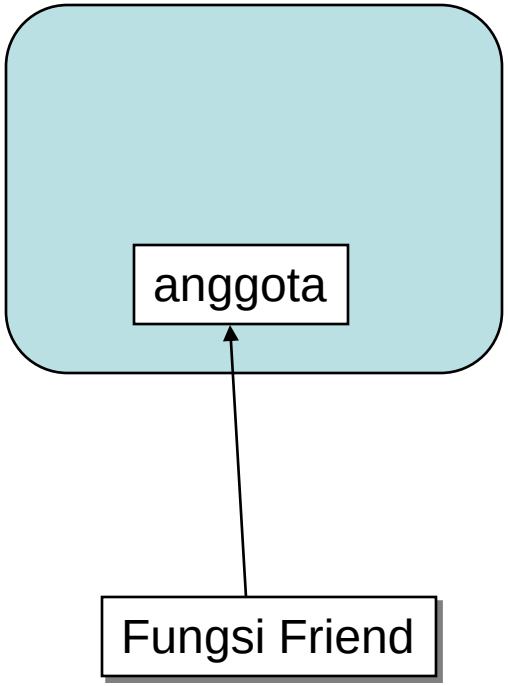
FRIEND

Pengantar

- Friend merupakan kontradiktif
- Friend meruntuhkan konsep enkapsulasi
- Friend memungkinkan pihak luar class untuk mengakses anggota class, termasuk yang bersifat private
- Greg perry: jika class friend terlalu banyak, mungkin diperlukan untuk mengkombinasikan class-class menjadi sebuah class.

Pengantar…2

- Fungsi friend adalah fungsi bukan anggota class yang dapat mengakses anggota class
- Fungsi ini dapat dipakai untuk mengakses anggota class baik proteksi maupun private.
- Pemanggilan fungsi friend melalui call by value



Kriteria penggunaan atribut friend:

- Sedapat mungkin hindari penggunaan friend. Penggunaan friend di antara kelas menunjukkan perancangan kelas yang kurang baik. Jika kelas A menjadikan kelas B sebagai friend maka kemungkinan besar kelas A dan B seharusnya tidak dipisahkan
- Jika operasi yang dijalankan oleh sebuah fungsi friend mengubah status dari objek, operasi tersebut harus diimplementasikan sebagai fungsi anggota
- Gunakan friend untuk overloading pada operator tertentu.

Contoh Program

```
Class mahasiswa{

  private:
      long nim;
     char nama[35];
     char jurusan[20];

  public:
     mahasiswa();
     void inisilisasi(long no, char *nama,
      char *jur);
     friend void tampil(mahasiswa mhs);
};


Void main(){
   mahasiswa mhs;
   mhs.inisialisasi(7363,”joko”,”Pilkom”);
      tampilkan(mhs);
}
```

Fungsi biasa

```
Mahasiswa::mahasiswa(){
     nim =0;
     strcpy(nama,””);
     strcpy(jurusan,””);
}

Void mahasiswa::inisialisasi(long no, char
         *nama, char *jur)
{
    nim=no;
    strcpy(ma                    ma);
    strcpy(ju
}


    Void tampil(mahsiswa mhs){
   cout<<“NIM    : “<< mhs.nim << endl;
   cout<<“Nama : “ << mhs.nama<<endl;
   cout<<“Jurusan : “ << mhs.jurusan<<endl;
}
```

Tidak diawali dg nama class

Fungsi untuk Lebih dari Satu Objek

```
Class hasil_ujian{

  private:
      long nim;
      float nilai;

  public:
     hasil_ujian(long no, float hasil);
     friend float nilai terbesar(
     hasil_ujian a, hasil_ujian b,
     hasil_ujian c);
};


Void main(){
   hasil_ujian amir(898,90.4);
   hasil_ujian endah(899,87.8);
   hasil_ujian joko(900,93.9);
   cout<<“Nilai terbesar = “ <<
   nilai_terbesar(amir, endah, joko)
   <<endl;
}
```

