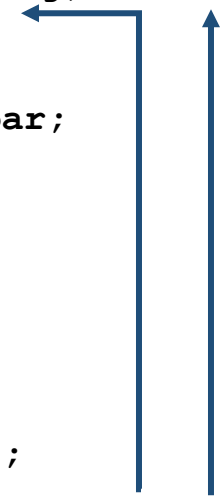
PENGIRIMAN PARAMETER

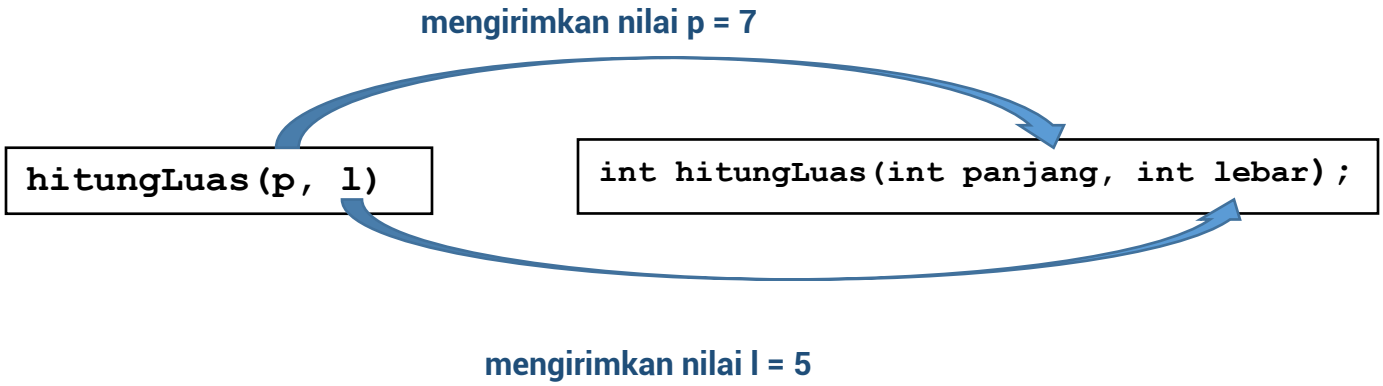
Sebuah fungsi maupun prosedur dapat dipanggil atau digunakan oleh fungsi atau bagian program yang lain. Komunikasi yang terjadi antara satu fungsi yang memanggil dengan fungsi yang dipanggil dilakukan dengan cara saling bertukar data atau informasi melalui **parameter**. Pada saat pemanggilan fungsi maupun prosedur terjadi pertukaran data, parameter aktual pada program pemanggil mengirimkan data pada parameter formal dalam fungsi.

Contoh:

```
#include<iostream>
usingname space std;

//pendefinisian fungsi
int hitungLuas (int panjang, int lebar){
     int luas;
     luas = panjang * lebar;
     return luas;
}
int main(){
     int luasPersegi;
     int p = 7; int l = 5;
     luasPersegi = hitungLuas (p, l);
}
```



Pada contoh program tersebut, pada saat fungsi hitungLuas dipanggil parameter aktual **mengirimkan nilai** dari variabel p = 7 dan l = 5, ke parameter formal panjang dan lebar pada fungsi.



Yang dikirimkan hanyalah **nilainya**, sehingga pengiriman parameter tersebut merupakan pengiriman parameter dengan nilai atau disebut dengan ***call by value***.

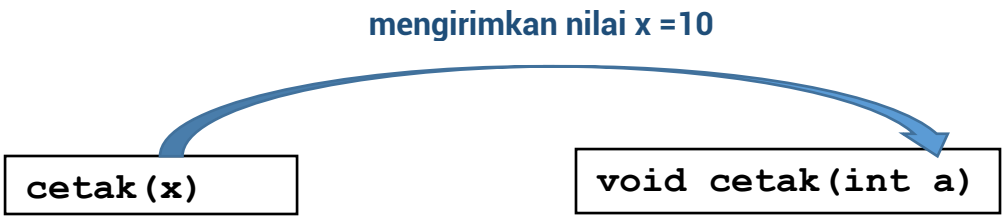
Pada sub program fungsi maupun prosedur terdapat dua buah cara untuk melewatkan parameter ke dalam sebuah fungsi maupun prosedur, yaitu **pemanggilan dengan nilai *(call by value)*** dan **pemanggilan dengan alamat *(call by reference)***.

