PEMROGRAMAN BERORIENTASI OBJEK

Input / Output

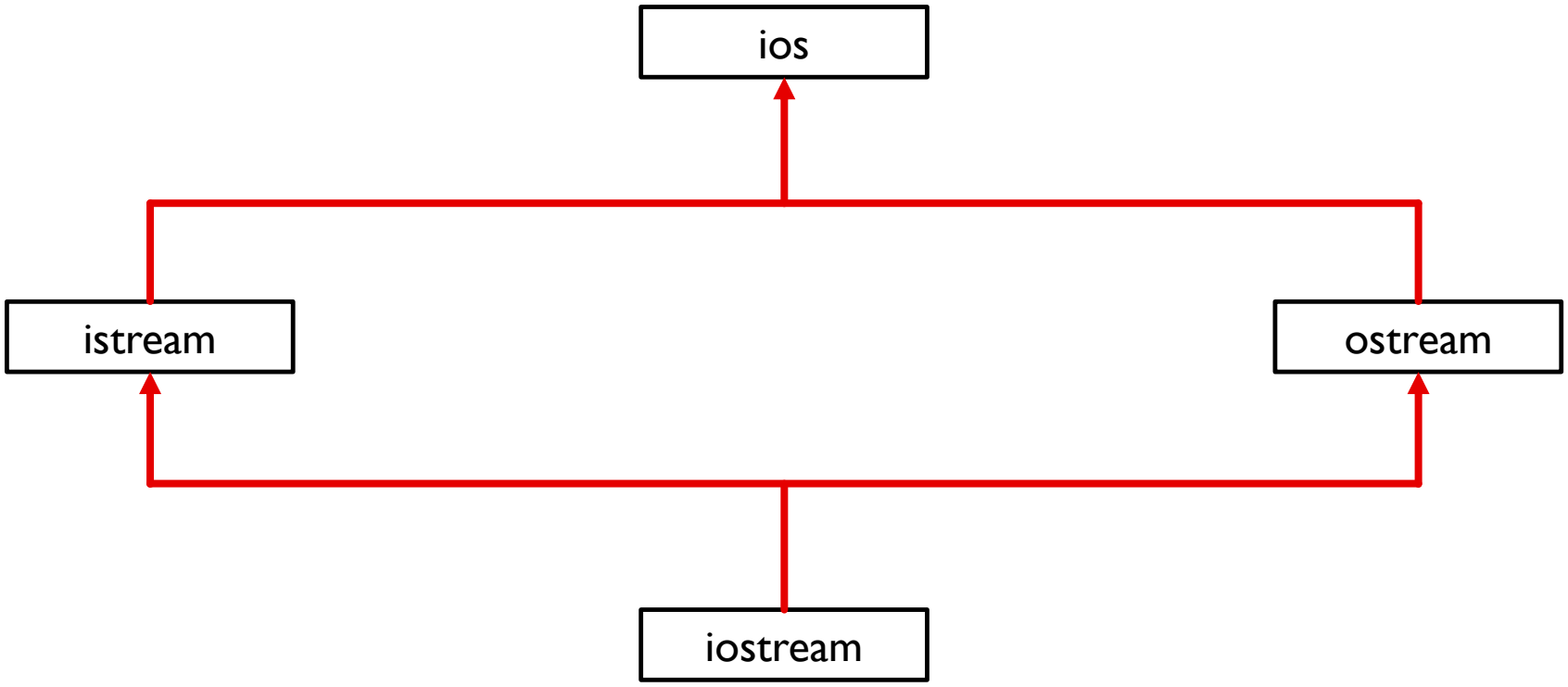
Telkom University

Budhi Irawan, S.Si, M.T

PENDAHULUAN

- Sejauh ini sudah sering digunakan **`cout`** *untuk menuliskan ke layar* dan **`cin`** *untuk membaca nilai dari keyboard* tanpa membahas lebih detail mengenai kegunaan dari sistem input dan output (I/O) tersebut.
- Sistem I/O yang terdapat didalam C++ sangatlah kompleks sehingga perlu sekali untuk mengetahui dasar-dasarnya terlebih dahulu.

