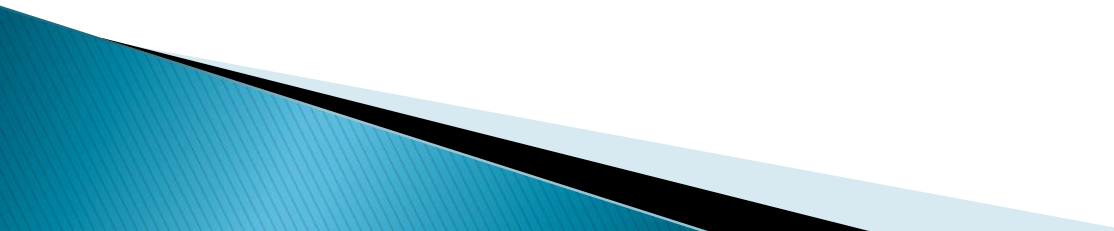
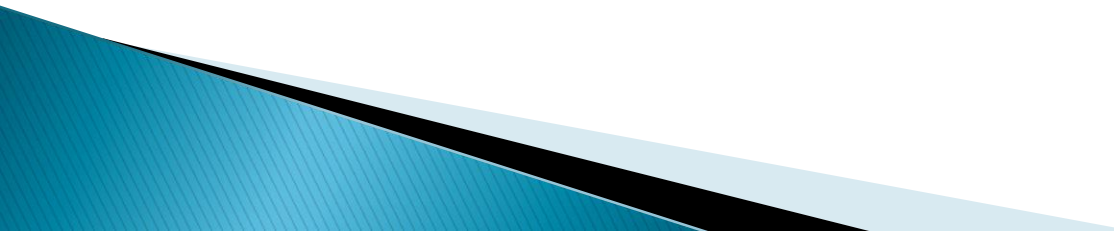


# Procedure , Function dan Parameter

# Pengantar Procedure dan Function

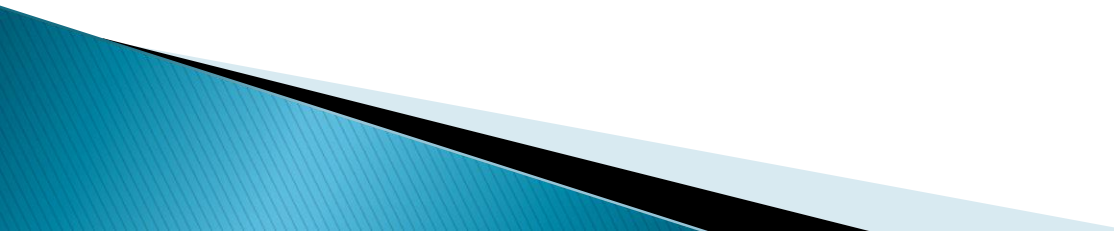
- ▶ Kode program panjang untuk program yang kompleks
  - ▶ Solusi → memecah program tersebut menjadi modul-modul menjadi lebih singkat.
  - ▶ Untuk memecah program yang kompleks, kita membutuhkan procedure dan function.
- 

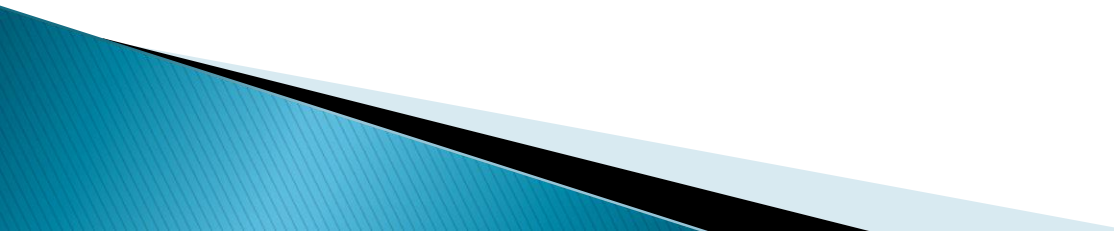
# Tujuan dari Procedure dan Function

- ▶ Merupakan penerapan konsep modular, yaitu memecah program menjadi modul-modul atau beberapa subprogram yang lebih sederhana.
  - ▶ Untuk instruksi yang sering dilakukan berulang-ulang, cukup dituliskan sekali saja dalam prosedur dan dapat dipanggil atau dipergunakan sewaktu-waktu bila diperlukan.
- 

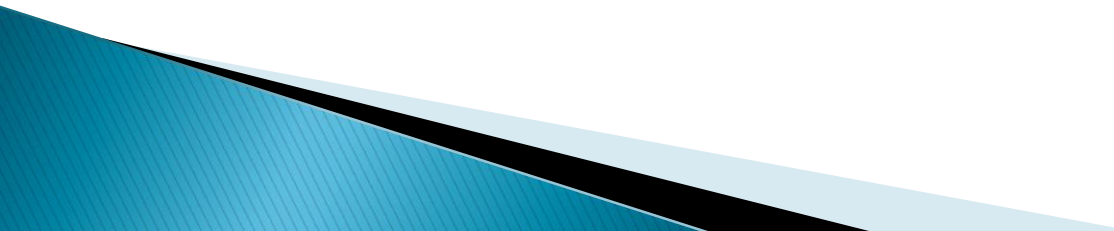
# Modular

Pemrograman Modular adalah suatu teknik pemrograman di mana program yang biasanya cukup besar dibagi-bagi menjadi beberapa bagian program yang lebih kecil sehingga akan mudah dipahami dan dapat digunakan kembali, baik untuk program itu sendiri maupun program lain yang memiliki proses yang sama.

