu nilai data konstanta ke suatu fungsi, hanya saja nilai yang dikirimkan tersebut senantiasa dapat berubah-ubah. Bentuk pengiriman tersebut dapat dilihat dari program berikut :

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
luas(float sisi);
main()
{
    float luas_bs, sisi_bs;
    clrscr();
    cout<<"\nMenghitung Luas Bujur Sangkar"<<endl;
    cout<<"\nMasukan Nilai Sisi Bujur Sangkar : ";
    cin>>sisi_bs;
    luas_bs = luas(sisi_bs);
```

```
cout<<"\nLuas Bujur Sangkar = "<<luas_bs<<" Cm";  
getch();  
}  
luas(float sisi)  
{  
    return(sisi*sisi);  
}
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Daftar Pustaka

Frieyadie, *Pemrograman C++ dengan Borland C++ 5.02 (Edisi Revisi)*. DIKTAT KULIAH PEMROGRAMAN KOMPUTER BINA SARANA INFORMATIKA, 2007.

Goodrich, Michael, Roberto Tamassia, and David Mount. *Data structures and algorithms in C++*. John Wiley & Sons, 2011.

Mehlhorn, Kurt, and Peter Sanders. *Algorithms and data structures: The basic toolbox*. Springer, 2010.