CALL BY VALUE

Perhatikan contoh program berikut!

```
#include <iostream>

using namespace std;

void cetak (int a){

     a = 0;

     cout<<"Nilai pada prosedur cetak =
     "<< a <<endl;

}

int main(){

     int x = 10;

     cout<<"Nilai sebelum pemanggilan
     prosedur = "<< x <<endl;

     cetak(x); //mengirim nilai x ke
     prosedur

     cout<<"Nilai setelah pemanggilan
     prosedur = "<< x <<endl;

}
```

Pada program tersebut, didefinisikan sebuah prosedur bernama prosedur cetak(). Variabel a adalah parameter formal dalam prosedur, sedangkan variabel x yang bernilai 10 adalah parameter aktual. Variabel `x=10` merupakan nilai yang akan dikirimkan oleh program utama ke prosedur cetak pada saat pemanggilan.



Pada saat pemanggilan yang dikirim ke prosedur cetak **hanya nilai dari variabel** `x` (`10`). Setelah nilai `10` dikirim ke prosedur `cetak()`, disimpanlah nilai `10` ini ke variabel **`a`** pada prosedur. Maka, ketika program tersebut dieksekusi nilai keluaran program adalah:

```
Hasil eksekusi:
Nilai sebelum pemanggilan prosedur
x = 10

Nilai pada prosedur cetak()
a = 0

Nilai setelah pemanggilan prosedur
x = 10
```

Berdasarkan hasil keluaran program dapat diketahui bahwa nilai variabel x sebelum dan sesudah pemanggilan prosedur cetak **bernilai tetap**, walaupun variabel x yang dikirimkan ke prosedur dirubah nilainya dalam prosedur. Hal itu terjadi sebab nilai a pada prosedur **hanyalah salinan nilai** dari variabel x.

Jadi ketika isi dari variabel **`a`** berubah dalam prosedur tidak akan mempengaruhi isi variabel `x` pada program utama — begitu juga sebaliknya. Proses pengiriman parameter pada contoh tersebut merupakan proses pengiriman parameter dengan nilai ***(call by value).***

CALL BY VALUE

Berdasarkan contoh tersebut dapat disimpulkan bahwa pemanggilan dengan nilai *(call by value)* merupakan proses **penyalinan nilai** dari parameter aktual ke parameter formal, yang dikirimkan adalah nilainya bukan alamatnya. Fungsi atau prosedur yang menerima kiriman nilai akan menyimpannya di **alamat terpisah** dari nilai asli yang digunakan oleh program yang memanggil fungsi. Dengan cara ini, perubahan nilai di fungsi maupun prosedur (parameter formal) **tidak akan merubah nilai asli** (parameter aktual) di bagian program yang memanggilnya.

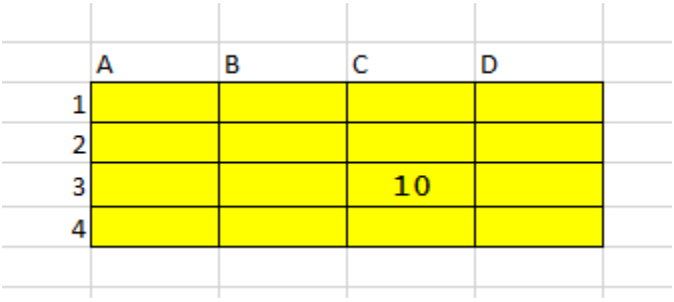
CALL BY REFERENCE

Pada pemanggilan parameter secara acuan ***(call by reference)***, pengirimanan data dilakukan dengan cara mengirimkan alamat dari suatu variabel ke dalam fungsi maupun prosedur yang dipanggil, jadi pada *call by refererence* yang dikirimkan adalah **alamat dari data bukan nilai datanya.**

