Pertemuan 1 : algoritma flowchart dan pseudocode

Pertemuan 2: pengenalan c++

Pertemuan 3: percabangan

Pertemuan 4: perulangan/looping

Pertemuan 5: function,struct,class

Pertemuan 6: inherit,polymorfisme(construktor,destructor,overload,override)

Definisi dan contoh yang mendukung

**PERTEMUAN 1 : ALGORITMA FLOWCHART DAN PSEUDOCODE**

**>>>**

**Algoritma :** urutan langkah – langkah logis penyelesaian masalah yang disusun secara sistematis.

1. Urutan logis berarti urutan langkah-langkah harus benar, dan langkah-langkah itu sendiri juga harus benar
2. Setiap orang punya pola pikir yang berbeda, karena itu akan menghasilkan algoritma yang berbeda
3. Algoritma dikatakan benar apabila dapat memecahkan masalah
4. Perlu dipertimbangkan masalah efisiensi dalam penyusunan algoritma

Contoh :

Mencetak semua bilangan ganjil dari 1-100

Tahap1: mulai

Tahap2: [inisialisasi]

Tahap3: ulangi Tahap3-Tahap5,sampai i=100

Tahap4: if(i mod 2==1)then,cetak nilai i

Tahap5: i++

Tahap6: selesai

**>>>**

**Pseudocode :** Merupakan kode yang menyerupai kode program yang sesungguhnya.

1. Tidak ada ukuran standart penulisan pseudocode.
2. Ditunjukan untuk dibaca manusia bukan dibaca mesin

Contoh :

Mencetak semua bilangan ganjil dari 1-100

Input :

-

Output :

i = non negatif integer

i = 1

while i<=100

if i mod 2 == 1 then

write (i)

endif

i++

endwhile

**>>>**

**Flowchart :** Merupakan gambaran dalam bentuk diagram alir dari algoritma-algoritma dalam suatu program yang menyatakan arah alur program tersebut.

1. Disajikan dalam bentuk gambar dan grafik
2. Dapat membantu dalam memahami alur program

Simbol simbol :

|  |  |
| --- | --- |
|  | Terminator  (mulai / berhenti) |
|  | process |
|  | Input / Output data |
|  | Descision  Percabangan |
|  | Document |
|  | Preparation  Proses inisialisasi |
|  | Manual input |

Contoh :

Mencetak semua bilangan ganjil dari 1-100

mulai

benar

benar

salah

salah

selesai

i + +

Output

Nilai i

i mod 2==1

i <=100

Inisialisasi

i =1

**PERTEMUAN 2 : PENGENALAN C++**

**>>>**

Bahasa C++ diciptakan oleh Bjarne Stroustrup di AT&T BellLaboratories awal tahun 1980-an.

tersusun dari 4 (empat) bagian utama, yaitu :

1. Bagian komentar yang ditandai dengan symbol // dan pasangan /\* … \*/

2. Bagian pengarah compiler yang ditandai dengan symbol # ( mis : #include )

3. Bagian deklarasi

Main ( ) merupakan contoh fungsi, sedangkan pesan adalah contoh data. Baik data maupun fungsi harus dideklarasikan. Data perlu dideklarasikan agar compiler tahu berapa byte memori yang harus disediakan untuk data yang bersangkutan, sedangkan fungsi perlu dideklarasikan agar compiler dapat memeriksa ketepatan pemanggilan fungsi yang bersangkutan. Deklarasi fungsi sering disebut pula prototype fungsi

4. Bagian definisi

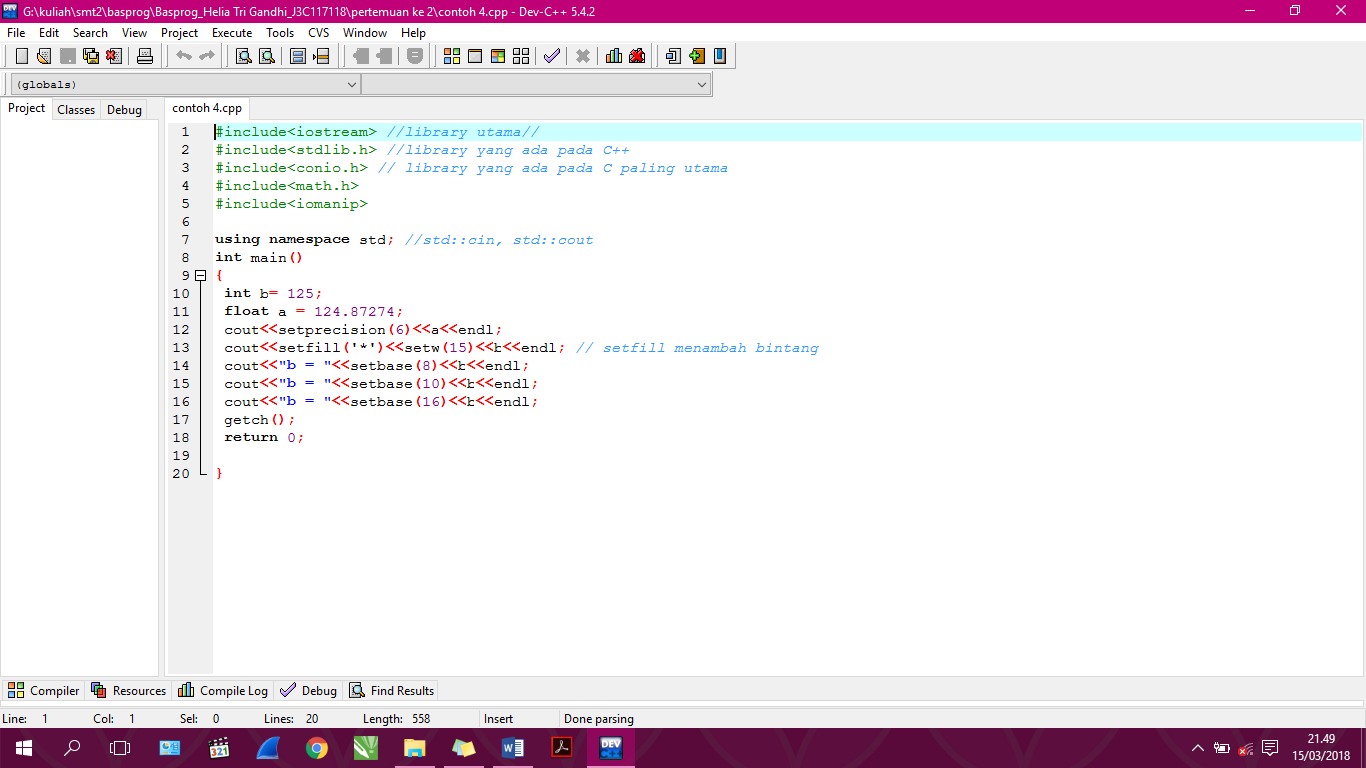
Input dan output

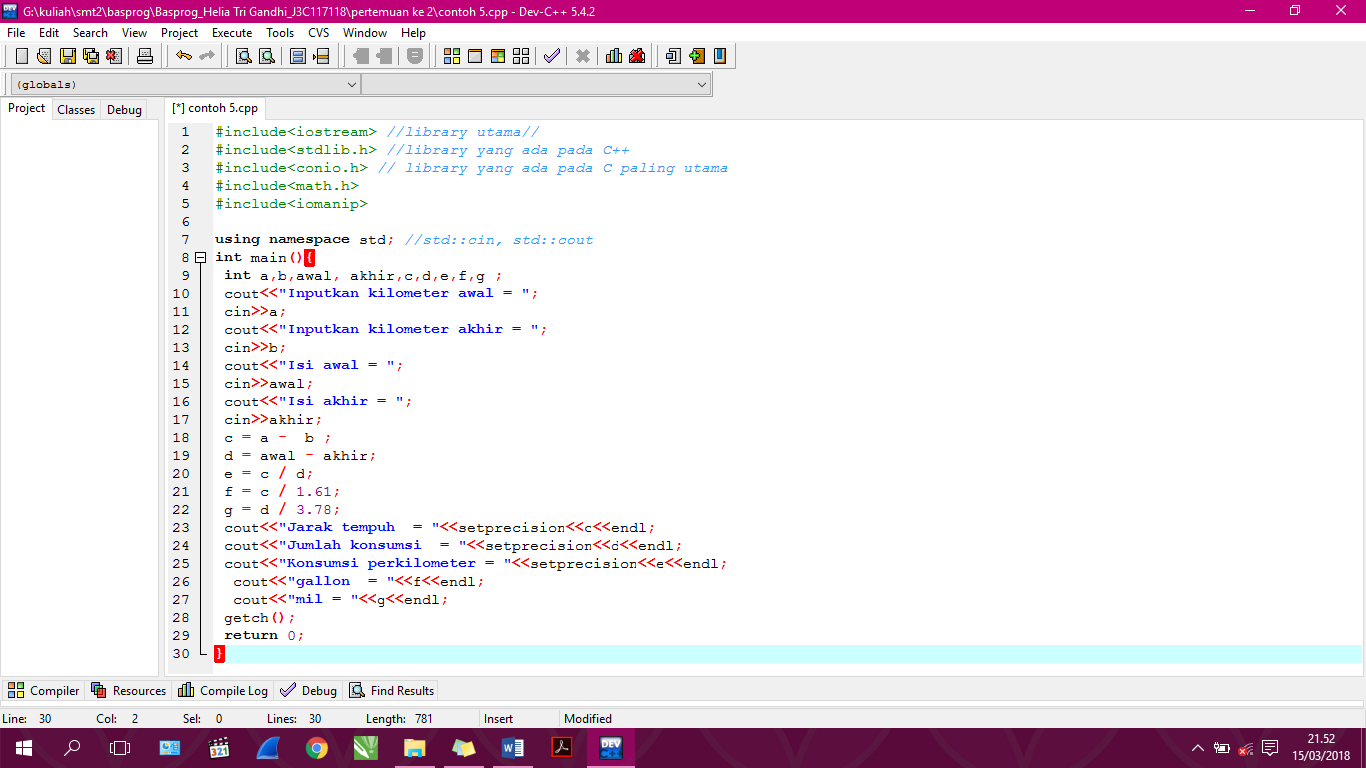
* cout : dengan menggunakan cout programmer dapat meletakan suatu informasi ke standart output(berupa layar). Biasanya diikuti ganti baris atau newline.

