I-RICH CP 2024

C++ STL

By Yesaya Rudolf

PENGENALAN

STL merupakan singkatan dari Standard Template Library. STL memuat banyak struktur data dan algoritma yang berguna di CP, seperti untuk mempersingkat waktu. Kita tidak perlu mengetahui detail implementasi dari template yang sudah dibuat.

STL pada C++ dideklarasikan pada namespace std, dan terdiri dari beberapa header. Untuk melakukan include terhadap banyak header sekaligus, gunakan #include <bits/stdc++.h>. Untuk menghindari penggunaan prefix std, gunakan using namespace std.

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;

int main(){
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int n;
    string s;
    cin >> n >> s;
    cout << "Halo, " << s << " dengan nomor " << n;
    return 0;
}
```

PROBLEM 1 - BAD APPLE!

Reimu dan Marisa merupakan sekelompok petualang yang sedang dalam perjalanan untuk mengalahkan Sakuya, sang manipulator waktu. Dalam perjalanannya, mereka menemukan sebuah keranjang yang berisikan n buah apel. Ternyata m buah apel tersebut sudah busuk. Mereka ingin melakukan pembagian buah yang baik tersebut sehingga masing-masing orang mendapatkan apel dalam jumlah genap. Jika Reimu dan Marisa masing-masing mendapatkan jumlah apel genap, keluarkan "Ya", sebaliknya keluarkan "Tidak".

TIPE DATA DASAR

Bilangan Bulat

Nama	Jangkauan	Format Specifier
short	-2^{15} to $2^{15}-1$	%hd
unsigned short	0 to $2^{16}-1$	%hu
int	-2^{31} to $2^{31}-1$	%d
unsigned int	0 to $2^{32}-1$	%u
long long	-2^{63} to $2^{63}-1$	%lld
unsigned long long	0 to $2^{64}-1$	%llu

Bilangan Real

Nama	Jangkauan	Format Specifier
float	3.4×10^{-38} to $3.4 \times 10^{+38}$	%f
double	1.7×10^{-308} to $1.7\times10^{+308}$	%lf

PROBLEM 2 - PLANETARY SEARCH

Asta merupakan seorang kepala peneliti pada Herta Space Station. Pada suatu saat, Asta memerlukan pencarian planet terdekat supaya tidak mengganggu penelitian yang sedang dilakukan. Asta memiliki data sebanyak n buah planet, yang masing-masingnya terdiri atas nama dan koordinat (x,y). Tentukan nama planet yang memiliki jarak terpendek dari Herta Space Station.

AUTO

Untuk menghilangkan nama tipe yang panjang, Anda dapat menggunakan auto untuk mendeklarasikan variabel. Nilai awal harus diberikan saat mendeklarasikan variabel, dan nilai tersebut digunakan untuk menentukan tipe variabel.

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;

int main(){
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    auto a = "Pekora"; // const char*
    auto b = "Pekora"s; // string
    auto c = 31415; // int
    auto d = 3.14; // double

    vector <ll> v = {2, 5, 7};
    auto e = v.begin(); // vector<ll>::iterator
    return 0;
}
```

PAIR

Struktur data yang digunakan untuk menyimpan dua buah objek dengan tipe data yang sama ataupun berbeda.

```
pair <int, int> p;
p.first = 3;
p.second = 5;
cout << p.first << ' ' << p.second << endl;

pair q(4, "Asta");
cout << q.first << ' ' << q.second << endl;
cout << get<0>(q) << ' ' << get<1>(q) << endl;
```

TUPLE

Generalisasi dari pair, yang mana tuple dapat menampung 2 atau lebih objek. Kedua buah pair/tuple dapat dibandingkan secara leksikografis.

```cpp
tuple <int, int, double> t(5, 9, 2.1);
cout << get<0>(t) << ' ' << get<1>(t) << ' ' << get<2>(t) << endl;

tuple <int, int, double> r(5, 9, 2.2);
cout << (t < r) << endl; // true
```

STRUCTURED BINDING

Mengikat nama-nama yang ditentukan ke anggota dari suatu tipe. Mendukung 3 jenis: array, tipe mirip-tuple, struktur yang didefinisikan sendiri. Gunakan address (simbol &) untuk mendapatkan refrence.

```cpp
pair <int, int> p(9, 10);
auto [x, y] = p;
auto& [a, b] = p;
cout << x << ' ' << y << endl;

a = 3;
y = 7;
cout << p.first << ' ' << p.second << endl;

tuple <int, int, double> t(5, 9, 2.1);
auto& [i, j, k] = t;
cout << i << ' ' << j << ' ' << k << endl;
```

VECTOR

Array dinamis adalah struktur data yang mirip dengan array statis, tetapi dengan kemampuan untuk mengubahukurannya secara otomatis saat dibutuhkan. Hal ini memungkinkan untuk menyimpan data tanpa harus mengetahui terlebih dahulu berapa banyak elemen yang akan diperlukan.

std::vector adalah implementasi array dinamis dalam C++ Standard Template Library (STL).