Fungsi yang menerima kiriman alamat akan menggunakan **alamat yang sama** untuk mendapatkan nilai datanya. Perubahan nilai pada fungsi akan merubah nilai asli di bagian program yang memanggil fungsi. Sebagai contoh, perhatikan program berikut:

```
#include <iostream>
using namespace std;
void cetak(int *a){
     a = 0;
     cout<<"Nilai pada prosedur cetak = "<<
     a <<endl;
}
int main(){
     int x = 10;
     cout<<"Nilai sebelum pemanggilan
     prosedur = "<< x <<endl;
     cetak(&x); //mengirim alamat x ke
     prosedur
     cout<<"Nilai setelah pemanggilan
     prosedur = "<< x <<endl;
}
```

Pada saat pemanggilan penulisan parameter aktual dituliskan dengan opeartor "&" yang berarti menunjukan **alamat memori** dari variabel x. Sebagai ilustrasi, nilai di dalam memori – RAM – komputer digambarkan seperti berikut:

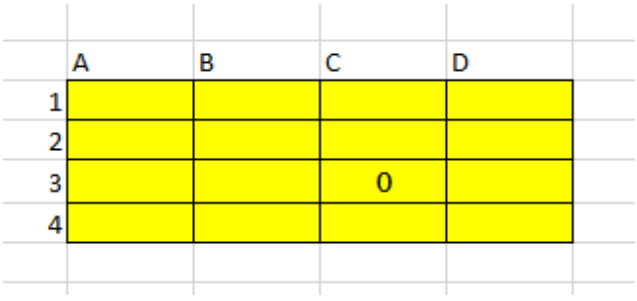


Gambar 1. **Nilai Variabel Awal x**

Variabel `x = "10"` (nilai 10 dimasukkan ke dalam variabel x). Variabel x ini memiliki alamat memori yaitu 3C, sehingga nilai 10 terletak pada alamat tersebut (lihat gambar). Ketika dilakukan pengiriman parameter ke dalam prosedur dengan *pass by reference,* yang dikirim adalah **alamat memori bukan nilai datanya**.

Sehingga parmaeter formal a pada prosedur **memiliki alamat memori yang sama** dengan parameter aktual x pada program utama. Terlihat pada pada parameter formal ditandai dengan tanda arsterik "*" yang menyatakan bahwa variabel tersebut adalah suatu variabel pointer (variabel yang berisi alamat dari variabel lain) .

Pada prosedur variabel a yang beralamat memori 3c dirubah nilainya menjadi 0, sehingga kini data pada alamat memori alamat 3c berubah nilainya. Oleh karena itu ketika program utama mencetak kembali nilai x yang beralamat sama, nilai data pada alamat tersebut telah berubah.



Ketika program tersebut dieksekusi maka nilai keluaran program adalah:

```
Hasil eksekusi:
Nilai sebelum pemanggilan prosedur
x = 10

Nilai pada prosedur cetak()
a = 0

Nilai setelah pemanggilan prosedur
x = 0
```

Nilai parameter aktual setelah prosedur cetak dipanggil berubah menjadi 0. Hal itu terjadi, karena alamat memori dari parameter aktual yang dikirimkan ke dalam prosedur. Sehingga ketika nilai dari parameter formal dalam prosedur berubah maka nilai parameter aktual pada program pemanggil berubah.

Apa Kegunaan *call by reference*?

Dari contoh program tersebut, variabel `x` dan `a` memiliki alamat memori yang sama sehingga jika salah satu dari variabel tersebut dirubah nilainya maka variabel lainnya yang memiliki alamat memori yang sama akan ikut berubah.

Jadi dari dalam badan prosedur atau fungsi kita dapat mengubah variabel yang berada di bagian program yang lain — asalkan menggunakan ***pass by reference.***