## Bahasa Pemrograman

**Kuliah 12 :: Inline Function, Polymorphism**

**Julio Adisantoso**

## Inline Function

- Kode fungsi secara langsung mengganti fungsi panggilnya.
- Inline class member functions
  - implicit, jika definisi fungsi merupakan bagian dalam class.
  - explicit, jika didefinisikan di luar class, dengan menggunakan keyword *inline*.
- Inline functions umumnya pendek, satu baris. Mengapa?



## Contoh

```
class myString {
   char *st;
   int len;
   public:
      char* getString() { return st; }
      // ...
};

inline int myString::getLength()
{
   return len;
}
```

**implicit**

**explicit**



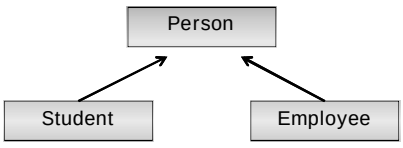
## Konsep OOP

- Encapsulation
  - ADT, Objek
- Inheritance
  - Penurunan/pewarisan sifat objek
- Polymorphism
  - Tiap objek tahu siapa dirinya



## Polymorphism

Perhatikan hirarki berikut:






## Polymorphism

```
class Person {
  protected:
    string nama;
  public:
    Person(string nm="") { nama=nm; }
    string getNama() { return nama; }
};
class Student : public Person {
  public:
    Student(string nm="") { nama=nm; }
};
class Employee : public Person {
  public:
    Employee(string nm="") { nama=nm; }
};
```

```
int main() {
   Person p;
   Student s("Unyil");
   Employee e("Ogah");
   p=s;
   cout << p.getNama() << endl;
   p=e;
   cout << p.getNama() << endl;
   return 0;
}
```

Kemampuan dari referensi untuk mengubah sifat menurut objek apa yang dijadikan acuan dinamakan **polimorphism**

## Isu dalam polymorphism

- Static binding
- Dynamic binding

Slide 6

## Static Binding

```
class Time {
   protected:
      int hour, minute, second;
   public:
      Time(int h=0, int m=0, int s=0) {
         hour=h; minute=m; second=s; }
      void write() {
         cout << ((hour<10)?"0":"") << hour << ":"
              << ((minute<10)?"0":"") << minute << ":"
              << ((second<10)?"0":"") << second
              << endl; }
};
```

Slide 7

## Static Binding

```
class ExtTime : public Time {
   string ap;
   public:
      ExtTime(int h=0,int m=0,int s=0,string z="pm") {
         hour=h; minute=m; second=s; ap=z; }
      void write() {
         cout << ((hour<10)?"0":"") << hour << ":"
              << ((minute<10)?"0":"") << minute << ":"
              << ((second<10)?"0":"") << second
              << ap << endl; }
};
void print(Time t) {
   cout << "Pukul : ";
   t.write(); }
```

Slide 8

## Static Binding

```
int main() {
   Time t1(8,30);
   ExtTime t2(10,45);
   print(t1);
   print(t2);
   return 0;
}
```

```
Pukul : 08:30:00
Pukul : 10:45:00
```

Static binding → menentukan fungsi mana yang dipanggil untuk objek tertentu berdasarkan tipe dari parameter formal.

Ketika menggunakan pass-by-value, akan terjadi static binding.

Slide 9

## Dynamic Binding

- Fungsi mana yang akan dipanggil dari class turunan berdasarkan tipe turunannya.
- Menggunakan kata kunci virtual.
- Membutuhkan pass-by-reference

Slide 10

## Dynamic Binding

```
class Time {
   protected:
      int hour, minute, second;
   public:
      Time(int h=0, int m=0, int s=0) {
         hour=h; minute=m; second=s; }
      virtual void write() {
         cout << ((hour<10)?"0":"") << hour << ":"
              << ((minute<10)?"0":"") << minute << ":"
              << ((second<10)?"0":"") << second
              << endl; }
};
```

Slide 11

## Dynamic Binding

```
class ExtTime : public Time {
   string ap;
   public:
      ExtTime(int h=0,int m=0,int s=0,string z="pm") {
         hour=h; minute=m; second=s; ap=z; }
      virtual void write() {
         cout << ((hour<10)?"0":"") << hour << ":"
              << ((minute<10)?"0":"") << minute << ":"
              << ((second<10)?"0":"") << second
              << ap << endl; }
};
void print(Time *t) {
   cout << "Pukul : ";
   t->write(); }
```

Slide 12

## Dynamic Binding

```
int main() {
   Time t1(8,30);
   ExtTime t2(10,45);

   Time *tp;
   tp=&t1;
   print(tp);
   tp=&t2;
   print(tp);
   return 0;
}
```

```
Pukul : 08:30:00
Pukul : 10:45:00pm
```

Slide 13