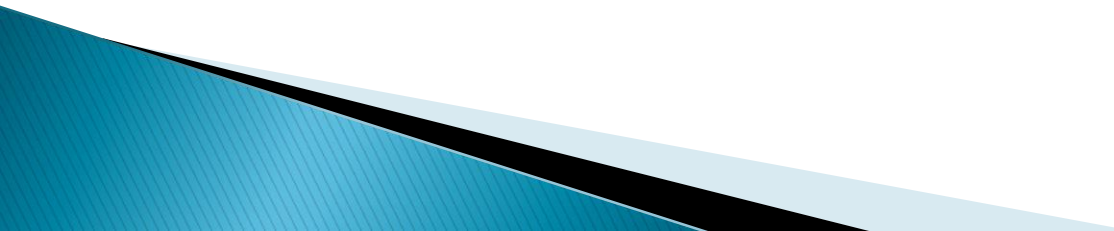


- ▶ Modul pada bahasa C++ dikenal dengan nama fungsi (function)
  - ▶ Bahasa C terdiri dari fungsi-fungsi, baik yang langsung dideklarasikan dalam program ataupun dipisah di dalam header file.
  - ▶ Fungsi yang selalu ada pada program C++ adalah fungsi **main**
- 

# Kelebihan modular :

- ▶ Program lebih pendek
  - ▶ Mudah dibaca dan dimengerti
  - ▶ Mudah didokumentasi
  - ▶ Mengurangi kesalahan dan mudah mencari kesalahan
  - ▶ Kesalahan yang terjadi bersifat “lokal”
  - ▶ Untuk aktifitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama secara berulang kali.
  - ▶ Kemudahan dalam menulis dan menemukan kesalahan (*debug*) program.
- 

# Teknik pemrograman modular :

- ▶ Program dipecah menjadi beberapa **subprogram** yang lebih kecil.
  - ▶ Subprogram (**modul**, ***routine***) kadang independen dari program utama sehingga dapat dirancang tanpa mempertimbangkan konteks tempat ia digunakan, bahkan dapat dirancang orang lain.
- 

# PROCEDURE

Pada dasarnya bahasa C++ tidak mengenal istilah prosedur, C++ hanya mengenal fungsi, prosedur dalam C++ dianggap sebagai fungsi yang tidak mengembalikan nilai, sehingga dalam pendeklarasian prosedur cukup dituliskan dengan kata kunci void dan diikuti dengan nama prosedur.

