OVERLOADING OPERATOR

Oleh:
Rasim
Ilkom - UPI

Pengantar Overloading Operator

- Konsepnya diilhami oleh operasi yang sering dialami dalam kehidupan sehari-hari, misalnya tanda +, - , x dan lain-lain

Contoh:

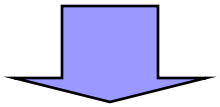
3 + 5

4,9 + 9,2

Dalam C++, dapat tangani menurut bawaannya

Pengantar Overloading Operator..2

matrik_A + matrik_B ?



Tidak dapat dilakukan oleh C++, sehingga perlu overloading terhadap operator tersebut.

Jadi sintak overloading operator:

Tipe operator simbol_operator(kelas objek)

Tipe operator simbol_operator(tipe var)

Overloading thd Operator Biner

```
Class Buah {
        int apel;
        int jeruk;
    public:
      Buah(int jum_apel=0, int jml_jeruk=0);
      void info_buah();
      Buah operator + (buah b2);
};
```

Overloading thd Operator Biner..2

```
void main(){
        Buah buah1(20, 5);
        Buah buah2(12, 4);
        Buah buah3;

        buah3 = buah1+ buah2;

       cout<<“objek buah 1” << endl;
       buah1.info_buah();

      cout<<“objek buah 2” << endl;
       buah2.info_buah();

      cout<<“objek buah 3” << endl;
       buah3.info_buah();
}
```

```
Buah::Buah(int jum_apel, int jum_jeruk){
     apel =jum_apel;
     jeruk=jum_jeruk;
}

void Buah::info(){
     cout<<“Jumlah apel = “ << apel
     << “ Jeruk = “ << jeruk << ‘\n’
     << endl;
}

Buah Buah::operator + (Buah b2){
   Buah tmp;
   tmp.apel = apel+b2.apel;
   tmp.jeruk = jeruk + b2.jeruk;
  return(tmp);
}
```

Overloading thd Operator Unary

- Nama_kelas operator ++(void) //awalan
- Nama_kelas operator ++(int) //akhiran
- Nama_kelas operator --(void) //awalan
- Nama_kelas operator --(int) //akhiran

Contoh

```
Class Buah {
        int apel;
        int jeruk;
    public:
      Buah(int jum_apel=0, int jml_jeruk=0);
      void info_buah();
      Buah operator ++ (int);
      Buah operator --(void)
};
```

```
void main(){
        Buah buah1(20, 5);

       cout<<“objek buah 1” << endl;
       buah1.info_buah();

      Buah buah2=buah1++;
      cout<<“objek buah 2” << endl;
       buah2.info_buah();

      Buah buah3=++buah1;
      cout<<“objek buah 3” << endl;
       buah3.info_buah();
}
```

```
Buah::Buah(int jum_apel, int jum_jeruk){
     apel =jum_apel;
     jeruk=jum_jeruk;
}

void Buah::info(){
     cout<<“Jumlah apel = “ << apel
     << “ Jeruk = “ << jeruk << ‘\n’
     << endl;
}

Buah Buah::operator ++ (int){
   return(Buah(apel++, jeruk++));
}

Buah Buah::operator -- (void){
   return(Buah(--apel, --jeruk));
}
```

Overloading Operator Majemuk

Operator majemuk adalah:

- +=
- /=
- -=
- *=

Contoh

```
Class Buah {
        int apel;
        int jeruk;
    public:
      Buah(int jum_apel=0, int jml_jeruk=0);
      void info_buah();
      Buah operator += (Buah);
};
```

```
void main(){
        Buah buah1(20, 5);
        BUah buah2(8, 23);

       cout<<“objek buah 1”<< endl;
       buah1.info_buah();
      cout<<“objek buah 2” << endl;
       buah2.info_buah();

       buah2 +=buah1;

      cout<<“objek buah 2” << endl;
      cout<<“setelah di += “<<endl;
       buah2.info_buah();
}
```

```
Buah::Buah(int jum_apel, int
jum_jeruk){
     apel =jum_apel;
     jeruk=jum_jeruk;
}

void Buah::info(){
     cout<<“Jumlah apel = “ << apel
     << “ Jeruk = “ << jeruk << ‘\n’
     << endl;
}

Buah Buah::operator +=(Buah b){
    apel += b.apel;
    jeruk +=b.jeruk;
    return(*this);
}
```