ex : cout**<<**”hello \n”;

* cin : digunakan untuk membaca data dari standart input, disertai dengan operator prngambilan **>>**.
* manipulator

|  |  |
| --- | --- |
| endl | Menyisipkan karakter newline |
| setw() | Mengatur lebar tampilan data |
| setfill() | Mengatur karakter yang di gunakan memenuhi bagian field yang ditentukan oleh setw() |
| dec,oct,hex | Untuk menampikan suatu data dalam bentuk desimal,oktal,hexadesimal |
| setbase() | Dapat digunakan untuk mengkonversi desimal,oktal,hexadesimal |
| setprecision() | Untuk mengatur jumlah digit pecahan yang ingin ditampilkan |
| getch() | Digunakan untuk memaca karakter tanpa perlu menekan enter |





**PERTEMUAN 3 : PERCABANGAN**

**<<<**

* Dalam C++ percabangan digunakan untuk memecahkan persoalan dan mengambil keputusan dari beberapa pernyataan.
* Ada beberapa jenis percabangan yang bisa kita gunakan dalam pemrograman C++ diantaranya sebagai berikut:

1. **If**

--digunakan jika hanya ada satu kondisi

**if (condition)**

**Statement;**

Statement diatas akan dilakukan jika kondisinya bernilai true(tidak sama dengan nol). Apabila statement lebih dari satu maka :

**if (condition)**

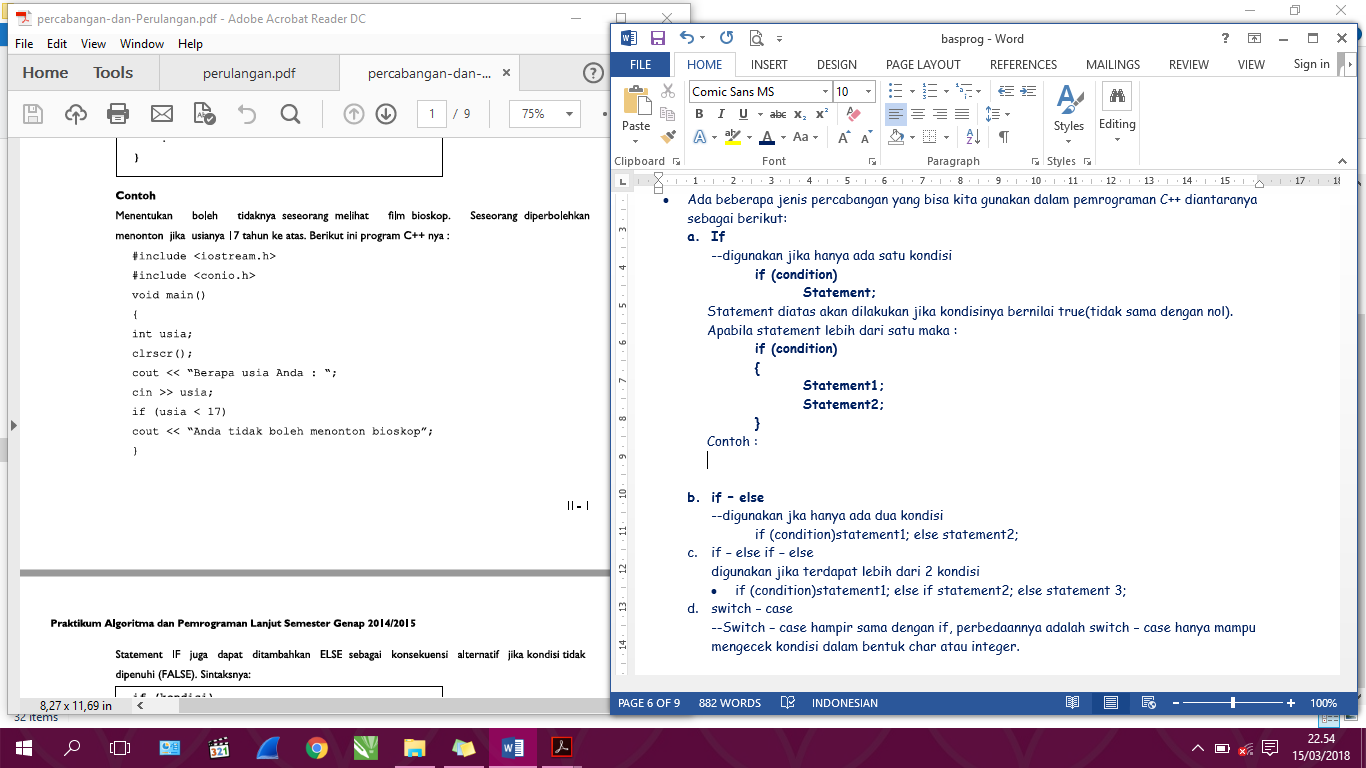
**{**

**Statement1;**

**Statement2;**

**}**

Contoh :



1. **if – else**

--digunakan jka hanya ada dua kondisi, ditambahkan else sebagai konsekuensi alternatif jika kondisi tidak terpenuhi (false)

**if (condition)**

**{**

**Statement1;**

**Statement2;**

**}**

**else**

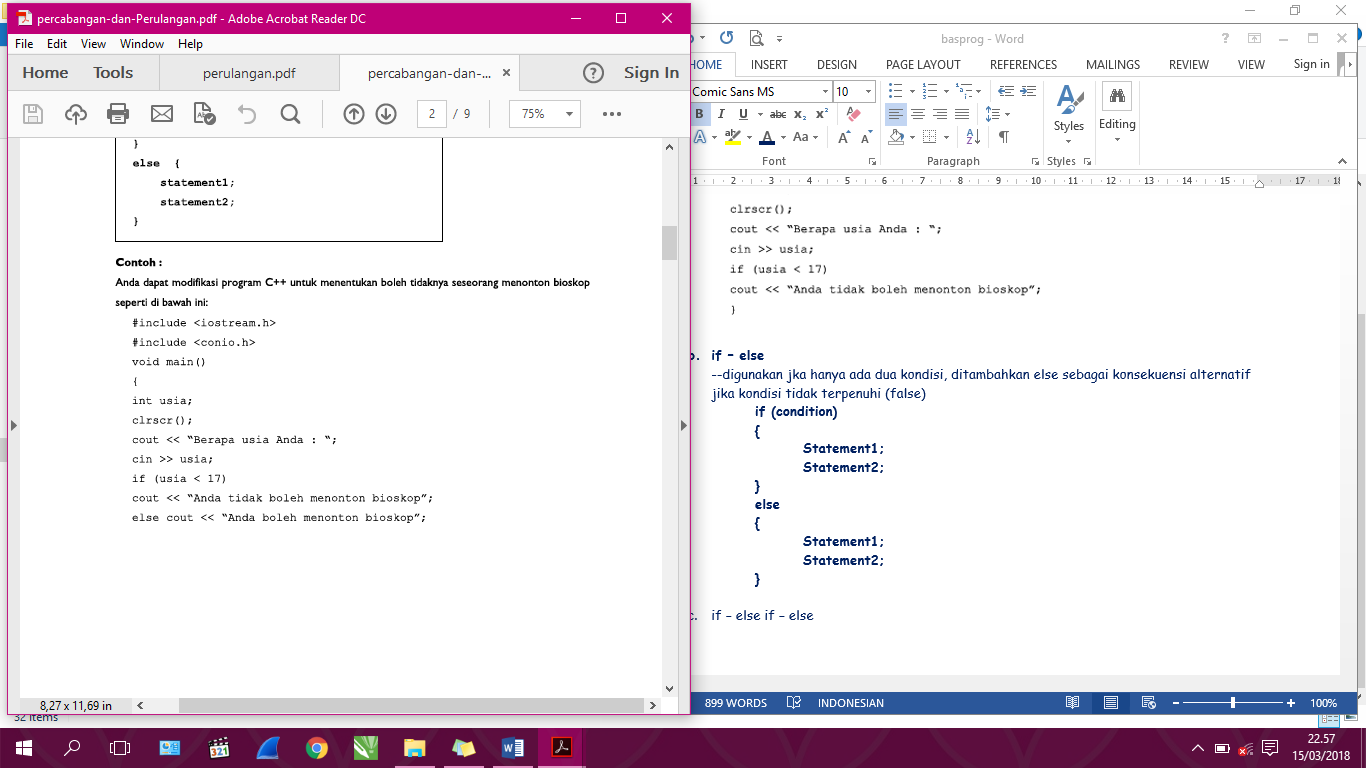
**{**

**Statement1;**

**Statement2;**

**}**

Contoh :



1. **switch – case**

--Switch – case hampir sama dengan if, perbedaannya adalah switch – case memperbolehkan percabangan pada multiple outcomes.