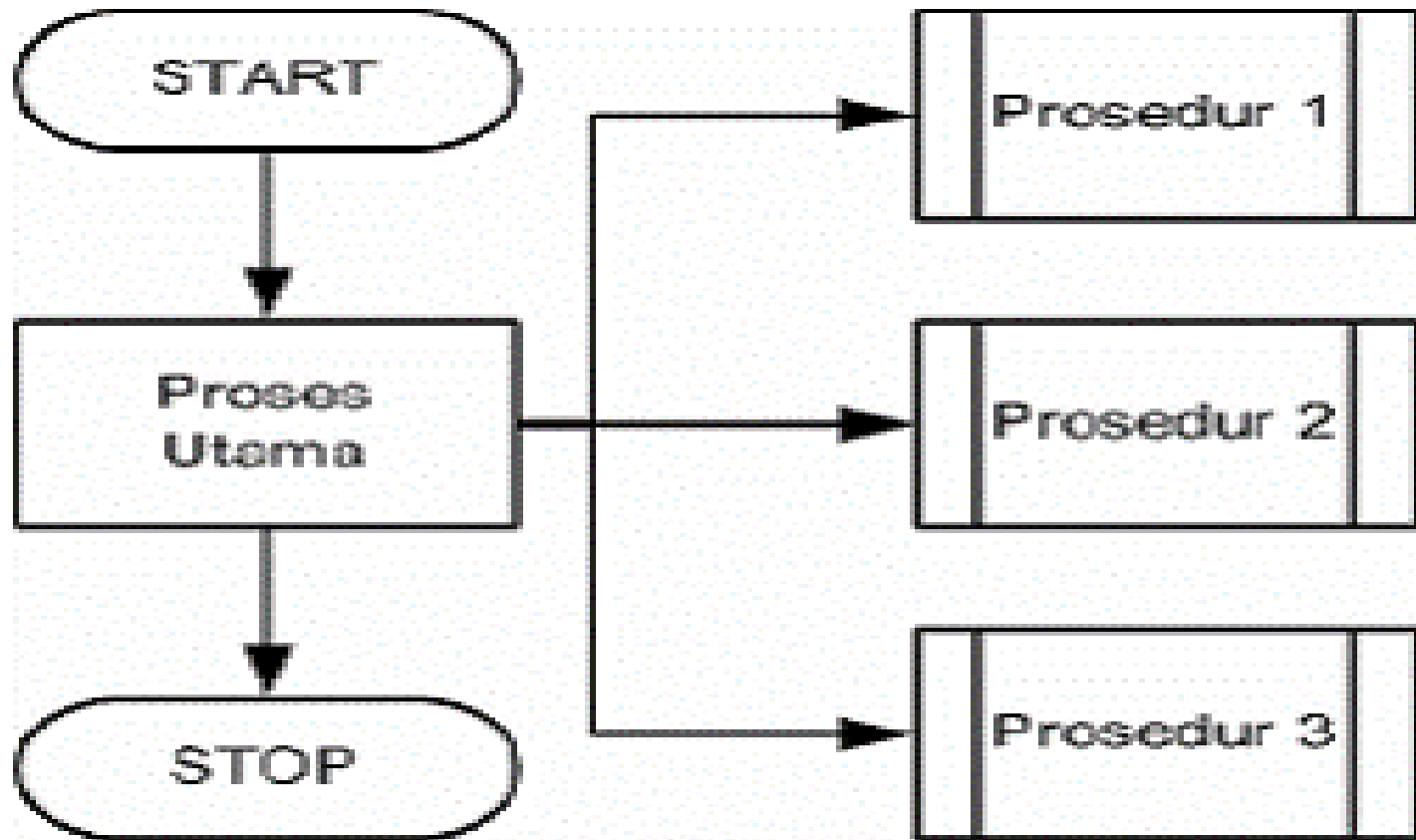


# Procedure dalam Pascal

- ▶ Merupakan sub program yang tidak mengembalikan nilai secara langsung.
- ▶ Prosedur memiliki struktur yang sama dengan struktur program, yaitu terdiri dari nama prosedur, deklarasi-deklarasi dan bagian deskripsi (*statement* atau aksi di dalam prosedur).
- ▶ Semua deklarasi di dalam prosedur bersifat lokal sehingga hanya bisa digunakan oleh prosedur itu saja, sedangkan deklarasi di dalam program utama bersifat global sehingga dapat dikenali di seluruh bagian program.

## Perbedaan Program Utama dan Procedure

- ▶ **PROCEDURE** mempunyai judul\_prosedur sedangkan Program utama tidak.
- ▶ Untuk tanda berakhirnya Procedure diakhiri dengan tanda End; sedangkan Program Utama End.



**Gambar 6.6. Skema penggunaan prosedur.**

Gambar 6.6. menunjukkan ada proses utama yang terjadi, dan ada prosedur yang sebenarnya merupakan bagian dari proses utama ini. Ketika proses utama membutuhkan suatu tugas tertentu, maka proses utama akan memanggil prosedur tertentu menyelesaikan tugas tersebut.

# Contoh

- ▶ Buatlah prosedur untuk menghitung luas segitiga dan luas persegi panjang

## **Prosedur Luas Segitiga:**

Input : –

Output : luas

Proses :

1. menghitung luas =  $0.5 \times (\text{alas} \times \text{tinggi})$
2. Mencetak luas

## **Prosedur Persegi Panjang:**

Input : –

Output : lua\_prsg

Proses :

1. menghitung luas =  $\text{panjang} \times \text{lebar}$
2. Mencetak luas\_prsg

## **Program Utama :**

Input : alas,tinggi

Output : –

Proses :

1. Membaca nilai alas yang diinput user
2. Membaca nilai tinggi yang diinput user
3. Pemanggilan procedure Luas Segitiga dan Procedure Persegi Panjang

# Pseudocode Procedure

## PROCEDURE Luas\_Segitiga:

{menghitung luas segitiga,  $luas = (alas \times tinggi) / 2$ }

### DEKLARASI :

luas : float

### DESKRIPSI

luas =  $0.5 * (alas * tinggi)$

write (luas)

## PROCEDURE Persegi\_pj:

{menghitung luas persegi\_pjg,  $luas = panjang * lebar$ }

### DEKLARASI :

luas\_prsg : float

### DESKRIPSI

luas\_prsg = panjang \* lebar

write (luas\_prsg)

