



IPB University
— Bogor Indonesia —

KOM120C -- BAHASA PEMROGRAMAN

Template Programming #2

- STL (Lanjutan)
- Operator Overloading

Tim Pengajar Bahasa Pemrograman IPB University

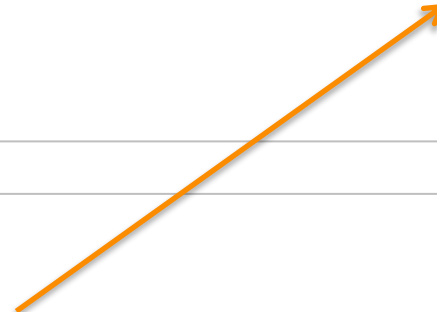
STL :: map

Himpunan elemen yang terdiri atas **key** dan **value**. Nilai disimpan dalam urutan tertentu dari key. Merupakan **associative array**.

Sintaks:

```
#include <map>
map<keytype, valuetype> mapname;
```

```
st["G6421002"]="Ji Pyeong";
st["G6421005"]="Siapa Saja";
```



Contoh:

```
map<string, string> st;
st.insert(pair<string, string>("G6421005", "Siapa Saja"));
st.insert(pair<string, string>("G6421002", "Ji Pyeong"));
st.insert(pair<string, string>("G6421002", "Gundala Putra Petir"));

map<string, string>::iterator p;
for(p=st.begin(); p!=st.end(); ++p)
    cout << p->first << " " << p->second << endl;

for(const auto& ptr : st)
    cout << ptr.first << " --> " << ptr.second << endl;
```

STL :: map

Bagaimana jika value berupa class.

Contoh Problem:

Diinginkan untuk menyimpan data pegawai yang menggunakan class Person.

Setiap pegawai memiliki ID yang unik. Contoh data:

"3204", "Siapa Saja", 17, 170, 72.5

"3201", "Ji Pyeong", 19, 180, 70.2

```
map<string, Person> st;  
Person p;  
p.setPerson("Siapa Saja", 17, 170, 72.5);  
st.insert(pair<string, Person>("3204", p));  
  
p.setPerson("Ji Pyeong", 19, 180, 70.2);  
st.insert(pair<string, Person>("3201", p));  
  
map<string, Person>::iterator ptr;  
for(ptr=st.begin(); ptr!=st.end(); ++ptr)  
    cout << ptr->first << " --> " << (ptr->second).getNama() << endl;
```

```
3201 --> Ji Pyeong  
3204 --> Siapa Saja
```

STL :: map

Bagaimana menyusun map secara terbalik atau mengikuti logika tertentu?
STL Map tidak memiliki fitur descending order.

Prosedur:

- Pindahkan struktur MAP ke struktur VECTOR yang memiliki algoritme SORT.
- Gunakan algoritme SORT dengan menggunakan logika pembandingan tertentu (compare).

```
// descending logics
bool cmp(pair<string, Person>& a, pair<string, Person>& b)
{
    return a.first > b.first;
}

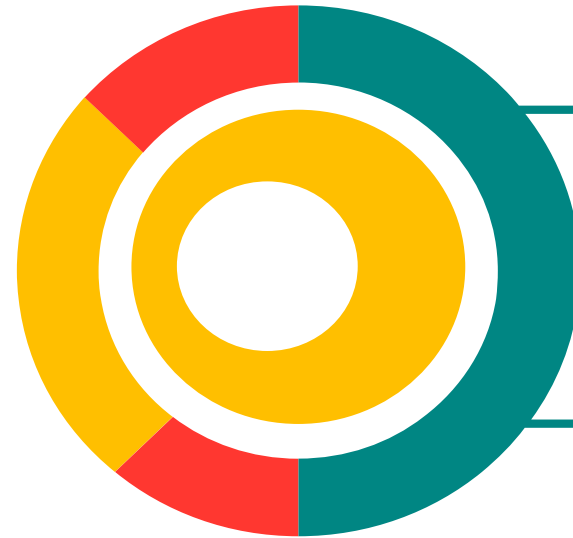
void sort(map<string, Person>& M)
{
    vector<pair<string, Person> > A;

    for (auto& it : M) {                // copy map to vector
        A.push_back(it);
    }
    sort(A.begin(), A.end(), cmp);

    for (auto& it : A) {                // print all elements
        cout << it.first << " --> "
              << (it.second).getNama() << endl;
    }
}
```