**switch(switch\_ekspresi)**

**{**

**case case\_pilihan1:**

**statement1;**

**statement2; //blok1**

**break;**

**case case\_pilihan2:**

**statement1;**

**statement2; //blok2**

**break;**

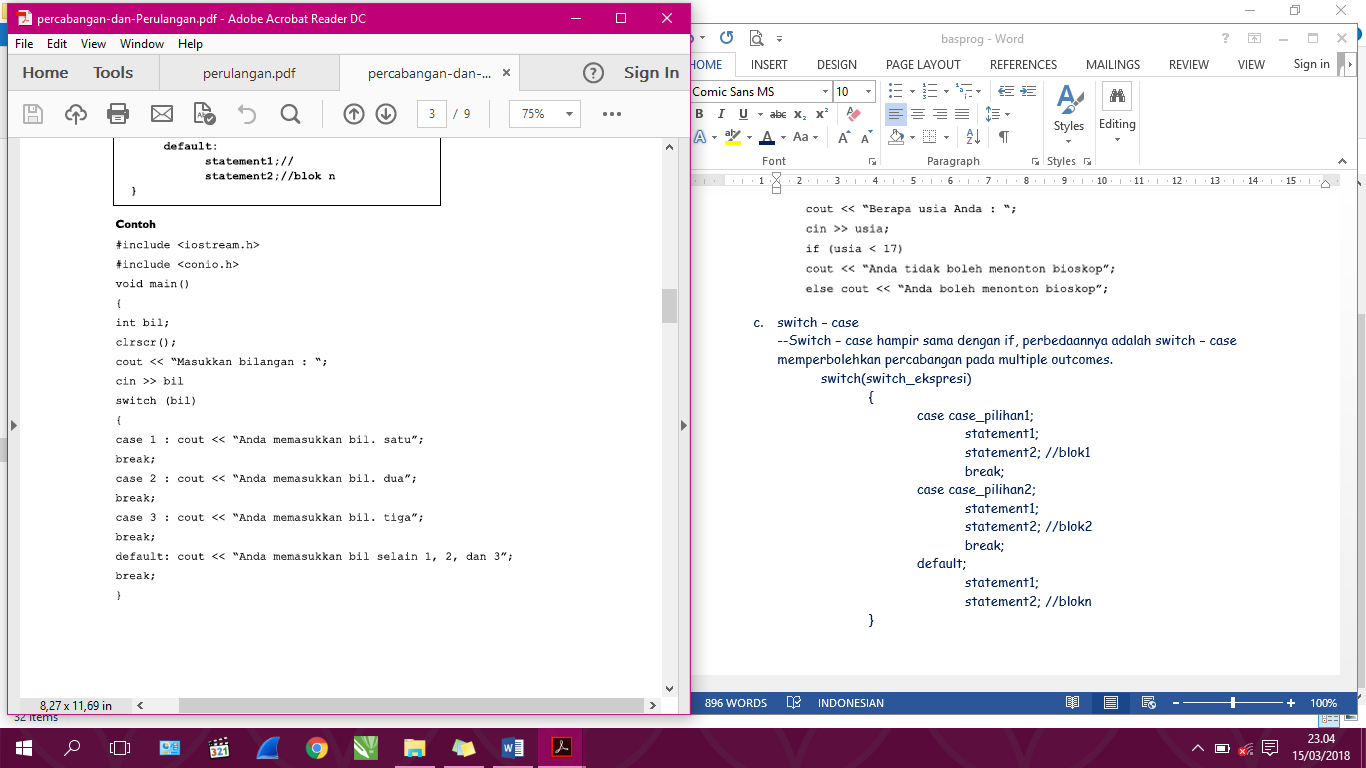
**default:**

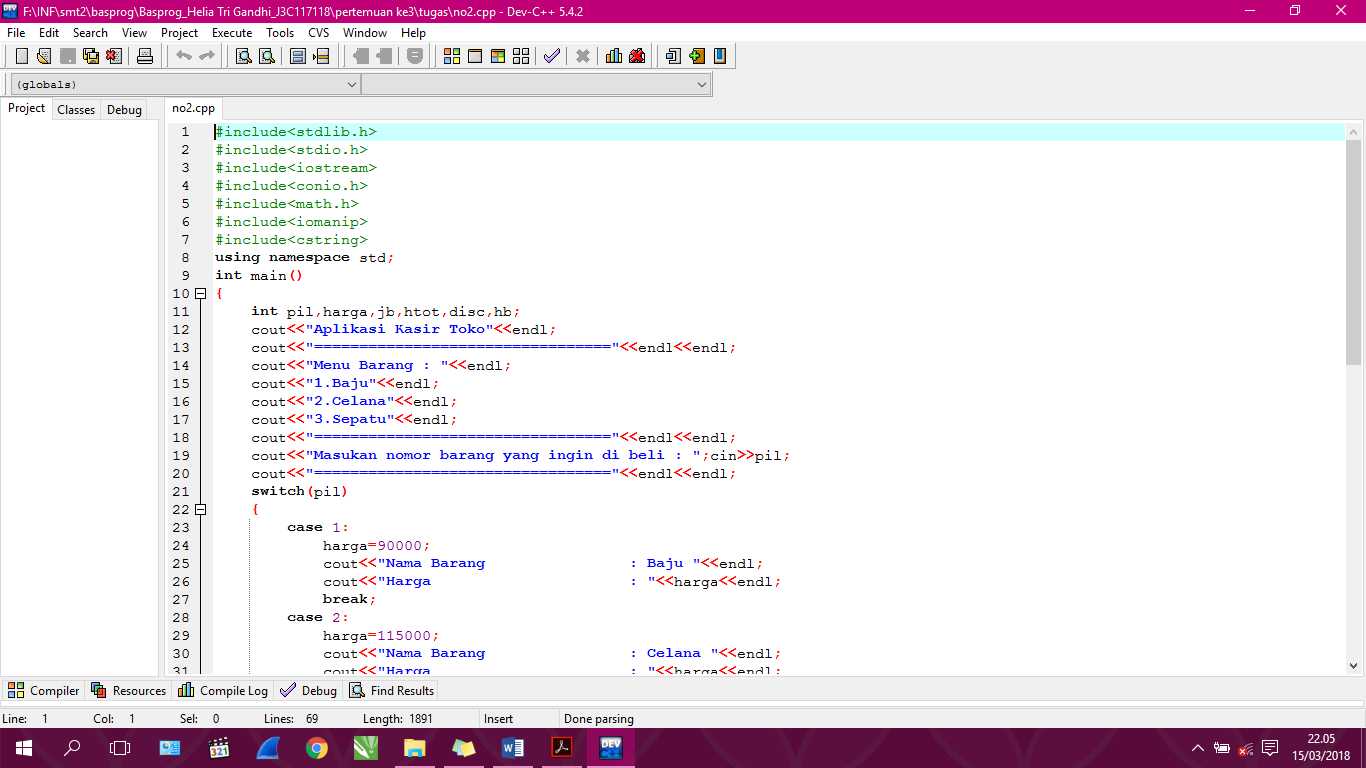
**statement1;**

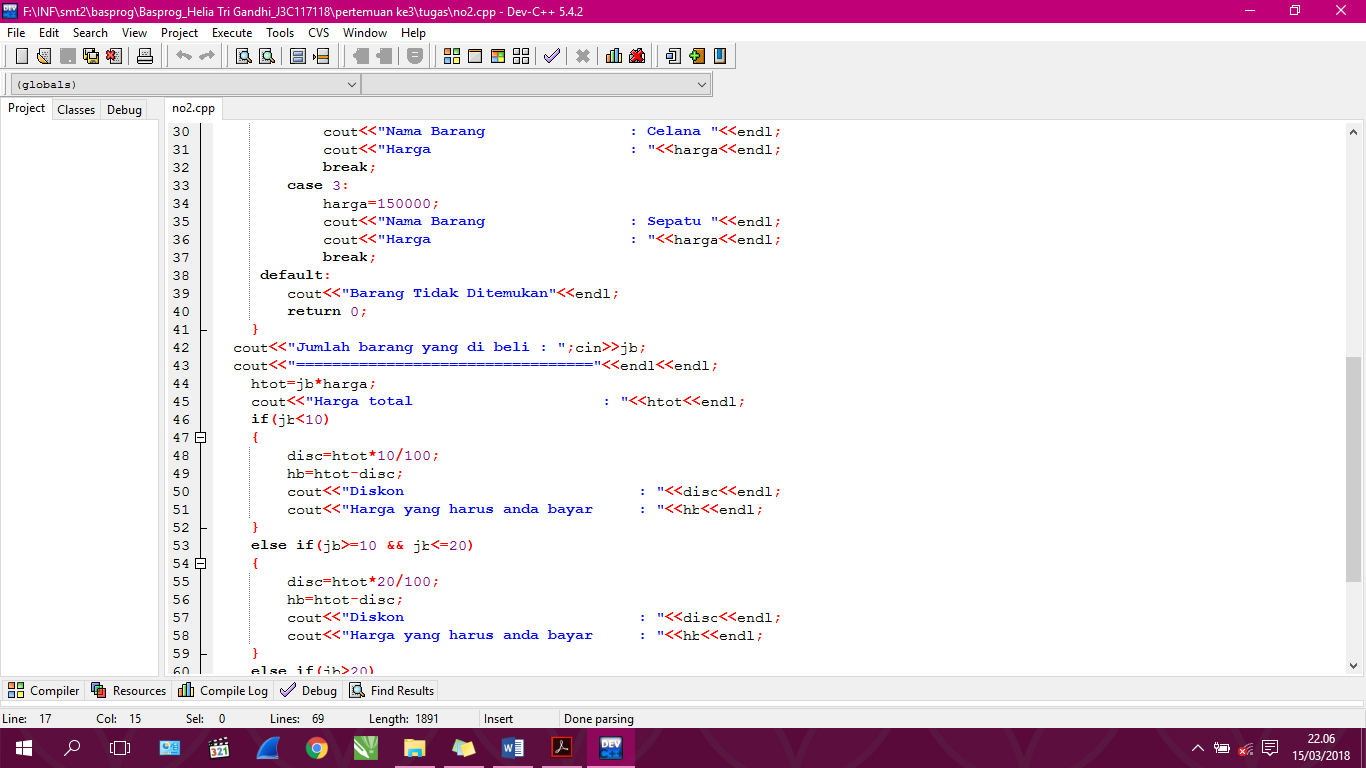
**statement2; //blokn**

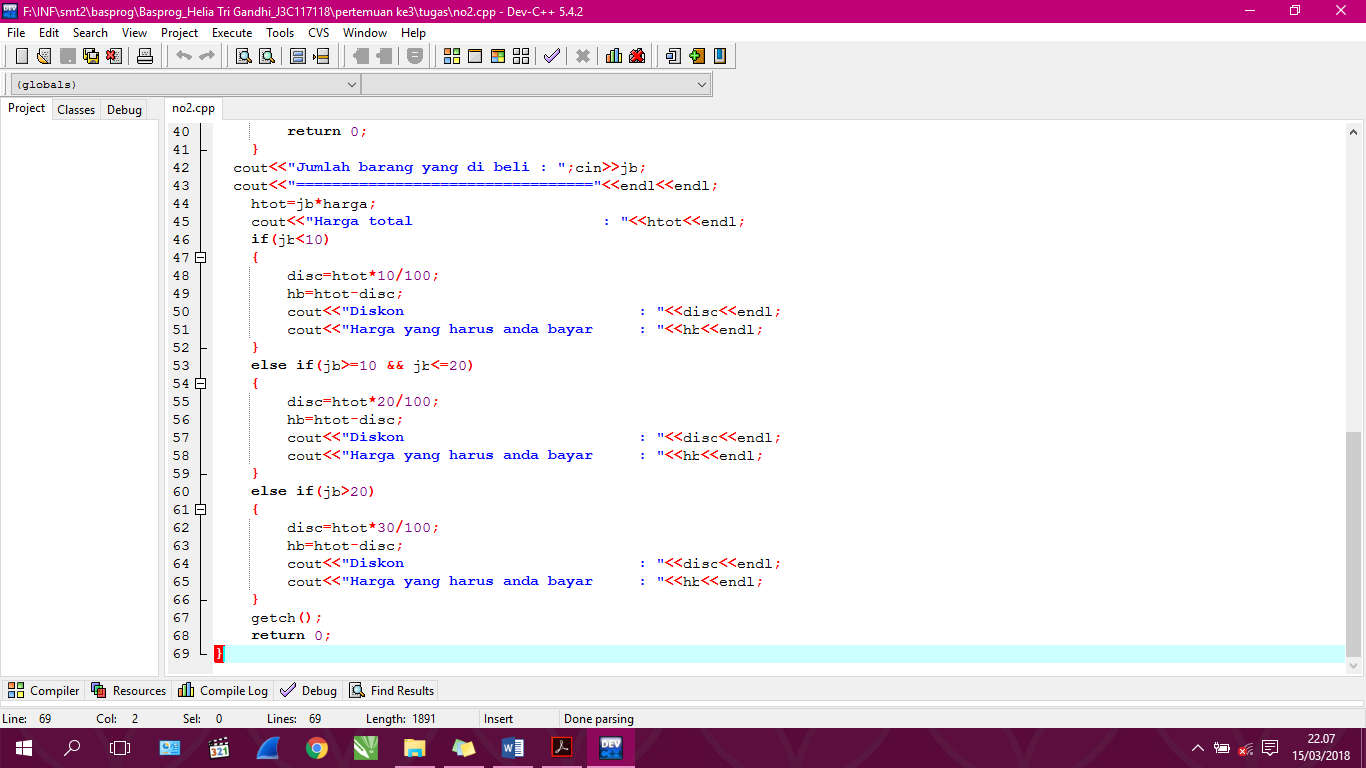
**}**

Contoh :









**PERTEMUAN 4 : PERULANGAN/LOOPING**

**>>>**

* **FOR –** digunakan untuk menghasilkan perulangan sejumlah kali yang dispesifikasikan. Jumlah perulangan diketahui atau dapat ditentukan sebelum eksekusi.