## Program\_Utama:

{menghitung luas segitiga, luas persegi\_pjg dengan procedure

### DEKLARASI :

alas, tinggi, panjang, lebar : int

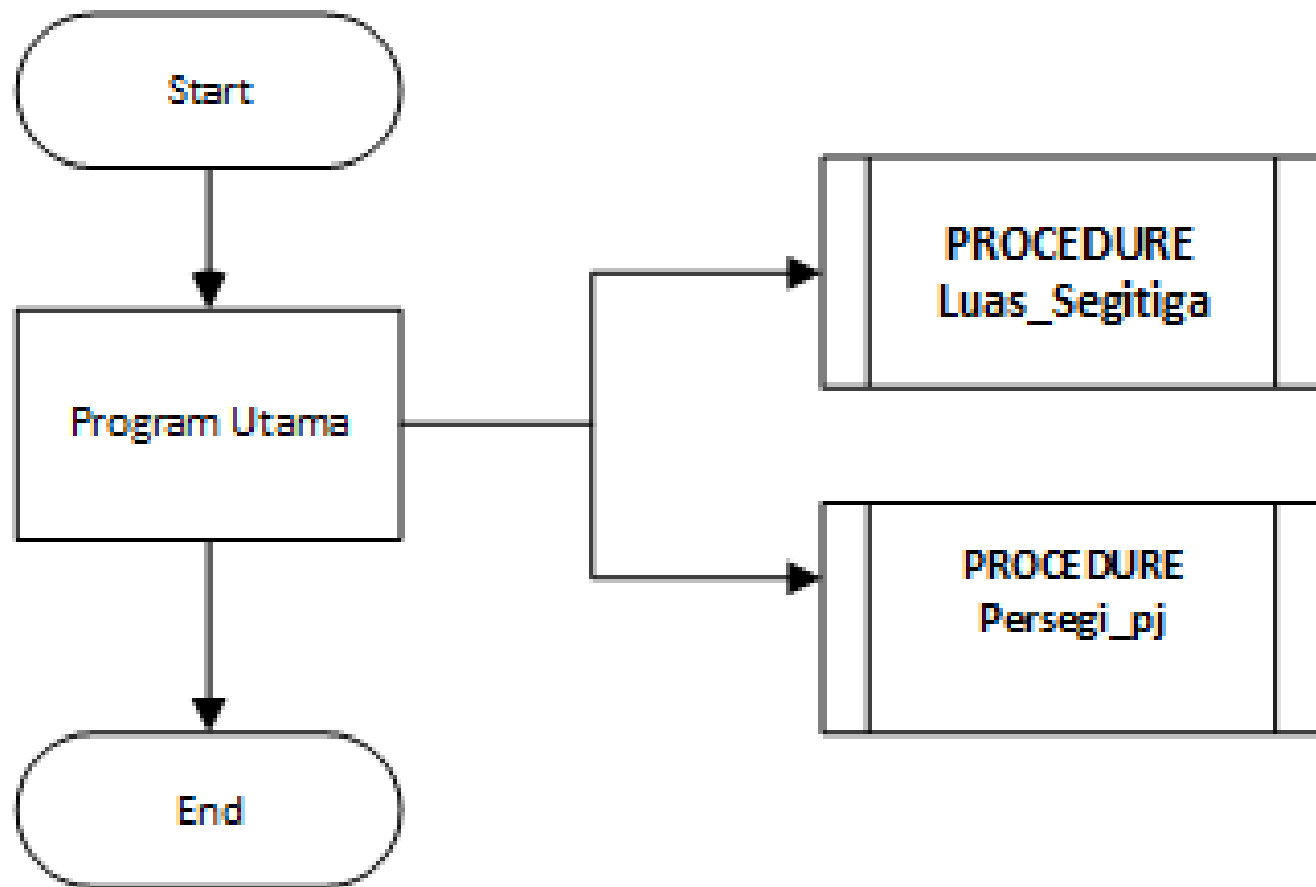
procedure Luas\_Segitiga, persegi\_pj

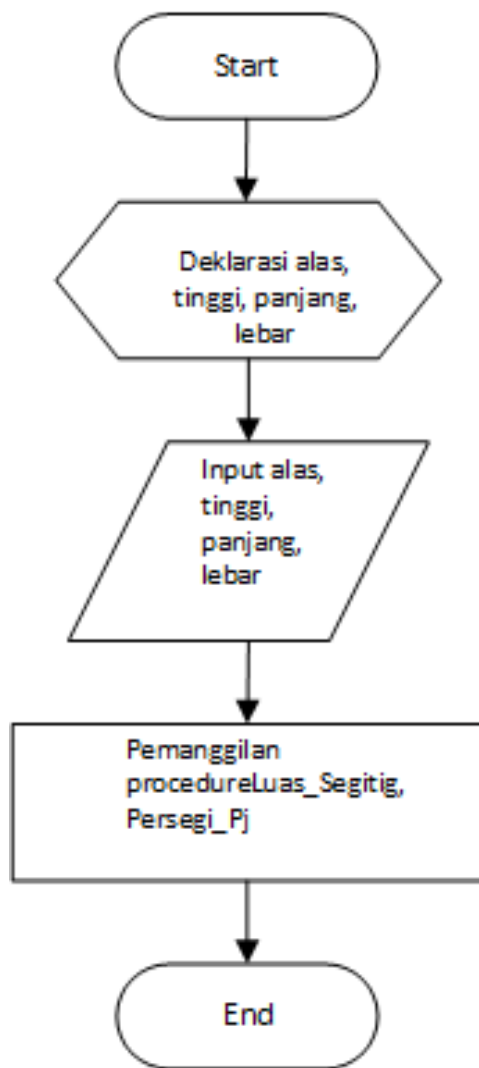
### DESKRIPSI

input = alas, tinggi, panjang, lebar

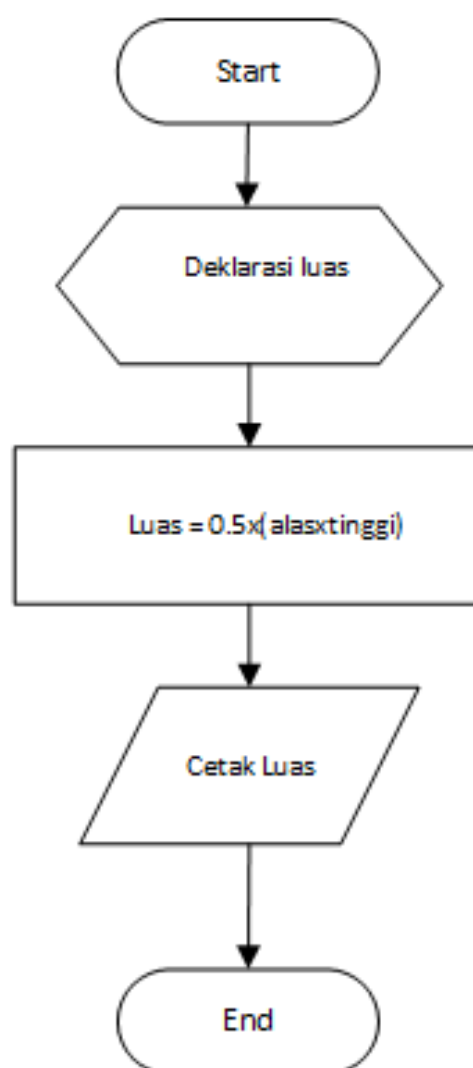
Luas\_Segitiga

Persegi\_Pj

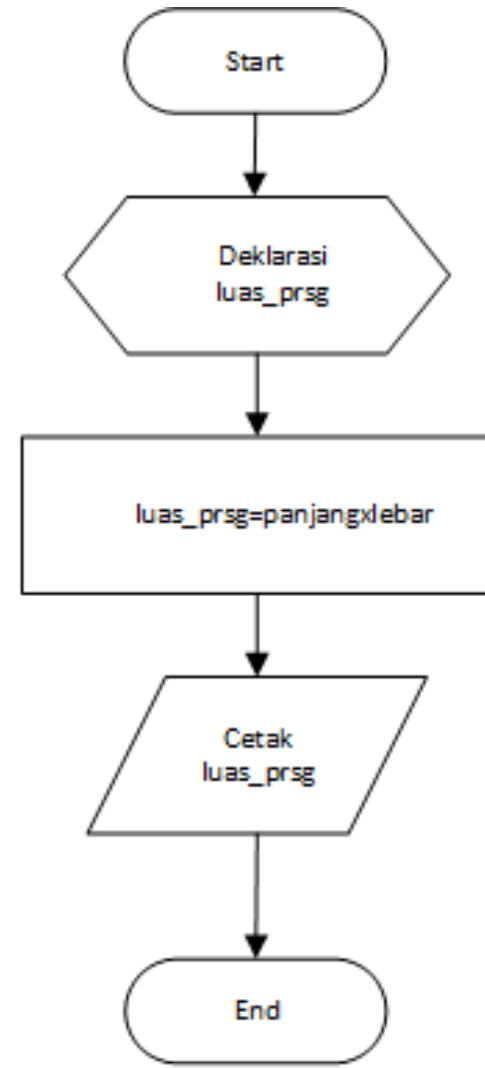




Program Utama



PROCEDURE Luas\_Segitiga



PROCEDURE Persegi\_pj:

```

#include <iostream>
#include <stdio.h>
#include <conio.h>

int al,tg,pj,lb;
void segitiga()
{
    float luas=0.5*(al*tg);
    cout<<"luas segitiga = "<<luas<<endl;
}
void persegi()
{
    float luas_prs=pj*lb;
    cout<<"luas persegi = "<<luas_prs<<endl;
}

int main()
{

    cout<<"alas = ";cin>>al;
    cout<<"tg = ";cin>>tg;
    cout<<"pj = ";cin>>pj;
    cout<<"lb = ";cin>>lb;
    segitiga();
    persegi();
    getch();
    return 0;
}