VECTOR

```cpp
// Constructor
vector <ll> v1; /* Membuat vector kosong */
vector <ll> v2(10); /* Membuat vector dengan size 10 berelemen 0 */
vector <ll> v3(10, 5); /* Membuat vector dengan size 10 berelemen 5 */
vector <vector<ll>> v4; /* Vector 2 dimensi */
```

```cpp
// Penjelajahan dan Akses
v2[8] = 1;
cout << "Ukuran v1: " << v1.size() << "\n";

for (ll i = 0; i < v2.size(); i++) {
    cout << v2[i] << ' ';
} cout << '\n';

for (auto &elm: v3) {
    cout << elm << ' ';
} cout << '\n';
```

```
Ukuran v1: 0

Isi dari v2(10) dengan v2[8] = 1: 0 0 0 0 0 0 0 0 1 0

Isi dari v3(10, 5): 5 5 5 5 5 5 5 5 5 5
```

VECTOR

```cpp
// Modifikasi Nilai
/* v1[0] = 1; Tidak bisa dilakukan karena v1
tidak memiliki index ke-0 (masih kosong) */
v1.push_back(1);
for (ll i = 5; i > 1; i--){
    v1.push_back(i);
}
for (auto &elm: v1){
    cout << elm << ' ';
} cout << '\n';

v2.pop_back();
v2.pop_back();
v2.insert(v2.begin(), 3);
v2.insert(v2.begin()+2, 4);
for (auto &elm: v2){
    cout << elm << ' ';
} cout << '\n';
```

```
Isi dari v1 setelah push_back:
1 5 4 3 2

Isi dari v2 setelah pop_back
dan insert: 3 0 4 0 0 0 0 0 0
0
```

VECTOR

```cpp
sort(v1.begin(), v1.end());
cout << "Setelah Sort: ";
for (auto &elm: v1) cout << elm << ' ';
cout << '\n';
cout << "Apakah ketemu: " << (find(v1.begin(), v1.end(), 6) != v1.end()) << endl;
/* Tidak Ketemu, berada pada end*/
```

```
Setelah Sort: 1 2 3 4 5
Apakah ketemu: 0
```

BINARY SEARCH

- lower_bound(): Mengembalikan sebuah iterator yang menunjuk ke **elemen pertama** dalam rentang [first, last) yang **tidak lebih kecil** (lebih besar sama dengan) dari nilai yang diberikan, atau last jika tidak ditemukan elemen seperti itu.
- upper_bound(): Mengembalikan sebuah iterator yang menunjuk ke **elemen pertama** dalam rentang [first, last) yang **lebih besar** dari nilai yang diberikan, atau last jika tidak ditemukan elemen seperti itu.
- binary_search(): Memeriksa apakah sebuah elemen yang **setara** dengan nilai yang diberikan muncul dalam rentang [first, last).

Note: iterator adalah generalisasi dari pointer, dimana iterator langsung menunjuk reference dari suatu data.

BINARY SEARCH

```cpp
vector <ll> v = {1, 2, 3, 3, 3, 6, 7, 9, 10, 11, 12}; // Size: 11
cout << "Binary Search 3: " << binary_search(v.begin(), v.end(), 3) << endl;
cout << "Index Lower Bound 3: " << lower_bound(v.begin(), v.end(), 3) - v.begin() << endl;
cout << "Index Upper Bound 3: " << upper_bound(v.begin(), v.end(), 3) - v.begin() << endl;
cout << "Index Lower Bound 8: " << lower_bound(v.begin(), v.end(), 8) - v.begin() << endl;
cout << "Index Upper Bound 8: " << upper_bound(v.begin(), v.end(), 8) - v.begin() << endl;
cout << "Index Lower Bound 13: " << lower_bound(v.begin(), v.end(), 13) - v.begin() << endl;
cout << "Index Upper Bound 13: " << upper_bound(v.begin(), v.end(), 13) - v.begin() << endl;
```

```
Binary Search 3: 1
Index Lower Bound 3: 2
Index Upper Bound 3: 5
Index Lower Bound 8: 7
Index Upper Bound 8: 7
Index Lower Bound 13: 11
Index Upper Bound 13: 11
```

THANK YOU