**for (ungkapan1; ungkapan2; ungkapan3)**

**{**

**Blok Pernyataan 1**

**Blok Pernyataan n**

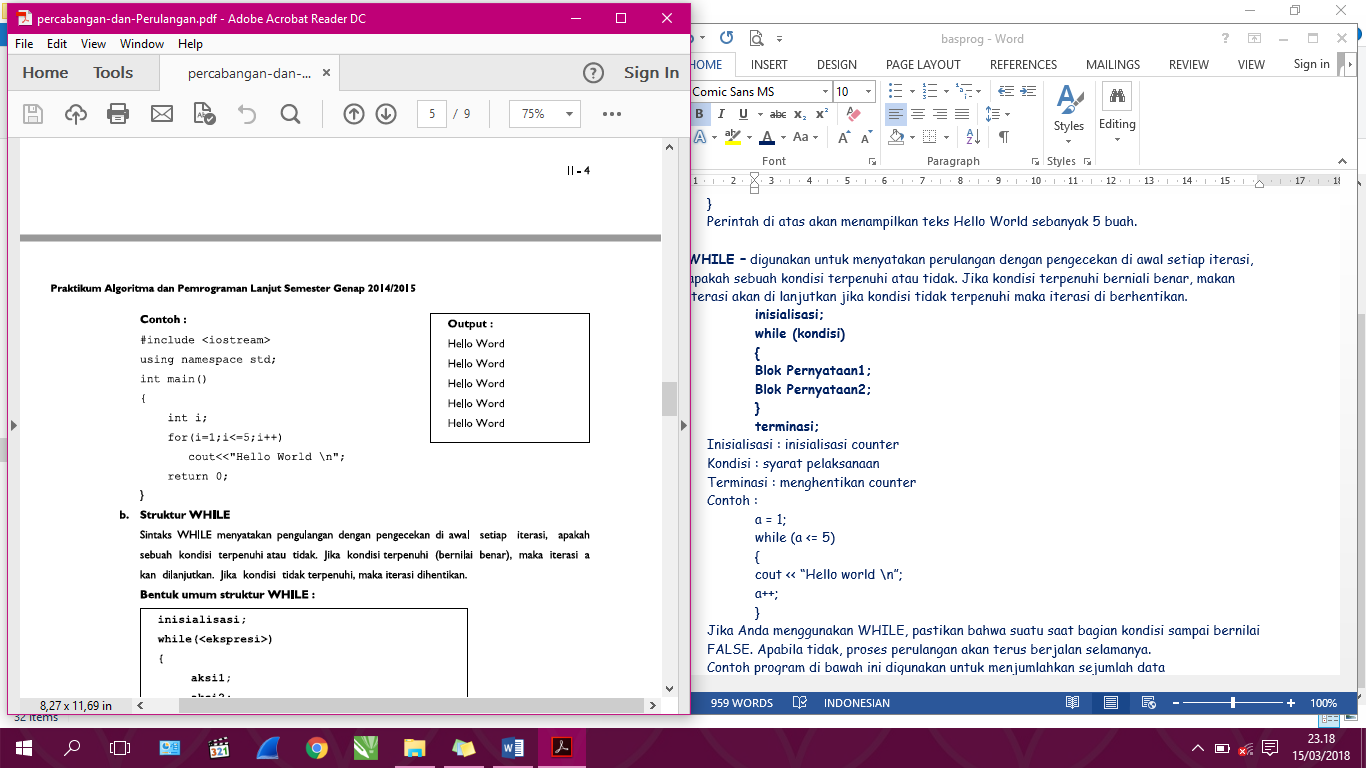
**}**

Ungkapan1 merupakan statement awal (inisialisasi)

Ungkapan2 merupakan kondisi/syarat perulangan dilakukan

Ungkapan3 merupakan statement control untuk perulangan (increment / decrement)

Blok Pernyataan merupakan pernyataan yang dijalankan jika syarat terpenuhi.



* **WHILE –** digunakan untuk menyatakan perulangan dengan pengecekan di awal setiap iterasi, apakah sebuah kondisi terpenuhi atau tidak. Jika kondisi terpenuhi berniali benar, makan iterasi akan di lanjutkan jika kondisi tidak terpenuhi maka iterasi di berhentikan.

