বিস্‌মিল্লাহির রহ্‌মানির রহিম

প্রথম খণ্ড

2015-10-17 20:04:06
লেখাটা শুরু করতেই কম্পিউটার ক্র্যাশ.. অজানা কারনে.. যাইহোক..
STL বা Standard Template Library নিয়ে পড়া শুরু করার আগে জানতে হবে Template কি জিসিন.. আমার কাছে যে বই আছে সেখান থেকে পড়া শুরু করলাম.. বালাগুরুটাই আগে ধরলাম Template বুঝার জন্য..

2015-10-17 20:18:39
আমরা লাস্ট একটা প্রবলেমের কোড দেখার সময় (c++ এর) নিচের হেডার ফাইলটা দেখছিলামঃ

```
#include<bits/stdc++.h>
```

গুগলিং করে জানলাম এইটাদিলে সব স্ট্যান্ডার্ড হেডার (C এবং C++) এর কোডে যুক্ত হয়ে যায়। মানে এইটা দিলে আর কোনো হেডার ফাইল যুক্ত করতে হবে না :) এতে কিকি হেডার ফাইল যুক্ত হয় তা নিচের লিঙ্কে দেখতে পার।
তথ্যসূত্রঃ
https://gist.github.com/eduarc/6022859
http://gcc.gnu.org/onlinedocs/libstdc++/manual/using_headers.html
এইটা সব যায়গাতে GCC কমপাইলার এ কাজ করে.. এমনকি ACM-ICPC তেও।
তথ্যসূত্রঃ
https://www.quora.com/Is-it-good-practice-to-use-include-bits-stdc++-h-in-programming-contests-instead-of-listing-a-lot-of-includes

সো এখন থেকে জাস্ট ইউজ দ্যা আপার হেডার ফাইল.. :)

bits/stdc++.h হেডার ফাইলে প্রায় সব standard হেডার ফাইল সংযুক্ত থাকে বলে মনে হতে পারে রানটাইম বেশী লাগবে! হেডারফাইল বেশী হলে কম্পাইলেশন সময় বেশী লাগতে পারে, কিন্তু রানটাইমে এর প্রভাব থাকার কথা না। কারন রানটাইমে যা লাগবে হেডারফাইল থেকে তাই নেয়া (call করা) হয়। হেডারফাইল একটা থাকলে কোনো ফাংশন কল করলে যে সময় লাগবে, একধিক থাকলেও একই সময় লাগবে। কারণ হেডার ফাংশনের কোনো কিছু কল করলে তা খুঁজতে হয়না, সরাসরি ঐ ফাংশনের অ্যাড্রেসে কল চলে যায়। তাই আমার মনে হয়না bits/stdc++.h ব্যবহার করলে কোনো রকম TLE (Time Limit Exceed) হবে। আর জানা মতে contest এর জ্ঞানীজনেরা এটা ব্যবহার করেন। (2015-11-05 23:08:04)

2015-10-17 20:35:57
Template দিয়ে প্রোগ্রামিং করাকে বলা হয় Generic Programming. Generic মানে বুঝায় সব কাজের জন্য কমন কিছু একটা ব্যবহার করা। এইটার মানে একটু পরেই বুঝবে। এইটা মূলত algorithm এই ব্যবহার করা হয়। যাতে একই algorithm দিয়ে বিভিন্ন Datastructure নিয়ে কাজ করা যায়।

Template অনেক যায়গাতেই ব্যবহার করা যায়। প্রথমে দেখা যাক Class এ Template এর ব্যবহার।

Class এ Template ব্যবহারের সাধারণ ফর্ম হল এরকমঃ

```
template<class type-name> class class-name
```

এখানে **type-name** এর যে কোনো একটা নাম দিতে হবে, যেটি ক্লাসের ভিতরে/বাহিরে **ডাটাটাইপের** স্থানে কোন লিখার সময় লিখতে হবে, মানে ক্লাসের সব ডাটাটাইপের (int, float, double, char) এর যায়গাতে **type-name** এ দেয়া নামটা ব্যবহার করব এবং আর ক্লাসের Object তৈরি করার সময় বলে দেয়া হবে এই বানানো নামটির স্থানে কোন ডাটাটাইপ ব্যবহার করা হবে।
দ্রষ্টব্য **type-name** এ দেয়া নামটি সব যায়গাতে দিতেই হবে এমন না। যেমন লুপ চালানোর জন্য int i ইত্যাদি variable নিতে হবে। এখানে **type-name** এ দেয়া নামটি দিলে সমস্যা হতেই পারে।