Pemakain Operator =

- Objek = nilai_dengan_tipe_data_dasar
- Nilai_dengan tipe_data_dasar = Objek
- Objek1 = Objek2 dengan kelas berlainan
  - Rutin pengkonversi dilakukan pada kelas pemberi
  - Rutin pengkonversi dilakukan pada kelas penerima

Objek = nilai_dengan_tipe_data_dasar

```
class string{
        char st[80];
    public:
        string() {
            strcpy(st,"");
        }
        string (char *s){
           strcpy(st, s);
        }
        void info(){
              cout<<"st = "<< st << endl;
        }
};
```

```
void main(){
    string salam;
    salam.info();

    salam="hai";
    salam.info();
}
```

Nilai_dengan tipe_data_dasar = Objek

```
class jarak{
      int km;
      int meter;
   public:
      jarak(int n_km, m_meter);
      operator double(){
            double mil=(1000 * double(km)+meter)/1600;
            return(mil);
      }
};
```

```
void main(){
     jarak x_y(115,7);
     double mil=x_y;
     cout<<“jarak dalam mil = “ << mil << endl;
}

Jarak::jarak(int n_km, int meter){
   km=n_km;
   if (n_meter>1000 {
            km=km+(n_meter/1000);
            meter=(n_meter % 1000);
  }
}
```

Objek1 = Objek2 dengan kelas berlainan

Rutin pengkonversi dilakukan pada kelas penerima

```
class kartesian{
        double x;
        double y;
    public:
        kartesian(double x=0, double y=0);
        void info_kartesian();
        double info_x();
        double info_y();
};
```

```
class polar{
        double panjang;
        double sudut;
    public:
        polar(double panjang=0, double sudut=0);
        polar (kartesian k);
        void info_koordinat();
};
```

```
void main(){
    kartesian posisi1(4,3);
    posisi1.info_koordinat();

    polar posisi2=posisi1;
    posisi2.info_koordinat();
}

kartesian::kartesian(double x, double y){
    kartesian::x=x;
    kartesian::y=y;
}
void kartesian::info_koordinat(){
    cout<< “koordinat kartesian : “ <<endl;
    cout<<“x = “ << x << endl;
    cout<<“y = “ << y <<endl;
}

Double kartesian::info_x(){
    return(x);
}
Double kartesian::info_y(){
    return(y);
}
```

```
polar ::polar(double panjang, double sudut){
      polar::panjeng=panjeng;
      polar::sudut= sudut;
}

Polar::polar(kartesian k){
   double x=k.info_x();
   double y=k.info_y();
   panjang = sqrt(x*x+y*y);
   sudut=atan(y/x);
}
void polar::info_koordinat(){
    cout<<“Koordinat polar :”<<endl;
     cout<<“panjang = “ << panjang<<endl;
     cout<<“sudut = “ << sudut << “radian” <<endl;
}
```

Objek1 = Objek2 dengan kelas berlainan

Rutin pengkonversi dilakukan pada kelas pemberi

```
Class polar;
class kartesian{
        double x;
        double y;
    public:
        kartesian(double x=0, double y=0);
        void info_kartesian();
        operator polar();
};
```

```
class polar{
        double panjang;
        double sudut;
    public:
        polar(double panjang=0, double sudut=0);
        void info_koordinat();
};
```

```
 void main(){
      kartesian posisi1(4,3);
      posisi1.info_koordinat();

      polar posisi2=posisi1;
      posisi2.info_koordinat();
 }

kartesian::kartesian(double x, double y){
     kartesian::x=x;
     kartesian::y=y;
}
void kartesian::info_koordinat(){
     cout<< “koordinat kartesian : “ <<endl;
     cout<<“x = “ << x << endl;
     cout<<“y = “ << y <<endl;
}
```

```
kartesian::operator polar(){
   double panjang = sqrt(x*x+y*y);
   double sudut=atan(y/x);
   return (polar(panjang, sudut));
}

polar ::polar(double panjang, double sudut){
      polar::panjeng=panjeng;
      polar::sudut= sudut;
}

void polar::info_koordinat(){
    cout<<“Koordinat polar :”<<endl;
     cout<<“panjang = “ << panjang<<endl;
     cout<<“sudut = “ << sudut << “radian” <<endl;
}
```