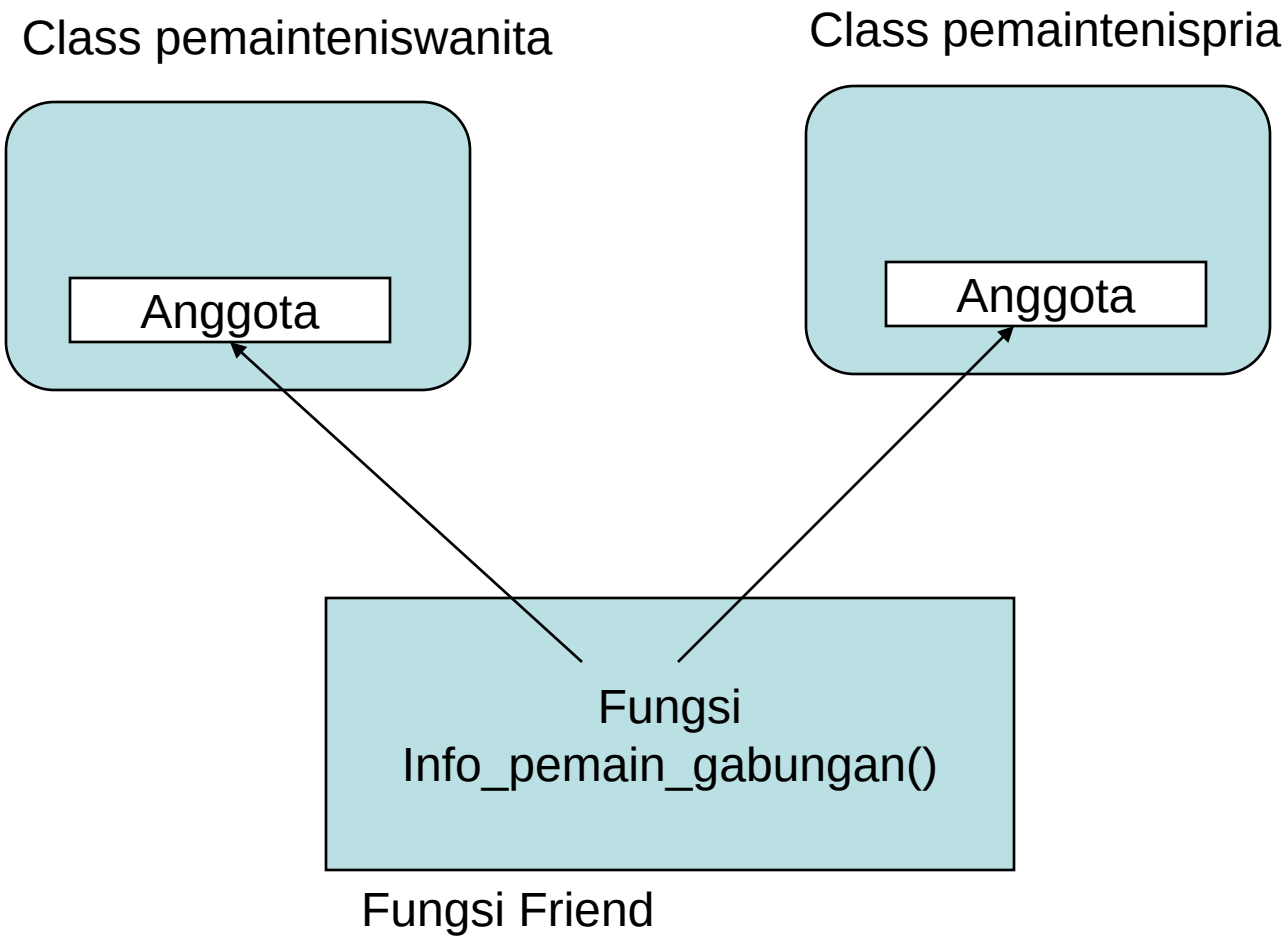
```
hasil_ujian :: hasil_ujian(long no, float hasil){
      nim =no;
      nilai=hasil;
}


Void nilai_terbesar(hasil_ujian a,
          hasil_ujian b, hasil_ujian c){

       float maks=a.nilai;
      maks=(b.nilai>maks) ? b.nilai : maks;
      maks=(c.nilai>maks) ? C.nilai : maks;

      return(maks);
}
```