HIRARKI KELAS PROSES I/O



HIRARKI KELAS PROSES I/O

- Pada gambar diatas dapat dilihat bahwa ***ios*** merupakan *kelas dasar* ***(virtual base class)*** yang berisi *fasilitas untuk proses input dan output data*.
- Didalam kelas ini juga didefinisikan anggota-anggota yang dapat digunakan untuk proses penentuan format data dalam ***proses input maupun output***.
- Dalam C++, kelas ***ios*** dijadikan sebagai kelas dasar dari kelas ***istream (Input Stream)*** dan ***Ostream (Output Stream)***

HIRARKI KELAS PROSES I/O

- Kelas ***istream*** adalah *kelas yang dibuat khusus untuk menangani masalah – masalah input dengan mengekstrak fasilitas – fasilitas input yang terdapat pada kelas* ***ios*** dan tentunya juga melalui penambahan – penambahan fasilitas lainnya.
- Kelas ***ostream*** adalah *kelas yang digunakan untuk menangani masalah – masalah output*.

HIRARKI KELAS PROSES I/O

- Dari kedua kelas tersebut kemudian dibuat lagi sebuah kelas baru yang dinamakan ***iostream***.
- Oleh karena diturunkan dari dua buah induk, maka *kelas **iostream** ini otomatis dapat menangani masalah – masalah yang bisa ditangani oleh kelas **istream** dan kelas **ostream***.
- Hal inilah yang menyebabkan *kita menggunakan **iostream** sebagai standar untuk melakukan operasi input dan output (I/O) data*.

STREAM

- ***Stream*** adalah *suatu* ***peralatan logika (logical device)*** *yang berguna untuk mendapatkan atau memberikan informasi*.
- ***Stream*** akan dihubungkan ***dengan peralatan fisik (seperti keyboard, screen ataupun printer) melalui sistem I/O***.
- Semua ***stream*** mempunyai prilaku yang sama sehingga fungsi I/O dapat dioperasikan ke peralatan fisik yang berbeda.

STREAM

- Sebagai contoh, jika akan melakukan penulisan data, maka cara yang digunakan untuk menuliskan ke screen ataupun ke printer adalah sama.
- Dalam bahasa C, untuk melakukan hal – hal yang berhubungan dengan proses input dan output data digunakan ***file header <stdio.h>***, namun dalam C++, file header standar yang digunakan adalah ***<iostream>.***

STREAM

- Pada saat program C++ memulai proses eksekusi terdapat ***empat buah stream*** yang secara otomatis akan terbuka

Nama Stream	Kegunaan	Peralatan Standar
Cin	Input standar	Keyboard
Cout	Output standar	Layar (screen)
Cerr	Kesalahan output standar	Layar (screen)
clog	Cerr yang terbuffer melalui file log	Layar (screen)

INPUT MENGGUNAKAN CIN

- Seperti telah dikemukakan sebelumnya bahwa **`cin`** adalah *suatu **stream** yang akan merespon proses input yang dilakukan*.
- ***Stream*** ini hanya tersedia jika dimasukan file ***header*** <iostream> didalam program yang dibuat.

INPUT MENGGUNAKAN CIN

```
#include <iostream>

using namespace std;

int main() {

  // Mendeklarasikan variabel yang bertipe int
  int X;

  // Memberikan informasi kepada user untuk melakukan input
  cout<<"Masukkan sebuah bilangan bulat :";

  // Menggunakan cin untuk merespon (membaca) input yang akan dilakukan
  cin>>X;

  // Menampilkan nilai X
  cout<<"Nilai X: "<<X;

  return 0;
}
```

OUTPUT MENGGUNAKAN COUT

- Dalam C++, ***stream* `cout`** *digunakan untuk mencetak output ke peralatan standar (screen).*
- Operator yaang digunakan adalah operator **<<**, sama halnya dengan operator **>>** yang digunakan dalam ***stream* `cin`**, operator **>>** juga telah di-***overload*** sehingga dapat digunakan untuk ***mencetak nilai*** dari berbagai macam ***tipe data ke layar***.