```

D:\BAHAN NGAJAR KULIAH

```

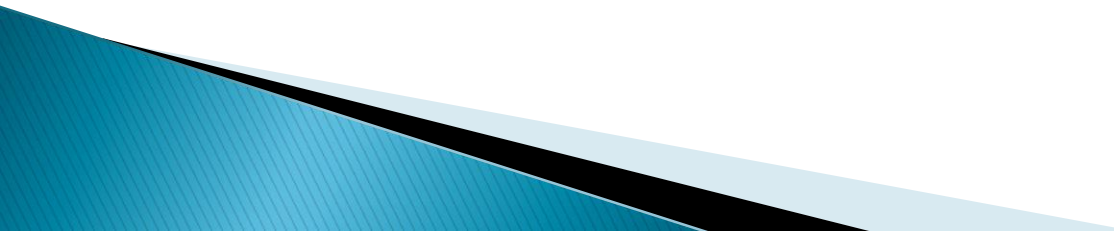
alas = 7
tg = 3
pj = 6
lb = 4
luas segitiga = 10.5
luas persegi = 24

```



# Contoh Program Prosedur

```
▶ #include<iostream>
▶ #include<conio>
▶ int addition (int a, int b)
▶ {
▶     int r;
▶     r=a+b;
▶     return (r);
▶ }
▶ void main ()
▶ {
▶     int z;
▶     z = addition (5,3);
▶     cout << "The result is " << z;
▶     getch();
▶ }
```



## Try This

```
#include <conio.h>
#include <iostream>
using namespace std;
int a=2; int b=3;
void penjumlahan()
{ int hasil=a+b;
  cout<<hasil;
}
int main()
{
  penjumlahan();
  getch();
}
```


# Jenis fungsi di C++ :

- ▶ Fungsi yang tidak mengembalikan nilai (void)
- ▶ Fungsi yang mengembalikan nilai (nonvoid)

# Fungsi void

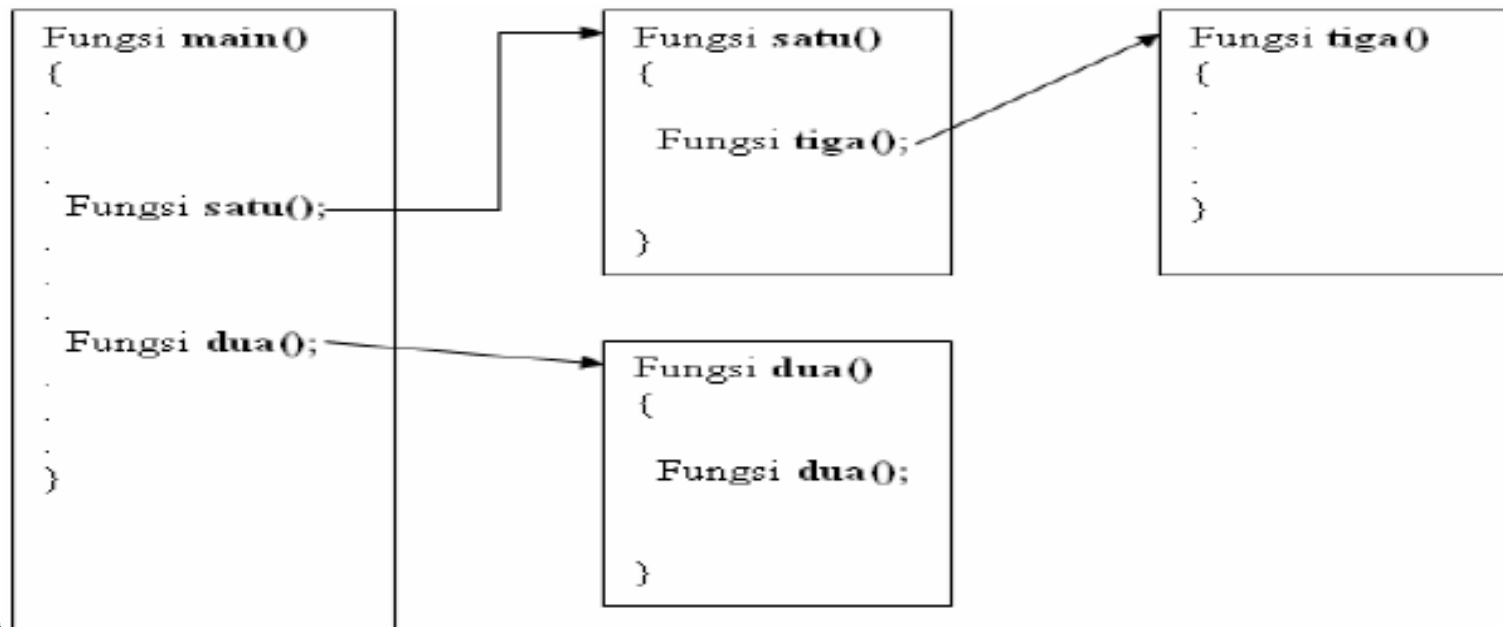
- ▶ Fungsi yang void sering disebut juga prosedur
- ▶ Disebuta void karena fungsi tersebut tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut.
- ▶ Ciri:
  - tidak adanya keyword return.
  - tidak adanya tipe data di dalam deklarasi fungsi.
  - menggunakan keyword void.
- ▶ Tidak dapat langsung ditampilkan hasilnya
- ▶ Tidak memiliki nilai kembalian fungsi
- ▶ Keyword void juga digunakan jika suatu function tidak mengandung suatu parameter apapun.