Friend untuk Lebih dari Satu Class



Contoh Program

```
Class pemaintenispria;
Class pemainteniswanita{
   private:
     char nama[35];
     char asal[20];
  public:
     pemainteniswanita(char *Nama, char
      *Asal);
     friend void
     info_campuran(pemainteniswanita x,
     pemaintenispria y);
}
Class pemaintenispria{
   private:
     char nama[35];
     char asal[20];
  public:
     pemaintenispria(char *Nama, char
     *Asal);
     friend void
     info_campuran(pemainteniswanita x,
     pemaintenis pria y);
}
```

```
Void main(){
Pemainteniswanita wanita("Sales","Yugo");
Pemaintenispria pria("Sampras","AS");
Info_campuran(wanita, pria);

}

pemainteniswanita(char *nama, char
      asal){
     strcpy(nama, Nama);
     strcpy(asal, Asal);
}

pemaintenispria(char *nama, char
      asal){
     strcpy(nama, Nama);
     strcpy(asal, Asal);
}

Void info_campuran(pemainteniswanita x,
     pemaintenispria y){
   cout<<x.nama<<"→"<<x.asal<<endl;
   cout<<y.nama<<"→"<<y.asal<<endl;
}
```

Fungsi Anggota Friend

- Fungsi anggota class dapat berkedudukan sebagai fungsi friend
- Fungsi ini dapat mengakses anggota class yang mendeklarasikanya

Contoh

```
Class pemainteniswanita{
   private:
     char nama[35];
     char asal[20];
  public:
     pemainteniswanita(char *Nama, char *Asal);
     void info_campuran(conts pemaintenispria &y);
}

Class pemaintenispria{
   private:
     char nama[35];
     char asal[20];
     friend void pemainteniswanita::info_campuran(conts pemaintenispria &y);


public:
     pemaintenispria(char *Nama, char *Asal);

}
```

CLASS FRIEND

- Suatu class dapat dijadikan friend class yang lain
- Class ini digunakan kalau ada dua class untuk mengakses bagian private

Contoh

```
Class D3{
  private:
    int ruang_d3;
  public:
    D3(){
          ruang_d3=10;
    }
  friend class S1;
}


Class S1{
  private:
    int ruang_s1;
  public:
    S1(){
       ruang_s1=6;
    }
```

```
  Void info_ruang(D3 x){
     cout<<“Ruang D3 : “ <<x.ruang_d3<<endl;
     cout<<“Ruang S1 : “<<ruang_s1<<endl;
     cout<<“Ruang Gabungan”<<
      x.ruang_d3 + ruang_s1 <<endl;
  }

};


Void main(){
  D3 r3;
  S1 r1;

  r1.info_ruang(r3);
}
```

Friend & Overloading operator

- Overload terhadap operator ==
- Overload terhadap operator <<
- Overload terhadap opeator >>

Overload terhadap operator ==

```
Class tgl{
    private:
      int tanggal;
      int bulan;
      int tahun;
     public:
      tgl(int T, int B, int Th);
      friend int operator ==(tgl T1, tgl T2);
};

Void main(){
     tgl tgl_lahir(22,8,1987);
     tgl hari_pendidikan(2,5,2007);
    if (tgl_lahir == hari_pendidikan)
        cout<<“lahir di hari pendidikan
         nasional”<<endl;
    else
      cout<<“lahir di hari biasa “<<endl;
}
```

```
Int operator ==(tgl T1, tgl T2){
   if (T1.tanggal != T2.tanggal) ||
      ( T1.bulan != T2.bulan) ||
      ( T1.tahun != T2.tahun)
          return(0);
   else
         return(1);
}

Tanggal::tanggal(int T, int B, int Th){
    tanggal = T;
    bulan = B;
    tahun = Th;
}
```

OPERASI FILE

Operasi Dasar

- Membuka atau mengaktifkan file
- Melaksanakan pemrosesan file
- Menutup file

Membuka File

- Deklarasikan :
  - Ofstream nama_objek
- Deklarasikan :
  - Nama_objek.open(nama_file)

Menulis ke file

- Nama_objek <<“string…” << endl

Menutup file

- Nama_objek.close()