নিচে একটা খুউউউউবই সিম্পল একটা ক্লাসের template ব্যবহার দেখালামঃ

```
#include<bits/stdc++.h>

using namespace std;

template<class T> class math // ক্লাসের নাম math এবং এখানে ডাটাটাইপের নাম দিয়েছি "T"। এখানে যা-তা দেয়া যাবে।
{
public:
	T add(T a, T b) // যেহেতু T দিছিলাম টেমপ্লেটে, তাই এখানেও সব যায়গাতে T ব্যবহার করেছি।
	{
	return a+b;
	}
//উপরের ফাংশনটা যদি শুধু int এর জন্য করা হত তাহলে সব T এর যায়গাতে int বসত। এখন T দেয়াতে কি আমরা T কে যেকোনো ডাটা টাইপ দিয়েই Replace করতে পারব। নিচে Object বানানো দেখ।
};

int main()
{
	math<int> objInt;
	math<float> objFloat;
	math<double> objDouble;
	math<char> objChar;

//অবজেক্টের নামের পরে কৌনিক ব্রাকেট দিয়ে আমরা বলে দিয়েছি অবজেক্টটার ডাটাটাইপ কি হবে। এই কৌনিক ব্রাকেটের ভিতর যা দিব, সব T এর যায়গাতে সেই ডাটাটাইপ Replace হয়ে Object/class টা কাজ করবে।

	int a=5, b=10;
	float c=5.5, d=15.3;
	double e=25.3, f=25.5;
	char g='a' , h=2;

	cout << objInt.add(a, b) << endl;  ///Result: 15
	cout << objFloat.add(c, d) << endl; ///Result: 20.8
	cout << objDouble.add(e, f) << endl; ///Result: 50.8
	cout << objChar.add(g, h) << endl; ///c

	return 0;
}
```

আশা করি কোডটা দেখে template কি জিনিস তা বুঝা হয়েছে। দেখি আর কি জানতে পারি। ২০ মিনিট আছি জেগে আর..

2015-10-17 20:53:34

বালাগুরু তে Class Template এবং Function Template দিয়েই Template শেষ করেছে। তো আজ ফাংশন পর্যন্তই দেখব।

উপরের উদাহরণে দেখানো হয়েছে শুধু এক ধরনের ডাটাটাইপ সম্পূর্ণ ক্লাসের জন্য। তো যদি একাধিক ডাটাটাইপ ব্যবহার করতে হয় তখন কি করব???

একাধিক ডেটাটাইপের জন্য কি করতে হবে নিচের কোডে দেখা নো হয়েছে। কোড দেখলেই আশা করি সব পরিষ্কার হবেঃ

```
#include<bits/stdc++.h>

using namespace std;

template<class kala, class dhola> class math //এখানে ২টা Generic টাইপ, kala এবং dhola; আর ইচ্ছা হলে আরো যুক্ত করা যাবে কমা (,) দিয়ে।
{
public:
```

```
        void display(kala a, dhola b) // ফাংশনের ব্যবহার করা হয়েছে দুইটাই
        {
        cout << a << " AND " << b << endl;
        }
};

int main()
{
        math<int, float> objAbul;
        math<double, char> objKabul;

        objAbul.display(19, 71.9); /// 19 AND 71.9
        objKabul.display(19.71, 97); /// 19.71 AND a

        return 0;
}
```

বলার কিছু নাই.. প্রোগ্রামই সব বুঝা যাওয়া কথা.. :)

আজকের লাস্ট টপিক Function Template.

নিচের কোডে ফাংশন টেমপ্লেটের কাজ/ব্যবহার দেখানো হলঃ

```
#include<bits/stdc++.h>

using namespace std;

template<class ding> ding add(ding a, ding b) //শুরুতে শুধু আগের মত template<...> দিতে হবে। এবং কৌনিক ব্রাকেটের ভিতর যত গুলো Generic Datatype লাগবে কমা দিয়ে দিতে হবে আগের মত। এখানে শুধু একটা নেয়া হয়েছে।
{
        return a+b;
}

int main()
{
        int a=5, b=10;
        float c=5.5, d=15.3;
        double e=25.3, f=25.5;
        char g='a' , h=2;

        cout << add(a, b) << endl;  ///Result: 15
        cout << add(c, d) << endl; ///Result: 20.8
        cout << add(e, f) << endl; ///Result: 50.8
        cout << add(g, h) << endl; ///c

// এখানে ফাংশনের ক্ষেত্রে আলাদা করে ডাটাটাইপ লিখে দিতে হবে না। কম্পাইলার নিজেই সব বুঝে নিবে।

        return 0;
}
```

আশা করি ফাংশনে টেমপ্লেটের ব্যবহারও বুঝা গেছে।

আশা করি এবার STL এ যেত পারব এখন.. :)

দ্বিতীয় খণ্ড

STL হচ্ছে Template ব্যবহার করে তৈরি করা কিছু Class এবং Function এর এক বিশাল কালেকশন। এটি নিয়ে সংক্ষেপে যা না বললেই নয় তা বলার চেষ্টা করছি।

