Standard template library (STL)

pengenalan

C++

Vincent Sebastian The

C++ Libraries

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <stack>
#include <queue>

#include <deque>
#include <string>
#include <algorithm>
#include <set>
#include <map>
#include <list>
#include <cctype>
#include <bitset>
```

#include <cstdio>

Misalkan ada input yang memiliki format: ‘Tes#5 : ‘3.

```
scanf(“Tes #%d: %d”, &i, &j);


printf(“Hasil tes %d: %.3lf\n”, i, p);
//print 3 angka dibelakang koma.
```

Tipe data dalam scanf/printf

Tipe Data	Lambang
int	%d
unsigned int	%u
long long	%lld
unsigned long long	%llu
char	%c
cstring	%s
float	%f
float desimal dibulatkan	%g
double	%lf
double desimal dibulatkan	%lg

#include <iostream>

```
cin>>”tes #”>>i>>”:”>>j;
cout<<“tes “<<i<<“:”<<p;
```

Operasi pada iostream sangat lambat, sehingga sebisa mungkin hindari penggunaannya untuk input yang sangat banyak.

#include <cstring>

```
//deklarasi cstring (seperti deklarasi array of char)
char s[1000];

//memasukan nilai s2 ke s1
strcpy(s1, s2);

//membandingkan 2 string, mengembalikan nilai 0 bila
kedua string sama
i = strcmp(s1, s2);

//mengembalikan panjang string (kompleksitas O(n) )
i = strlen(s1);
```

#include <cstdlib>

```
//mengembalikan nilai absolute.
j = abs(i);

//menterminasi program
exit(0);

//menghasilkan nilai random
i = rand();
```

#include <cmath>

```
//cos, sin, tan
i = cos(60 * PI / 180);  //menghasilkan cos 60o

//acos, asin, atan
i = acos(1);

//mengasilkan nilai PI
PI = 2*acos(0);

//akar dari suatu bilangan
i = sqrt(j);
```

```
//menghasilkan i^j
k = pow(i, j);

//menghasilkan jarak phytagoras
d = hypot(dx, dy);

//absolute dari float atau double
f = fabs(p);

//floor & ceil
i = floor(p);
i = ceil(p);
```

#include <vector>

Vector dapat dianalogikan sebagai tipe data array yang ukurannya bisa berubah-ubah.

```
//deklarasi
vector<int> vi;

//memasukkan data ke belakang vector
vi.push_back(i);

//mendapatkan ukuran vector sekarang (kompleksitas O(n))
i = vi.size();

//mengosongkan vector
vi.clear();
```

```
//vector dapa mengambil nilai array maupun vector lain

int m[500];
vector<int> vi, vt;

vi = m;
vt = vi;

//namun array tidak dapat mengambil nilai vector
```

#include <stack>

```
//deklarasi stack
stack<int> st;

//memasukkan variabel ke (atas) stack
st.push(1);

//mengeluarkan bilangan dari (atas) stack
st.pop();

//mengetahui apakah stack kosong
if(st.empty())
```

```
//Mengetahui nilai di atas stack
i = st.top();

//Mengetahui ukuran stack
si = st.size();
```

#include <queue>

```
//deklarasi
queue<int> q;

//memasukkan variabel ke (belakang) queue
q.push(i);

//mengeluarkan variabel dari (depan) queue
q.pop();

//mengambil nilai di depan queue
i = q.front();
```

```
//Mengetahui nilai di depan queue
i = q.front();

//Mengetahui ukuran queue
si = q.size();
```

#include <deque>

Double ended queue (deque) adalah tipe data mirip queue dimana operasi insersi dan delesi dapat dilakukan di kedua ujungnya. Operasi yang mungkin:

```
deque<int> dq;

dq.push_back(i);  //operator []
dq.push_front(i); dq[3] = i;
dq.pop_back();
dq.pop_front();
i = dq.front();
i = dq.back();
if(dq.empty())
i = dq.size();
```

#include <string>

Dalam implementasi penyimpanannya, string sebenarnya adalah vector of char. Yang berarti karakteristik vector seperti mengambil nilai array of char (cstring) juga berlaku di string. Karena string berasal dari C++, string harus dibaca dengan iostream, tidak bisa dengan scanf/printf.

```
//deklarasi
string s1,s2;

//baca dan tulis
cin>>s1;
cout<<s2;
```

```
//operasi dasar pada string
string s1,s2;
char c;

s1 = s2;
s1 = s1+s2;
s1 = “abc” + s2;
s1 = s2 + c;

//mengetahui panjang dari sebuah string
len = s1.length();

//mengosongkan string
s1.clear(); //atau
s1 = “”;
```

Cstring dan String Lanjutan

```
//variabel
char str1[50],str2[50];
string s1,s2;

//Membaca string dalam satu line
gets(str1);    //cstring
getline(cin, s1); //string

//operasi membaca merupakan operasi yang sangat lambat sehingga
//lebih baik membaca lewat cstring, baru memasukannya ke string
gets(str1);
s1 = str1;

//memasukkan nilai string ke cstring
strcpy(str1, s1.c_str());
```

```
//membaca string seperti pada stdin
char str1[50];

strcpy(str1, “Tes #4: 5”);
sscanf(str1, “Tes #%d: %d”, &i, &j);
//nilai i=4 dan j=5;

//menulus seperti pada stdout
sprintf(str1, “Jawab: %d”, i);
```

#include <algorithm>

```
int m[50];
vector<int> vi;