OUTPUT MENGGUNAKAN COUT

```
#include <iostream>

using namespace std;

int main() {

   int X = 100;

   // Melakukan output terhadap nilai X
   cout<<X;

   return 0;
}
```

OUTPUT MENGGUNAKAN COUT

- Program diatas menggunakan kutip tunggal (‘’) untuk menuliskan sebuah karakter spasi, hal ini karen spasi adalah *sebuah karakter, bukanlah string*.
- Walaupun demikian dapat juga digunakan kutip ganda (“”) untuk menuliskan sebuah spasi, namun hal ini dianggap spasi tersebut *sebagai string bukan sebagai karakter*.

MENGATUR FORMAT INPUT/OUTPUT

- C++ mengizinkan untuk mengatur format dari operasi-operasi I/O yang dilakukan sebagai contoh misalnya menentukan lebar kolom dari data yang akan ditampilkan ke layar, menentukan berapa digit angka dibelakang koma, dan yang lainnya.
- Dalam C++ terdapat ***dua buah*** cara untuk melakukan hal ini, cara **pertama** yaitu *diiakses secara langsung anggota dari kelas* ***ios*** dan yang **kedua** *adalah dengan menggunakan fungsi khusus yang disebut dengan manipulator, yang dimasukan sebagai bagian dari ekspresi I/O*.

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

- Kelas ***IOS*** mempunyai sebuah ***tipe enumerasi*** dengan nama **fmflags** yang nilai-nilainya digunakan untuk melakukan penentuan ***format data***. Nilai yang dimaksud adalah :

Adjustfied	basefield	booalpha	dec
Fixed	floated	hex	internal
Left	oct	right	scientific
Showbase	showpoint	showpos	skipws
Unitbuf	uppercase		

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

- Ketika ***flag*** `oct` diaktifkan, maka output akan ditampilkan dalam bilangan oktal (basis 8).
- Apabila kita mengaktifkan ***flag*** `hex`, maka output akan ditampilkan dalam bentuk bilangan hexadecimal (basis 16) dan untuk mengembalikan ke format bilangan desimal (basis 10), maka ***flag*** yang perlu diaktifkan adalah ***flag*** `dec` dan seterusnya.

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

- Setelah mengetahui kegunaan masing-masing ***flag***, maka selanjutnya perlu diketahui cara untuk mengatur atau mengaktifkan flag-flag tersebut.
- Untuk menangani permasalah seperti ini C++ menyediakan fungsi **`setf()`**, yang memiliki bentuk umu sebagai berikut :

```
fmtflags setf(fmtflags flag);
```

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

```
#include <iostream>

using namespace std;

int main() {

  // Mengaktifkan flag uppercase
  cout.setf(ios::uppercase);

  cout<<"Saya Sedang Belajar PBO Menggunakan C++"<<endl;

  // Mengaktifkan flag showpos
  cout.setf(ios::showpos);

  cout<<12;

  return 0;
}
```

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

- Tampak pada hasil di atas bahwa teks "Saya Sedang Belajar PBO Menggunakan C++" akan ditampilkan dalam huruf kapital, hal ini terjadi karena telah diaktifkan ***flag*** **`uppercase`**.
- Selain itu, bilangan 12 akan ditampilkan dengan +12 yang menandakan bahwa bilangan tersebut adalah bilangan positif, ini disebabkan karena telah mengaktifkan ***flag*** **`showpos`** sebelum menggunakan **`cout`**.

MENGGUNAKAN ANGGOTA DARI KELAS *IOS*

- Pada kode di atas, parameter dari fungsi **`setf()`** ditulis dalam bentuk **`ios::uppercase`** dan **`ios::showpos`**.
- Hal ini menunjukan bahwa **`uppercase`** dan **`showpos`** merupakan data anggota (milik) dari kelas **`ios`**.