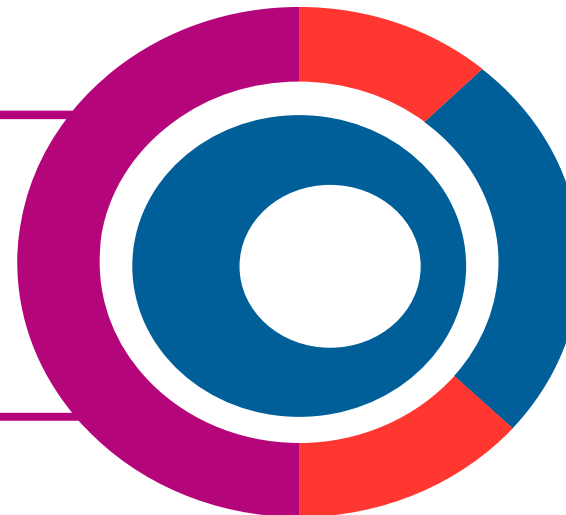
Overloading

C++ memungkinkan untuk membuat satu atau lebih definisi untuk nama fungsi atau nama operator dalam lingkup yang sama.



Function Overloading

Fitur dalam OOP yang mengizinkan 1 atau lebih definisi berbeda dari suatu fungsi dengan nama yang sama.



Operator Overloading

Fitur dalam OOP yang mengizinkan 1 atau lebih definisi berbeda dari suatu operator dengan nama yang sama.

Bagaimana function overloading bekerja?



Exact match

Nama fungsi dan argumen sesuai → EXECUTE !



Promoted to Appropriate Type

char, unsigned char, short → int
float → double



Standard Conversion

Konversi baku C++ dari tipe tertentu ke tipe lain yang bersesuaian.

Contoh: Integer conversion: unsigned char, unsigned short int, unsigned int, unsigned long int, unsigned long long int.

else → ERROR!

Operator Overloading

C++ memungkinkan kita membuat fungsi operator sesuai dengan definisi yang dikehendaki.

Contoh: operator penjumlahan (+) dapat digunakan untuk menjumlahkan berbagai data (int, float, double, char), bahkan untuk menjumlahkan objek dari class (String), atau objek dari class yang dibuat oleh user.

Caranya?

Suatu operator X dapat dituliskan sebagai fungsi dengan prototipe sebagai berikut:

```
<return type> operatorX (<argument list>)
```

Contoh Kasus

Terdapat class Complex untuk mengimplementasikan objek bilangan complex yang terdiri atas 2 bagian data, yaitu bilangan real dan imajiner. Objek dapat diolah menggunakan operator aritmatika +.

```
class Complex
{
    private:
        int real, imager;
    public:
        Complex(int r=0, int i=0) { real=r; imager=i; }
        Complex operator+ (Complex const &c) {
            Complex res;
            res.real = real + c.real;
            res.imager = imager + c.imager;
            return res;
        }
        void print() {
            cout << real << " + i" << imager << endl;
        }
};
```

```
int main()
{
    Complex c1(10,5), c2(2,4);
    Complex c3, c4;
    c3=c1+c2;
    c4=c1.operator+(c2);
    c1.print();
    c2.print();
    c3.print();
    c4.print();

    return 0;
};
```


Prefix and Postfix Operator Overloading

C++ memiliki 2 jenis increment/decrement operator:

- Prefix increment and postfix increment
- Prefix decrement and postfix decrement

Implementasi untuk Class Complex (contoh):

```
Complex& operator++();           // Prefix increment
Complex operator++(int);         // Postfix increment

Complex& operator--();          // Prefix decrement
Complex operator--(int);         // Postfix decrement
```

Latihan

Gunakan konsep OOP

Deskripsi

Buat program mengelola bilangan pecahan a b/c.

Format Masukan

Beberapa baris operasi bilangan pecahan:

set a b c	inisialisasi bilangan pecahan a b/c
p	menampilkan bilangan pecahan sesederhana mungkin
add a b c	menambah bilangan pecahan yang ada dengan a b/c
mul a b c	mengalikan bilangan pecahan yang ada dengan a b/c
inc	postfix increment
dec	postfix decrement
end	akhir dari pengolahan

Format Keluaran

Beberapa baris bilangan pecahan sesederhana mungkin.

Contoh Input

```
set 4 2 8
p
add 0 6 8
inc
p
```

Contoh Output

```
4 1/4
5
```