**inisialisasi;**

**while (kondisi)**

**{**

**Blok Pernyataan1;**

**Blok Pernyataan2;**

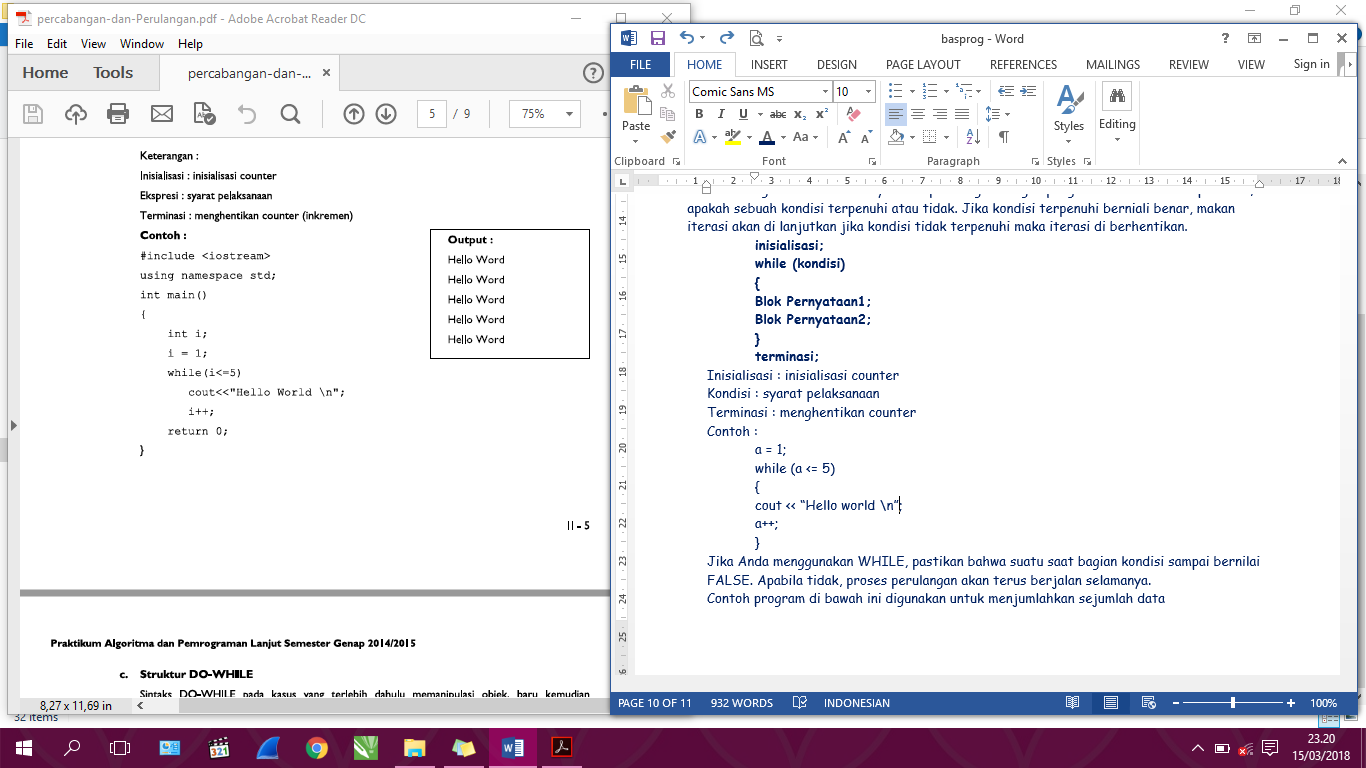
**}**

**terminasi;**

Inisialisasi : inisialisasi counter

Kondisi : syarat pelaksanaan

Terminasi : menghentikan counter



Jika Anda menggunakan WHILE, pastikan bahwa suatu saat bagian kondisi sampai bernilai FALSE. Apabila tidak, proses perulangan akan terus berjalan selamanya.

Contoh program di bawah ini digunakan untuk menjumlahkan sejumlah data

angka. Angka yang akan dijumlahkan diinputkan satu-persatu. Proses pemasukan

data angka akan berhenti ketika dimasukkan angka –1. Setelah itu tampil hasil

penjumlahannya.

While juga dapat menggunakan operator logika

**while ((kondisi1) && (kondisi2))**

**{**

**Blok Pernyataan**

**}**

* **Do...While –** terlebih dahulu memanipulasi objek, baru kemudian memeriksa kondisi objek tersebut.

**inisialisasi;**

**do**

**{**

**Blok Pernyataan**

**}**

**while (kondisi);**

**terminasi;**

Perbedaan dengan WHILE sebelumnya yaitu bahwa pada DO WHILE statement perulangannya dilakukan terlebih dahulu baru kemudian di cek kondisinya. Sedangkan WHILE kondisi dicek dulu baru kemudia statement perulangannya dijalankan. Akibat dari hal ini adalah dalam DO WHILE minimal terdapat 1x perulangan. Sedangkan WHILE dimungkinkan perulangan tidak pernah terjadi yaitu ketika kondisinya langsung bernilai FALSE.

Contoh:

a = 1;

do

{

cout << “Hello world \n”;

a++;

}

while(a==0)

Perintah di atas akan muncul satu buah Hello World. Bandingkan dengan yang

berikut ini:

a = 1;

while(a==0)

{

cout << “Hello world \n”;

a++;

}

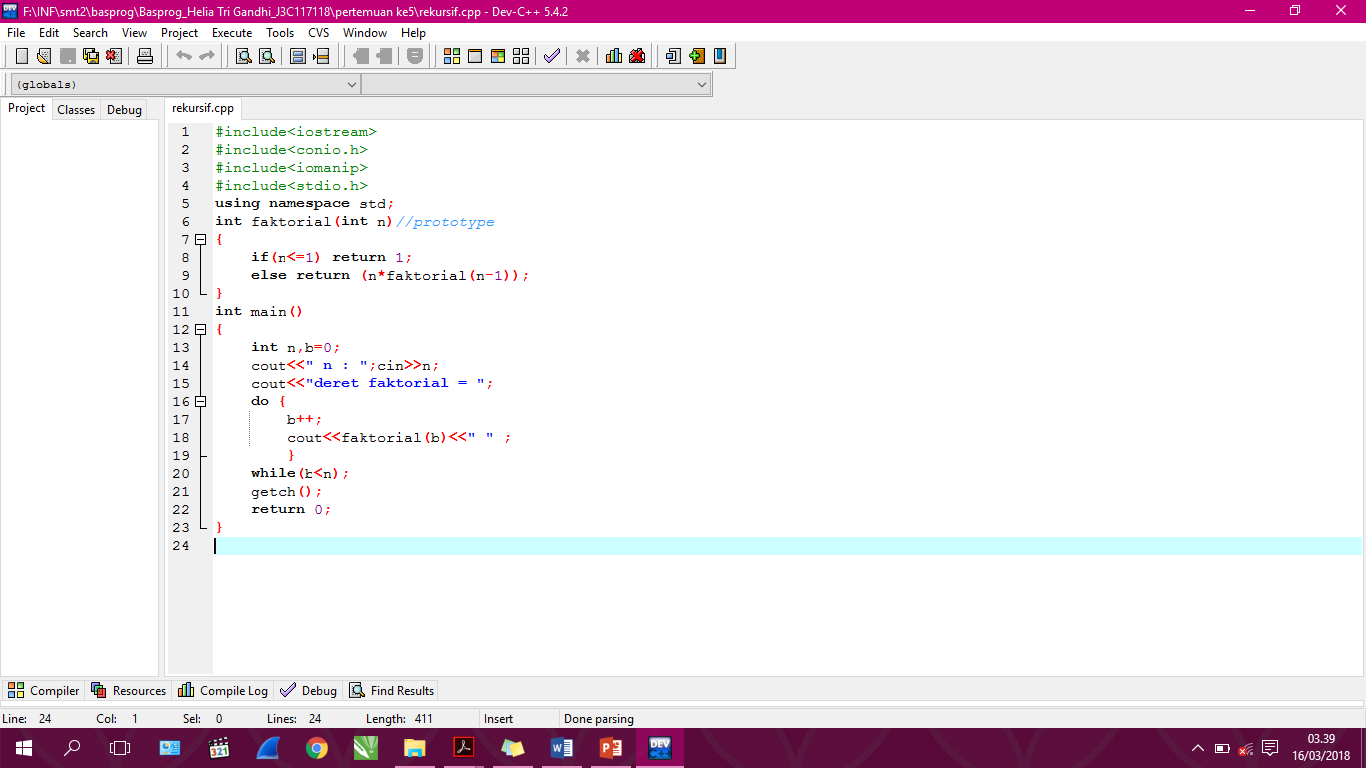
Perintah di atas sama sekali tidak menampilkan Hello World, karena kondisinya langsung FALSE.

**PERTEMUAN 5: FUNCTION,STRUCT,CLASS**

**>>>**

**Fungsi rekursif :** fungsi yang telah di bahas sebelumnya di panggil dari bagian lai di luar tubuh fungsi yag bersangkutan atau dengan kata lain fungsi yag memanggil dirinya sendiri.

Contoh :



**Function :** satu blok kode yang melakukan tugas tertentu atau satu blok instruksi yang di eksekusi ketika dipanggil dari bagian lain dalam suatu program.

Bentuk umum deklarasi fungsi :

**Tipenilaibalik fungsi (tipepara, … ) ;**

Keterangan :

- tipenilaibalik = tipe nilai yang dikembalikan dengan statemen ‘return’.