MENGGUNAKAN MANIPULATOR

- Setelah mengetahui pengaturan format I/O menggunakan cara pertama maka cara kedua yang dapat dijadikan alternatif lain yaitu dengan menggunakan manipulator.
- Dalam C++, terdapat beberapa manipulator yang merupakan fitur baru yang ditambahkan, ini berarti bahwa compiler C++ klasik (belum distandarisasi) tidak mendukung adanya manipulator.

MENGGUNAKAN MANIPULATOR

Manipulator	Kegunaan	Operasi
`boolalpha`	Mengaktifkan flag boolalpha	Input/Output
`dec`	Mengaktifkan flag dec	Input/Output
`endl`	Menampilkan baris baru dan mebuang stream	Output
`ends`	Menampilkan Null	Output
`fixed`	Mengaktifkan flag fixed	Output
`flush`	Membuang Stream	Output
`hex`	Mengaktifkan flag hex	Input/Output
`internal`	Mengaktifkan flag internal	Output
`left`	Mengaktifkan flag left	Output
`oct`	Mengaktifkan flag oct	Input/Output
`right`	Mengaktifkan flag right	Output

MENGGUNAKAN MANIPULATOR

- Untuk menggunakan manipulator yang tercantum diatas maka harus disertakan file header <iomanio>.

```
#include <iostream>
#include <iomanip>

using namespace std;

int main() {

  // Menggunakan flag setfill, setw dan endl
  cout<<setfill('*')<<setw(8)<<12<<endl;

  // Menggunakan flag oct dan endl
  cout<<oct<<64<<endl;

  // Menggunakan flag hex
  cout<<hex<<16<<endl;

  return 0;
}
```

INPUT OUTPUT PADA FILE

- Operasi I/O terhadap file merupakan hal yang sering dijumpai dalam banyak kasus didunia pemrograman.
- Melalui cara yang seperti ini maka dapat menyimpan data yang dimasukan oleh user kedalam file.
- Selain itu juga dapat mengambil data yang tersimpan didalam file untuk diproses dan ditampilkan didalam program sesuai dengan kebutuhan yang sedang dihadapi.

KELAS-KELAS YANG BERHUBUNGAN DENGAN OPERASI FILE

- Untuk menjalankan operasi I/O maka harus dimasukan **`file header <iostream>`**
- Untuk keperluan pengaksesan file terdapat beberapa kelas yang didefinisikan didalamnya, diantaranya **`ifstream, ofstream dan fstream`**, kelas-kelas ini merupakan turunan dari kelas **`istream, ostream dan iostream`**.
- **`istream, ostream dan iostream`** adalah kelas turunan dari **`kelas ios`** yang menyebabkan kelas **`ifstream, ofstream dan fstream`** juga dapat mengakses data dan semua operasi yang terdapat pada **`kelas ios`**.

MEMBUKA DAN MENUTUP FILE

- Dalam C++, membuka file dengan menghubungkan ke sebuah stream, ini berarti bahwa sebelum dapat membuka file maka harus mendapatkan ***stream*** terlebih dahulu.
- Apabila akan dibuat ***variabel stream*** untuk proses input, maka variabel tersebut harus kita deklarasikan dengan tipe **`ifstream`**.
- Begitu juga dengan ***variabel stream*** untuk proses output dan input dan I/O, masing-masing harus dideklarasikan sebagai ofstream dan fstream.

MEMBUKA DAN MENUTUP FILE

- Berikut ini contoh kode yang menunjukan cara deklarasi ***varibel stream*** untuk keperluan pengaksesan file :

```
// variabel stream untuk proses input
ifstream input;
// variabel stream untuk proses output
ofstream output;
// variabel stream untuk proses input/output
fstream inout;
```

MEMBUKA DAN MENUTUP FILE

- Setelah mendeklarasikan ***variabel stream***, langkah selanjutnya adalah menghubungkan ***stream*** tersebut ke ***file yang akan diakses***
- Untuk melakukan hal ini perlu digunakan fungsi **`open ()`** yang merupakan anggota dari masing-masing stream diatas.
- Dengan kata lain fungsi **`open ()`** dimiliki oleh stream ***ifstream***, ***ofstream***, dan ***fstream***.