# Fungsi non void

- ▶ Fungsi non-void disebut juga function
  - ▶ Disebut non-void karena mengembalikan nilai kembalian yang berasal dari keluaran hasil proses function tersebut
  - ▶ Ciri:
    - ada keyword return
    - ada tipe data yang mengawali deklarasi fungsi
    - tidak ada keyword void
  - ▶ Memiliki nilai kembalian
  - ▶ Dapat dianalogikan sebagai suatu variabel yang memiliki tipe data tertentu sehingga dapat langsung ditampilkan hasilnya
- 

# Pemanggilan fungsi

- ▶ Pada dasarnya fungsi dapat memanggil fungsi lain, bahkan fungsi dapat memanggil dirinya sendiri (rekursif)



# Contoh :

- ▶ Void : 

```
void tampilkan_jml (int a, int b)
{ int jml;
  jml = a + b;
  cout<<jml;}
```
- ▶ Non-void : 

```
int jumlah (int a, int b)
{ int jml;
  jml = a + b;
  return jml; }
```

# Contoh fungsi void :

```
#include <iostream.h>
#include <conio.h>
void luas(int &ls, int p, int l)
{ ls = p*l; }
main() {
    int pj, lb, hsl;
    cout<<"Panjang = ";cin>>pj;
    cout<<"Lebar    = ";cin>>lb;
    luas(hsl,pj,lb);
    cout<<"\nLuasnya = "<<hsl;
    getch();
}
```



# Contoh fungsi non void :

```
#include <iostream.h>
#include <conio.h>
int luas(int p, int l)
{return (p*l); }
main() {
    int pj, lb;
    cout<<"Panjang = ";cin>>pj;
    cout<<"Lebar    = ";cin>>lb;
    cout<<"\nLuasnya =
"<<luas(pj, lb);
    getch();
}
```

# Parameter

- ▶ Suatu variable yang berfungsi sebagai penampung nilai pada procedure atau function, yang diberikan oleh pemanggil procedure atau function.

Contoh:

Procedur

nama\_pro(parameter,parameter:tipe\_data);

Procedur hitung(angka1,angka2:integer);

Procedur bagi(angka1:integer; angka2:real);

■ ■ ■

- ▶ Parameter yang dikirimkan dari modul utama ke modul prosedur disebut dengan parameter nyata (actual parameter) dan parameter yang ada dan dituliskan pada judul prosedur disebut dengan parameter formal (formal parameter).
- ▶ Proses pengiriman data lewat parameter nyata ke parameter formal disebut dengan parameter passing.
- ▶ Parameter nyata dan parameter formal harus dengan tipe yang sama.

# Parameter dalam Procedure

- ▶ Parameter formal

Variabel yang ada pada daftar parameter dalam definisi fungsi. Pada fungsi jumlah () program 7-6 misalnya, x dan y dinamakan sebagai parameter formal.

```
float jumlah (float x,float y)
{
    return (x+y);
}
```

## ▶ Parameter Aktual

Parameter yang dapat berupa variabel atau konstanta maupun ungkapan yang dipakai dalam pemanggilan fungsi

```
main()
{
    ...
    x= jumlah (a,b);
    ...
}
```

# Contoh Program Parameter formal dan Aktual

```
#include <stdio.h>
void tukar (int x, y);
main()
{
    int a,b;
    a=99;
    b=11;
    printf("nilai sebelum pemanggilan fungsi :\n");
    printf("nilai a=%d nilai b=%d\n\n",a,b);
    tukar (a,b);
    printf("nilai sesudah pemnggilan fungsi :\n");
    printf("nilai a=%d nilai b =%d\n\n",a,b);
}
```

```
void tukar (int px,py)
int z;
z=px;
px=py;
py=z;
printf("nilai diakhiri fungsi :\n");
printf("nilai px=%d nilai py=%d\n\n",px,py);
}
```

# Variabel Lokal dan Variabel Global

## ► Variabel Lokal

- Dideklarasikan dalam subprogram
- Hanya dikenali secara lokal dalam sebuah subprogram
- tidak dapat dipanggil, diakses dan diubah oleh subprogram yang lain, bahkan oleh program utama

```
Void gaji (void)
{
Int x  → variabel lokal
.....
.....
}
```

## ► Variabel Global

- Dideklarasikan dalam program utama
- dapat dipanggil, diakses dan diubah oleh subprogram apapun yang terdapat dalam program tersebut

```
#include <stdio.h>
int i = 255;   → variabel global
Void tambah (void)
{
int x
.....
.....
}
```



# Tipe pengiriman parameter

- ▶ **Pass by Value**
- ▶ **Pass by Reference**

## Pass By Value

- ▶ Parameter yang dikirimkan berupa nilai (value)nya saja. Jadi apabila terjadi pengubahan nilai pada prosedur ataupun function tidak akan mempengaruhi nilai pada variabel yang dipassingkan, atau yang dikirimkan.