Tipe default nya : ‘int’. Untuk menyatakan fungsi yang tidak mengembalikan nilai balik, dideklarasikan sebagai : ‘void’

- fungsi = nama fungsi tersebut

- tipepara = tipe parameter, bila parameter lebih dari satu (1), masing-masing

dipisahkan dengan tanda koma ( , ) untuk menyatakan fungsi tanpa parameter dispesifikasikan : ‘void’. Bila tipe parameter tidak dispesifikasikan, defaultnya : ‘void’

Fungsi harus dideklarasikan terlebih dahulu sebelum didefinisikan. Maksudnya adalah memberitahu compiler jumlah dan tipe parameter yang diterima dan nilai balik fungsi (bila ada) agar compiler dapat memeriksa ketepatannya. Definisi fungsi itu sendiri adalah menspesifikasikan tugas fungsi tersebut.

Contoh :

# include <iostream.h>

Int tambah (int a, int b )

{

Int r ;

r = a + b ;

return 0 ;

}

Int main ( )

{

Int z ;

z = tambah ( 5,3 ) ;

cout << “Hasilnya = ‘ << z ;

return 0 ;

}

**Pemanggilan fungsi call by value / call by reference**

* **Call by value**

Pemanggilan nilai terjadi jika yang dikirimkan ke fungsi oleh pemanggilnya adalah berupa nilai, dan tidak berpengaruh pada variabel yang tertulis di dalam argumen pemanggilnya.

* **Call by reference**

Pemanggilan nilai terjadi jika yang dikirimkan ke fungsi oleh pemanggilnya adalah berupa referensia(alamat), dan diterima oleh fungsi sehingga perubahan terhadap variabel penerima ini akan mempengaruhi variabel argument yang memanggilnya

Contoh :

#include<iostream>

#include<conio.h>

#include<stdio.h>

using namespace std;

//CALL BY VALUE

int kali(int a,int b, int c)//parameter formal

{

int x;

x=a\*b\*c;

return x;

}

//CALL BY REFERENCE

void urut(int \*a, int\*b)

{

if(\*a<\*b)

{

\*a=\*a+\*b;

\*b=\*a-\*b;

\*a=\*a-\*b;

}

}

int main()

{

int a,b,c;

cout<<"CALL BY VALUE"<<endl;

cout<<"A : ";cin>>a;

cout<<"B : ";cin>>b;

cout<<"c : ";cin>>c;

cout<<"hasil perkalian ketiga bilangan adalah : "<<kali(a,b,c)<<endl;//parameter aktual

cout<<"CALL BY REFERENCE"<<endl;

cout<<"A : ";cin>>a;

cout<<"B : ";cin>>b;

cout<<"c : ";cin>>c;

urut(&a,&b);

urut(&a,&c);

urut(&b,&c);

cout<<"hasil descending adalah : "<<a<<" "<<b<<" "<<c<<endl;

getch();

return 0;

}

**>>>**

**Struct :** kumpulan data yang memiliki tipe data yang berbeda.

Misal : kita ingin membentuk tipe data struktur yang namanya kotak. Maka dapat dideklarasikan sebagai berikut :

struct nama\_struktur

{

elemen1;

elemen2;

};

Atau

typedef struct

{

elemen1;

elemen2;

}nama\_struct;

Contoh :



**>>>**

**Class :** suatu tipe data yang di dalamnya berisi data dan sebuah fungsi. Jika class mendefinisikan class yang sudah ada maka class baru tersebut disebut kelas turunan.

class namaClass

{

private:

members1;

protected:

members2;

public:

members3;

};

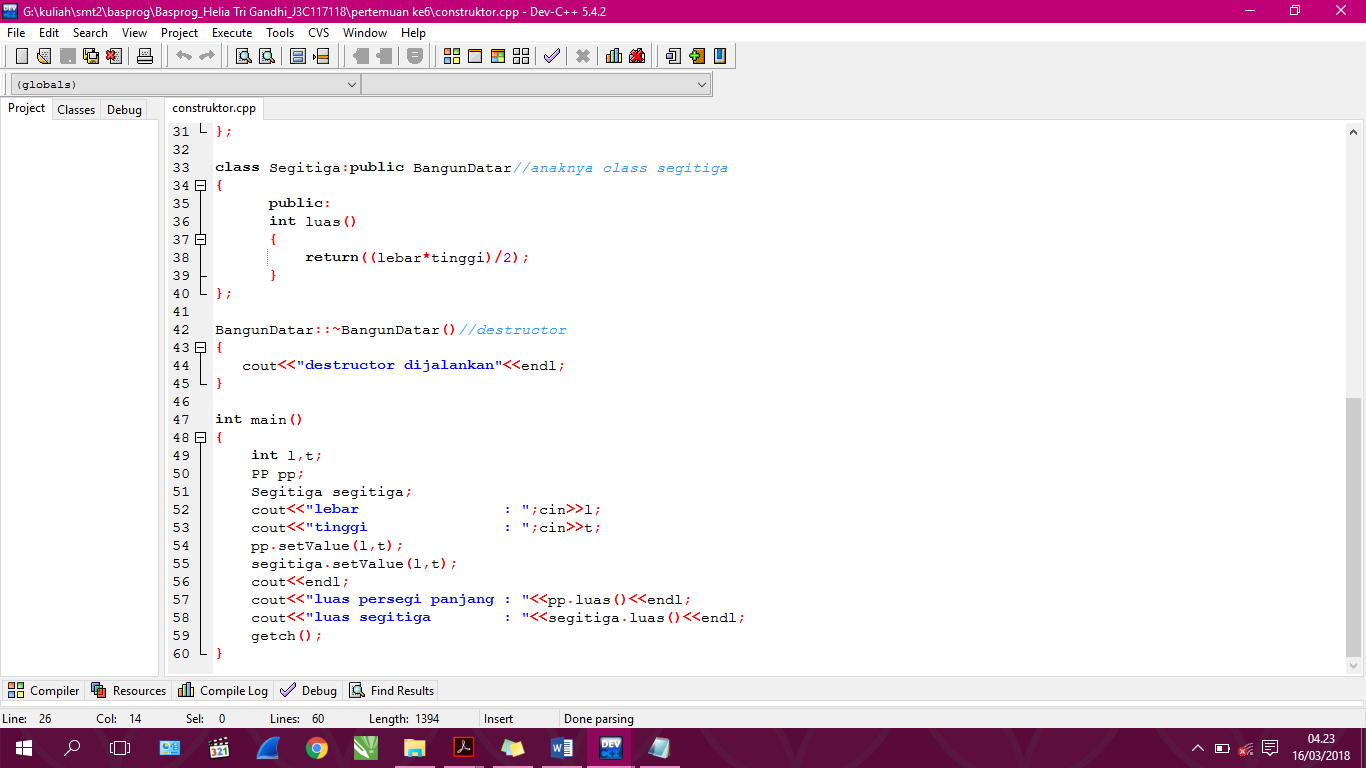
private : anggota kelas dapat diakses dari anggota lain pada kelas yang sama atau dari kelas "friend"

protected : anggota dapat diakses dari anggota class yang sama atau class friend,dan juga dari anggota class turunannya(derived)

public : anggota dapat diakses dari kelas manapun

Contoh :