MEMBUKA DAN MENUTUP FILE

- Berikut ini prototipe dari fungsi **open()** dalam masing-masing stream tersebut :

```
Void ifstream::open(const char* filename,
          ios::openmode mode = ios::in);
Void ofstream::open(const char* filename,
          ios::openmode mode = ios::out|ios::trunc);
Void fstream::open(const char* filename,
          ios::openmode mode = ios::in|ios::out);
```

MEMBUKA DAN MENUTUP FILE

- Parameter filename diatas adalah *nama file yang akan diakses, termasuk lokasinya (**pathnya**)*
- Sedangkan ***mode*** adalah *menandakan bagaimana file tersebut akan dibuka seperti pada prototipe diatas dilakukan kombinasi nilai-nilai **mode** tersebut dengan melakukan **operasi OR** (**menggunakan operator |terhadapnya**).*

MEMBUKA DAN MENUTUP FILE

- **`ios::app`** , *yang akan menyebabkan output dari file tersebut menjadi ditambahkan (append) pada bagian akhir baris.*
- **`ios::ate`** , *yang akan menyebabkan pencarian ke akhir file ketika file tersebut dibuka*
- **`ios::in`** , *yang akan menyebabkan file tesebut memiliki kapabilitas untuk input*
- **`ios::out`** , adalah *untuk output*

MEMBUKA DAN MENUTUP FILE

- **`ios::binari`** ,*yang akan menyebabkan file yang akan dibuka tersebut dalam mode biner namun secara default file yang dibuka berada dalam mode teks.*
- **`ios::trunc`** ,*yang akan menyebabkan isi file dengan nama yang sama dengan file yang telah dibuka, akan dipotong atau dibuang sehingga lebarnya menjadi nol*.

MEMBUKA DAN MENUTUP FILE

- Berikut ini contoh didalam C++ yang berguna untuk membuka suatu file untuk proses output :

```
// Mendeklarasikan variabel stream  untuk proses output
Ofstream output;
// Memanggil fungsi open()
Output.open("myfile", ios::out);
```

MEMBUKA DAN MENUTUP FILE

- Kita dapat melakukan ***pencegahan*** jika ternyata file yang akan dibuka tidak ada atau terdapat kesalahan lainnya.
- Untuk melakukan hal ini sebaiknya menambahkan suatu penanganan kesalahan pada kode diatas sehingga kodenya benjadi seperti berikut :

```
If (!output) {
 cout<<"file tidak dapat dibuka";
}
```

MEMBUKA DAN MENUTUP FILE

- Setelah file dibuka dan sudah selesai diproses sesuai dengan kebutuhan program, maka perlu menutup file tersebut.
- Cara yang digunakan untuk melakukan hal tersebut adalah dengan menggunakan fungsi **`close()`**.
- Fungsi **`close()`** tidak memiliki parameter, sebagai contoh, apabila kita menghubungkan file dengan ***stream*** yang bernama ***mystream***, maka perlu menutup file tersebut dengan perintah :

```
Mystream.close();
```

MEMBACA DAN MENULIS TEKS DARI/KE DALAM FILE

- Membaca dan menuliskan teks ke file sama mudahnya seperti kita melakukan input dan output data terhadap ***I/O console***, yaitu dengan menggunakan ***operator << dan >>.***
- Perbedaanya tidak menggunakan stream **`cin`** dan **`cout`**, melainkan menggunakan ***stream*** yang kita deklarasikan sendiri dan telah terhubung dengan sebuah file.