//semua fungsi yang memiliki parameter array atau
//vector, parameternya ditulis dalam bentuk iterator
reverse(m, m+n);  //membalik indeks ke-0 sampai ke-(n-1)
reverse(vi.begin(), vi.end());   //membalik seluruh isi
//vektor

//binary search, array harus sudah tersort terlebih dahulu
if(binary_search(m, m+n, 6))  //true bila ada angka 6 di m
//kompleksitas O(log(n))
```

```
//untuk memprint semua permutation
//asumsikan array m sudah ter-sort telebih dahulu

do {
   for(int i=0 ; i<n ; i++) {
      printf("%d ", m[i]);
   }
 } while(next_permutation(m, m+n));

 //mencari nilai maksimal dan minimal dari array
 i = *min_element(m, m+n);
 i = *max_element(m, m+n);

 //fungsi-fungsi lainnya
 i = min(j, k);
 i = max(j, k);
 swap(i, j);
```

SORT

```
//syntax dasar
sort(m, m+n);

//sort menggunakan fungsi sendiri, misalkan kita ingin
//mensort array, dengan semua bilangan ganjil berada di
//depan terlebih dahulu

bool cmp(int n1, int n2) {
 if(n1%2 == n2%2) { //kalau paritasnya sama
 return (n1>n2);
 } else {
 return (n1%2 == 0);
    }
 }

 //pemanggilannya
 sort(m, m+n, cmp);
```

fungsi komparasi yang dibuat (yang mengembalikan nilai Boolean), mempunyai dua parameter yang bertipe sama dengan tipe data yang akan dibandingkan, fungsi tersebut mereturn true jika parameter pertama (n1 pada contoh sebelumnya) diinginkan bernilai lebih besar (berada di kanan array) daripada n2 dan sebaliknya.

tidak hanya sort, fungsi-fungsi lain juga bisa ditambahkan parameter fungsi komparasi seperti,

```
if(next_permutation(m, m+n, cmp))
```

```
i = *min_element(m, m+n, cmp);
```

#include <set>

```
//set adalah tipe data yang menggunakan sruktur data
// red-black tree
//deklarasi
set<int> mySet;

//memasukkan data ke dalam set
mySet.insert(5);

//menghapus data dari set
mySet.erase(5);
```

```
//mengetahui ukuran set
si = mySet.size();

//mengosongkan set
mySet.clear();

//mengecek apakah sebuah bilangan berada dalam set
if(mySet.count(5))

//mencetak semua bilangan dalam set
for(set<int>::iterator it=mySet.begin(); it!=mySet.end() ; it++)
   printf(“%d”, (*it));
```

#include <map>

```
//Map merupakan tipe data yang juga memakai struktur
//red-black tree
//perbedaannya dengan set ialah setiap index pada map
//memiliki nilai
//Deklarasi
map<string, int> myMap;

//memasukkan data ke dalam map
myMap[“nomor”] = 1;

//mengganti nilai index
myMap[“nomor”] = 1;
```

```
//menghapus elemen dari map
myMap.erase(“nomor”);

//mengecek apakah sebuah indeks pernah di-assign
if(myMap.count(“nomor”))

//mengambil ukuran map
si = myMap.size();

//mengosongkan map
myMap.clear();

//Mencetak isi map
for(map<string,int>::iterator it=myMap.begin() ; it!=myMap.end() ; it++)
 printf(“%d %d”, it->first, it->second);
```

#include <list>

```
//list adalah tipe data yang memungkinkan untuk insersi
//nilai di tengah data secara cepat
//deklarasi
list<int> l;

//memasukkan nilai ke belakang list
l.push_back(5);

//mengakses, memasukkan, dan menghapus nilai di tengah
//list dilakukan dengan menggunakan iterator
list<int>::iterator it;
```

```
//mengakses elemen ke lima
it = l.begin();
for(i=0 ; i<4 ; i++) it++;
cout<<*it;

//memasukkan nilai sebelum ke elemen lima
l.insert(it, 4);

//menghapus elemen ke lima
l.erase(it);
```

Priority Queue (Heap)

```
//Priority Queue adalah STL yang mengimplementasikan
//penggunaan heap, priority queue termasuk dalam
//include <queue>

//deklarasi
priority_queue<int> pq;

//memasukkan nilai ke pq
pq.push(5);

//mengakses nilai di atas heap
i = pq.top();
```

```
//mengeluarkan di atas heap
pq.pop();

//mengecek apakah pq kosong
if(pq.empty())

//secara default priority queue mensort data dengan
//bilangan paling di atas adalah bilangan paling besar
//untuk membuat priority queue memiliki bilangan
//paling kecil di atas gunakan deklarasi berikut

priority_queue<int, vector<int>, greater<int> > pq;
```

#include <cctype>

```
//cctype adalah STL yang memudahkan dalam pengecekan
//tipe sebuah character misalkan untuk mengecek
//apakah sebuah karakter adalah angka
if(isdigit(ch))

//library yang ada
isalnum() isxdigit()
isalpha() isprint()
isdigit() isspace()
islower()
isupper()
```

#include <bitset>

```
//bitset adalah struktur data berisi kumpulan bit-bit
//(1 atau 0). Keunggulan bitset adalah semua operasi
//Boolean dapat digunakan
//deklarasi
bitset<40> b1,b2,b3;

//operasi operasi
b3 = b1&b2;  b3 = b1<<2;
b3 = b1|b2;  b3 = b1>>1;
b3 = b1^b2;  b3 = ~b1;
```