**PERTEMUAN 6: INHERITANCE,POLYMORFISME**

**(CONSTRUKTOR,DESTRUCTOR,OVERLOAD,OVERRIDE)**

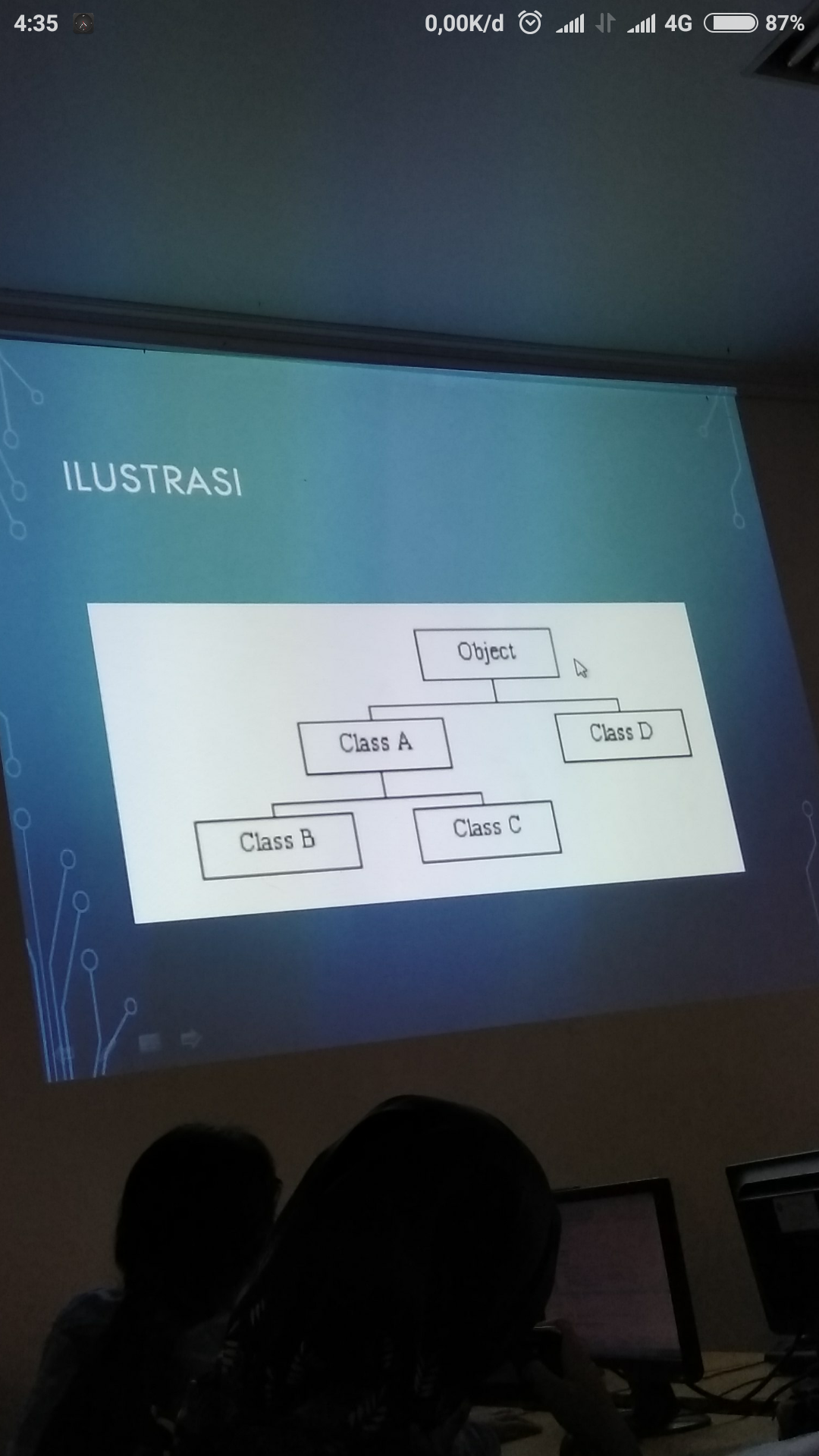
**<<<**

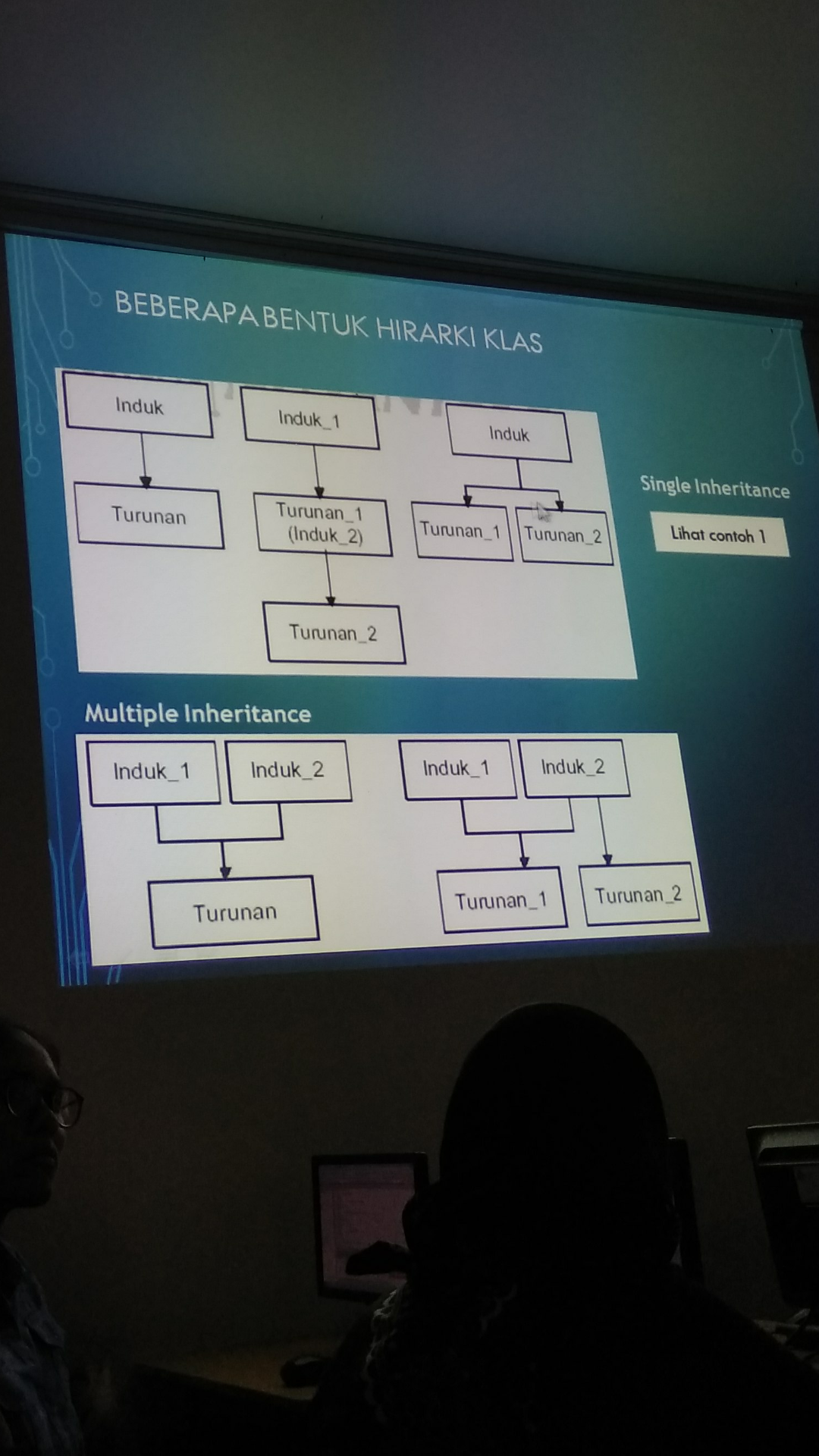
**Inheritance :**proses pewarisan data dan method dari suatukelas kepada kelas lain.

* Pewarisan bersifat menyeluruh.
* Semua data dan method milik kelas asalnya diturunkan ke kelas baru.
* Kelas yang mewariskan sifat kelas lain 🡪 kelas dasar (base class)
* Kelas yang diwariskan sifat kelas lain 🡪 kelas turunan(derived class)

Keuntungan Pewarisan

* Bersifat reusable :Tidak harus menyalin semua data dan method dari suatu kelas jika akan menggunakannya lagi
* Kemudahan dalam me-manage kelas yang memiliki data dan method yang sama Untuk memodifikasi suatu data atau method untuk semua subkelas / kelas anak, maka tidak perlu melakukan perubahan di masing-masing kelas anak melainkan hanya pada kelas induk saja.