MEMBACA DAN MENULIS TEKS DARI/KE DALAM FILE

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {

  // Mendeklarasikan stream untuk proses output
  ofstream output;
  output.open("D:/COBA.TXT");

  // Melakukan pencegahan terhadap terjadinya error
  if (!output) {
     cout<<"File tidak dapat dibuka"<<endl;
     return 1;
  }

  // Menuliskan teks ke dalam file
  output<<"Belajar I/O C++"<<endl;
  output<<"Membaca dan Menulis File"<<endl;
  output<<"Menjadi Lebih Mudah dan Sederhana"<<endl;

  // Menutup file
  output.close();

  return 0;
}
```

FUNGSI PUT() DAN GET()

- Fungsi **put()** dan **get()** dapat juga digunakan untuk proses penulisan dan pembacaan data ke/dari dalam file.
- Fungsi **put()** digunakan untuk menulis data kedalam file, sedangkan **get()** digunakan untuk membaca atau mengambil data dari dalam file.
- Bentuk umum fungsi **put()** dan **get ()** adalah :

```
ostream &put(char ch);
istream &get(char& ch);
```

FUNGSI PUT ( ) DAN GET ( )

```
#include <iostream>
#include <fstream>

using namespace std;

int main() {

  ofstream output;

  output.open("D:/TEST.TXT");

  if (!output) {
    cout<<"File tidak dapat dibuka"<<endl;
    return 1;
  }

  int C=65;
  while  (char(C) <= 'Z') {
    output.put(char(C));
    C++;
  }

  output.close();

  return 0;
}
```

FUNGSI `WRITE()` DAN `READ()`

- Selain cara-cara yang telah disebutkan sebelumnya, terdapat cara lain untuk melakukan proses penulisan dan pembacaan data didalam file yaitu dengan menggunakan **`write()`** dan **`read()`**.
- Dari namanya kita telah mengetahui nbahwa fungsi **`write()`** itu digunakan untuk proses penulisan data dan fungsi **`read()`** digunakan untuk membaca data.
- Kedua fungsi ini dapat digunakan untuk file bertipe teks atau biner.

FUNGSI `WRITE()` DAN `READ()`

- Berikut ini adalah bentuk umum dari fungsi **`write()`** dan **`read()`**

```
ostream &write(const char *buf, streamsize n);
istream &read(const char *buf, streamsize n);
```

- Fungsi **`write()`** akan menulis n buah karakter untuk dimasukan ke stream dari buffer yang ditunjuk oleh pointer buf.
- Fungsi **`read()`** akan membaca n buah karakter dari stream untuk ditempatkan ke buffer yang ditunjuk oleh ***pointer buf***.

FUNGSI WRITE() DAN READ()

```
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

struct SISWA {
  char NIM[9];
  char Nama[25];
  char Kota[15];
  int Usia;
};

int main() {
  // Mendeklarasikan variabel S
  // bertipe SISWA
  SISWA S;

  // Mengisikan nilai ke dalam variabel S
  strcpy(S.NIM, "10299009");
  strcpy(S.Nama, "Bob");
  strcpy(S.Kota, "Jakarta");
  S.Usia = 35;

  // Mendeklarasikan stream
  // untuk proses output
  ofstream OUTPUT;

  OUTPUT.open("D:/DATA",
              ios::out |
              ios::trunc |
              ios::binary);
```

FUNGSI WRITE() DAN READ()

```
if (!OUTPUT) {
     cout<<"File tidak dapat dibuka"<<endl;
     return 1;
  }

  // Menulis data ke stream
  OUTPUT.write((char *) &S, sizeof(S));

  OUTPUT.close();

  // Mendeklarasikan stream
  // untuk proses input
  ifstream INPUT;

  INPUT.open("D:/DATA",
             ios::in | ios::binary);

  if (!INPUT) {
     cout<<"File tidak dapat dibuka"<<endl;
     return 1;
  }

  // Membaca dari stream ke buffer
  INPUT.read((char *) &S, sizeof(S));

  // Menampilkan data
  cout<<S.NIM<<endl;
  cout<<S.Nama<<endl;
  cout<<S.Kota<<endl;
  cout<<S.Usia<<endl;

  INPUT.close();

  return 0;
}
```