Merekam string ke file

```
Void main(){
    ofstream file_keluaran;

    file_keluaran.open(“pelangi.txt”);
    cout<< “sedang merekam”<<endl;
    file_keluaran<<“pelang-pelangi” <<endl;
    file_keluaran<<“alangkah indahmu…”<< endl;

    file_keluaran.close();
}
```

Membaca string dari file

```
Void main(){

    conts int maks=80;
    char isi[maks+1];
    ifstream file_keluaran;

    file_keluaran.open(“pelangi.txt”);

    while (file_keluaran){
       file_keluaran.getline(isi, maks);
       cout<< isi <<endl;
    }

    file_keluaran.close();
}
```

Menambah string ke file

Pada bagian open perlu dirambahkan : ios::app

```
Void main(){
    ofstream file_keluaran;

    file_keluaran.open(“pelangi.txt”, ios::app);

    file_keluaran<<“pelukismu agung” <<endl;
    file_keluaran<<“siapakah gerangan…”<< endl;

    file_keluaran.close();
}
```

Memeriksa keberhasilan operasi file

- Good(): untuk memeriksa keberhasilan dari suatu operasi file. Hasil benar jika operasi berhasil dilakukan
- Eof() : untuk memeriksa akhir file, hasil benar jika akhir file telah dijumpai
- Fail() : untuk memeriksa sesuatu kesalahan. Hasil benar jika terjadi suatu kesalahan. Memeriksa 3 jenis kesalahan:
  - Kegagalan perangkat keras(bad sektor..)
  - Kegagalan baca/tulis
  - Kesalahan karena file tidak ada
- Bad() : untuk memeriksa adanya operasi yang tidak absah.

Contoh:

```
Void main(){

    ifstream file_masukan(“z:82282.933”);

    cout<<“good = “<<file_masukan.good() <<endl;
    cout<<“bad = “<<file_masukan.bad() <<endl;
    cout<<“fail = “<<file_masukan.fail() <<endl;
    cout<<“eof = “<<file_masukan.eof() <<endl;

}
```

Operasi file pada OO

- Merekam :
  - Objek.write((char*) &objek, sizeof(objek));
- Membaca:
  - Objek.read((char*)&objek, sizeof(objek));

Contoh merekam

```
Class buku{
  private:
     char kode[8];
     char judul[35];
     char pengarang[20];
     int jumlah;
  public:
     void entri();
}

Void rekam(buku B);

Void main(){
   buku fiksi;
   rekam(fiksi);
}
```

```
void buku::entri(){
  char tmp[15];
  cout<<“merekam data”<<
  endl;
  cout<<“kode = “;
cin.getline(kode,sizeof(kode));
cout<<“judul = “;
cin.getline(judul,sizeof(judul));
cout<<“pengarang = “;
cin.getline(pengarang,sizeof(
pengarang));
cout<<“jumlah = “;
cin.getline(tmp,sizeof(tmp));
Jumlah = atoi(tmp);

}
```

```
Void rekam(buku B){
  char jawab;
  ofstream file_B(“buku.dat”,
   ios::app);
  for ( ;  ;  ){
    buku.entri();
    file_B.write((char *)&buku,
    sizeof(buku));

    cout << endl;
    cout << “Mau masukan
    data lagi (Y/T)”;
     do{
        jawab=touper(getch());
      }while(!((jawab==‘Y’) ||
        (jawab==‘T’)));

    cout<<jawab<<endl;
    if (jawab ==‘T’)
       break;
  }
  file_B.close();
}
```

Contoh membaca

```
Class buku{
  private:
     char kode[8];
     char judul[35];
     char pengarang[20];
     int jumlah;
  public:
     void info();
}

Void baca(buku B);

Void main(){
   buku fiksi;
   baca(fiksi);
}
```

```
Void buku::info(){
  cout<<“Kode = “ << kode<<endl;
  cout<<“Judul = “ << judul <<endl;
   cout<<“Pengarang = “ << pengarang << endl;
  cout << “Jumlah  = “ << jumlah << endl;
}

Void baca(buku B){
  ifstream file_B(“buku.dat”);

  cout << “Daftar Buku” <<endl;;

  file_B.read((char *)&buku,  sizeof(buku));
  while (!file_B.eof())
     {
        buku.info();
        file_B.read((char *)&buku, sizeof(buku));
      }

   file_B.close();
}
```