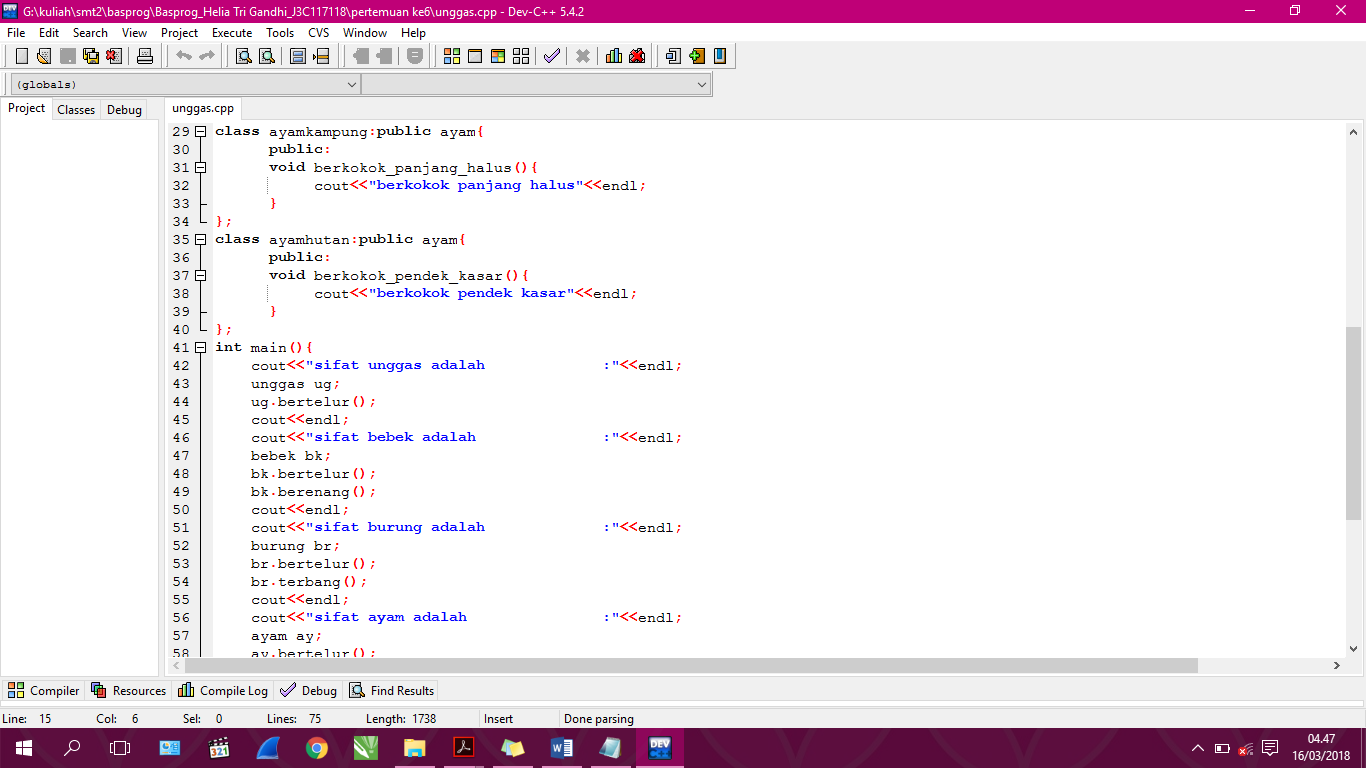
**<<<**

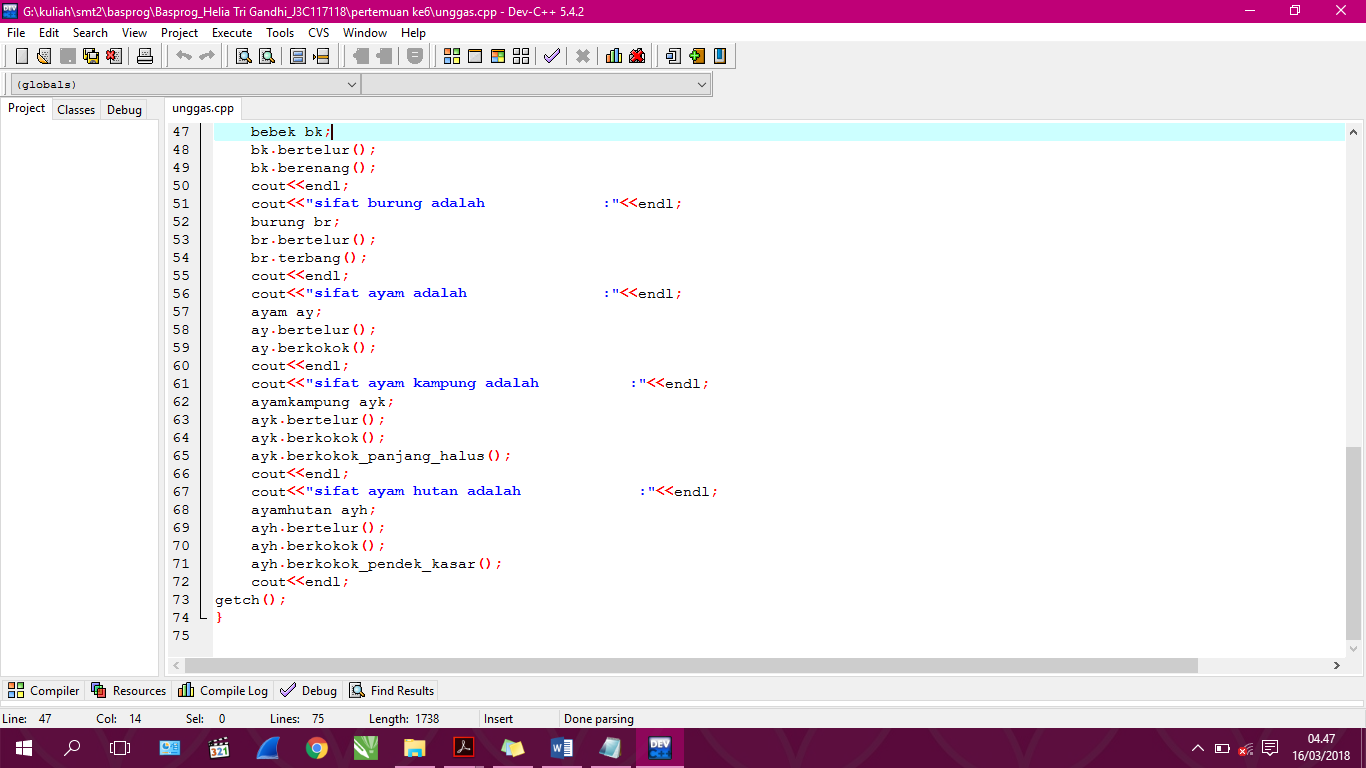
**Polimorfisme :** sesuatu yang memiliki banyak bentuk.

* Dalam pemrograman, polimorfisme dapat diartikan sebagai modul yang memiliki nama sama, namun memiliki behaviour (tingkah laku) yang berbeda sehingga listing code implementasinya juga berbeda
* Polimorfik artinya mempunyai banyak bentuk atau punya kemampuan untuk mendefinisi banyak bentuk.

Contoh :







**<<<**

**Constructor :** menthod khusus yang akan dijalankan secara otomatis pada saat sebuah object dibuat. Biasa digunakan untuk membuat proses awal dalam mempersiapkan objek dan dapat melakukan alokasi memori.

Aturan :

* Nama constructor harus sama dengan nama kelas yang mendefinisikannya
* Constructor tidak memiliki nilai balik
* Constructor harus diletakan pada bagian public
* satu kelas memiliki lebih dari constructor(overloading constructor)
* suatu constructor dapat di panggil dengan constructor lain dalam satu kelas
* tidak mengembalikan suatu nilai (tidak ada keyword return)
* dapat ditambah acces modifier, public, private, protected maupun default

**Destructor : (diawali dengan ~ )** menthod khusus yang akan dijalankan secara otomatis pada saat sebuah object akan sirna atau hilang. Biasa digunakan untuk membersihkan beberapa variabel, atau menjlankan proses tertentu sebelum objek di hapus.

Contoh :





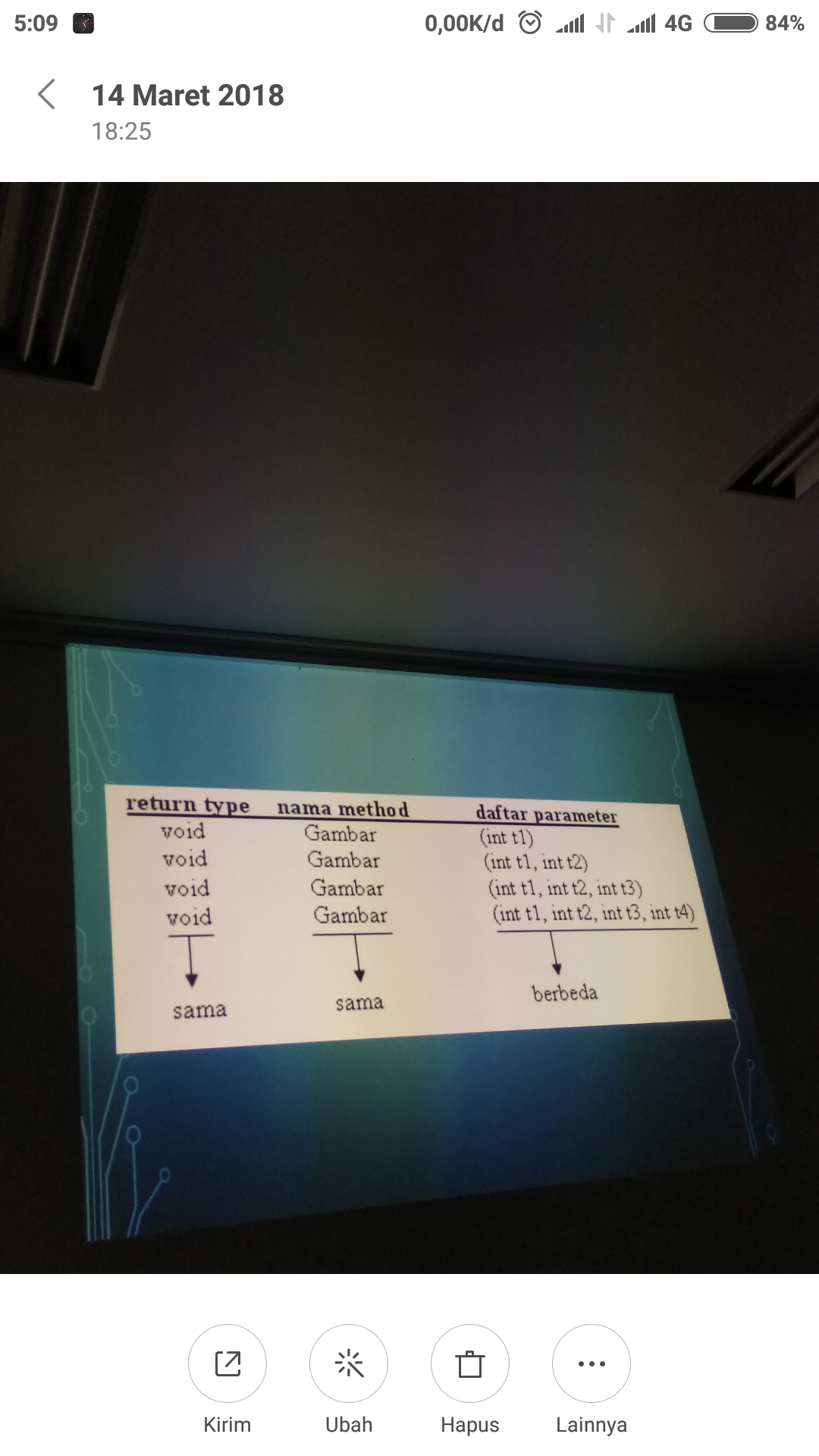
**<<<**

**Overloading function:** penggunaan kembali nama fungsi yang sama tetap dengan argument yang berbeda.

Tujuan :

Memudahkan penggunaan atau pemanggilan method dengan fungsiolitas yang mirip.

* Nama menthod harus sama
* Daftar parameter harus berbeda
* Returntype boleh sama atau berbeda



**Overriding function:** sebuah fungsi dalam class turunan yang memiliki nama,return type,dan argument function yang sama dengan fungsi dalam class induk.

* Nama menthod harus sama
* Daftar parameter harus sama
* Returntype boleh sama

Contoh :