# Try This

```
▶ #include <iostream>
▶ using namespace std; //membuat fungsi Tukar

▶ void Tukar(int a, int b){
▶     int c;
▶     c=a;
▶     a=b;
▶     b=c;
▶     cout<<"nnKeadaan di dalam fungsi"<<endl;
▶     cout<<"Angka Pertama = "<<a<<" Angka Kedua = "<<b<<endl;
▶ }
▶ //fungsi utama
▶ main(){
▶     int x,y;
▶     cout<<"Masukkan angka pertama : "; cin>>x;
▶     cout<<"Masukkan angka kedua : "; cin>>y;
▶     cout<<"nKeadaan Awal"; cout<<"nAngka Pertama = "<<x<<" Angka kedua = "<<y;
▶     Tukar(x,y);
▶     cout<<"nkeadaan Akhir"<<endl; cout<<"Angka Pertama = "<<x<<" Angka kedua = "<<y;
▶ }
```

# Pass By Value

```
fungsi.cpp
1  #include <iostream>
2  using namespace std; //membuat fungsi Tukar void Tukar(int a, int b){
3  Tukar(int a, int b){
4      int c;
5      c=a;
6      a=b;
7      b=c;
8
9      cout<<"\n\nKeadaan di dalam fungsi"<<endl;
10     cout<<"Angka Pertama = "<<a<<"Angka Kedua = "<<b<<endl;
11 }
12 //fungsi utama
13 main(){
14     int x,y;
15     cout<<"Masukkan angka pertama : "; cin>>x;
16     cout<<"Masukkan angka kedua : "; cin>>y;
17     cout<<"\nKeadaan Awal"; cout<<"\nAngka Pertama = "<<x<<" Angka kedua = "<<y; Tukar(x,y);
18     cout<<"\nkeadaan Akhir"<<endl; cout<<"Angka Pertama = "<<x<<"Angka kedua = "<<y;
```

D:\struct\fungsi.exe

```
Masukkan angka pertama : 40
Masukkan angka kedua : 23

Keadaan Awal
Angka Pertama = 40 Angka kedua = 23

Keadaan di dalam fungsi
Angka Pertama = 23Angka Kedua = 40

keadaan Akhir
Angka Pertama = 40Angka kedua = 23
-----
Process exited after 5.296 seconds with return value 0
Press any key to continue . . .
```

# Pass By Reference

- ▶ Parameter yang dikirimkan berupa acuan. Jadi apabila terjadi pengubahan nilai pada prosedur ataupun function akan mempengaruhi nilai pada variabel yang dipassingkan, atau yang dikirimkan. Kata kuncinya pemberian tambahan kata var pada pendeklarasian parameter.

## Try This

- ▶ `#include <iostream>`
- ▶ `using namespace std;`
- ▶ `//membuat fungsi Pangkat2`
- ▶ `void Pangkat2(int& X){`
- ▶ `X = X*X;`
- ▶ `cout<<"Nilai didalam Fungsi : "<<X<<endl;`
- ▶ `}`
- ▶ `//Fungsi utama`
- ▶ `main(){`
- ▶ `int bilangan;`
- ▶ `cout<<"Masukan Sebuah bilanganbulat : ";cin>>bilangan;`
- ▶ `cout<<endl;`
- ▶ `cout<<"Nilai awal : "<<bilangan<<endl;`
- ▶ `Pangkat2(bilangan);`
- ▶ `cout<<"Nilai akhir : "<<bilangan<<endl;`
- ▶ `}`

# Try This

fungsi.cpp

```
1  #include <iostream>
2  using namespace std;
3  //membuat fungsi Pangkat2
4  void Pangkat2(int& X){
5      X = X*X;
6      cout<<"Nilai didalam Fungsi : "<<X<<endl;
7  }
8  //Fungsi utama
9  main(){
10     int bilangan;
11     cout<<"Masukan Sebuah bilanganbulat : ";cin>>bilangan;
12     cout<<endl;
13     cout<<"Nilai awal  : "<<bilangan<<endl;
14     Pangkat2(bilangan);
15     cout<<"Nilai akhir : "<<bilangan<<endl;
16 }
```

D:\struct\fungsi.exe

Nilai didalam Fungsi : 400  
Nilai akhir : 400

-----  
Process exited after 10.58 seconds with return value 0  
Press any key to continue . . .

**Terima Kasih**