2015-10-19 21:51:52
STL এ তিন ধরনের জিনিস আছেঃ

1. Containers: ডাটা রাখার জন্য কিছু Data Structure: **Vector**, Deque, **List**, Set, Multiset, **Map**, multimap, stack, queue, priority_queue ইত্যাদি।
2. Algorithms: Container গুলোকে Traverse, Search, Sort, Merge ইত্যাদি করার জন্য পর্যাপ্ত algorithm function: count(), find(), copy(), remove(), replace(), reverse(), swap(), binary_search(), sort() ইত্যাদি।
3. Iterators: এগুলোকে pointer এর সাথে তুলনা করা হয়। ডাটাকে access/ব্যবহার করার জন্য এটার সাহায্য নেয়া হয়। পয়েন্টার যেভাবে কাজ করে এটাও সেভাবেই কাজ করে।

আমরা শুধুমাত্র Vector এবং Map নিয়ে আলোচনা করব। উপরের তিনটি জিনিসই এ আলোচনার ভিতর বর্ণনা করা হবে।

Vector

এটি এক ধরনের Array. তবে এর বিশেষ বৈশিষ্ট হল: এ যে কোন সময় এর আকার পরিবর্তন করতে পারে। এতে Random Access (যে কোনো সময় যে কোনো অবস্থানের মান নিয়ে কাজ করা) যায়। Unlimited ভ্যালু নিয়ে কাজ করার জন্য এর বিকল্প নেই।

Vector Declaration:

```
#include<vector>
...
...
...
vector<T> obj;
```

এখানে T এর যায়গাতে যে ডাটাটাইপ নিয়ে কাজ করা হবে তা বলে দিতে হবে। obj এর যায়গাতে vector অ্যারেটির নাম দিতে হবে।

Vector এর ব্যবহার Array থেকে আলাদা। push_back() একটি ফাংশন দিয়ে Vector এ ডাটা insert করতে হয়। তবে ডাটা output নেয়াটা Array এর মতই।
একটি সহজ প্রোগ্রাম দেখা যাকঃ

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
	vector<int> roll;
// roll নামে integer ডাটাটাইপের একটি vector ডিক্লেয়ার করা হয়েছে।

	int i;

	for(i=0 ; i<100 ; i++)
	{
		roll.push_back(i);
// push_back() ফাংশন ব্যবহার করা ইনপুট নেয়া হয়েছে।
```

```
        }

        for(i=0 ; i<100 ; i++)
        {
                cout << roll[i] << endl;
// Array এর স্টাইলেই Vector এ আউটপুট পাওয়া যায়।
        }

        return 0;
}
```

Vector এ সব ডাটাটাইপঃ int, float, double, char এবং string ধরনের ডাটা রাখা যাবে।
Vector এর কিছু প্রয়োজনীয় ফাংশনঃ

Functions

- at()
- back()
- begin()
- capacity()
- clear()
- empty()
- end()
- erase()
- insert()
- pop_back()
- push_back()
- resize()
- size()
- swap()

আরো একটি উদাহরণ দেখা যাক। এ উদাহরণে উপরের ফাংশন গুলোর কয়েকটির কাজ দেখানো হয়েছেঃ

```
#include<bits/stdc++.h>

using namespace std;

void display(vector<int> v)
{
        cout << endl << "\tDISPLAYing:" << endl;

        for(int i=0 ; i<v.size() ; i++)
        {
        cout << v[i] << endl;
        }
```

```
        cout << endl;
}

int main()
{
        vector<int> Data;

        int i, t;

        /// Inserting data 10 to 100 in Vector container "Data"
        for(i=1 ; i<11 ; i++)
        {
        Data.push_back(i*10);
        }

        display(Data);

        cout << "Delete last value:" << endl;
        Data.pop_back();

        display(Data);

        /* ---------------- */

        cout << "Get vector size:" << endl;
        cout << "\nSize: " << Data.size() << endl;

        cout << "\tErase one data:" << endl;

        cout << "      => Erase first data:" << endl;
        Data.erase(Data.begin());

        display(Data);

        /* ---------------- */

        cout << "      => Erase last data:" << endl;
        Data.erase(Data.begin() + Data.size()-1);

        display(Data);

        /* ---------------- */

        cout << "\tErase a sequence of data: (from location 3 to 5)" << endl;

        Data.erase(Data.begin()+2, Data.begin()+5);

        display(Data);

        /* ---------------- */

        cout << "Clear full Vector:" << endl;
        Data.clear();

        cout << "Check if Vector container is empty or not: " << endl;

        if(Data.empty()==true)
        cout << "Vector Container Empty!" << endl;
        else
        cout << "Vector Container NOT Empty!" << endl;
```

```
	return 0;
}
```

Vector এর সব প্রয়োজনীয় ফাংশন উপরের প্রোগ্রামে দেখানো হয়েছে।

তৃতীয় খণ্ড

2015-10-22 13:53:19

cin/cout vs printf/scanf:

- https://www.quora.com/Is-cin-cout-slower-than-scanf-printf/answer/Anders-Kaseorg
- https://gcc.gnu.org/onlinedocs/libstdc++/manual/io_and_c.html

কোনো ঝামেলা না করার জন্য পরের সব উদাহরণে printf() এবং scanf() ব্যবহার করব।

2015-10-22 14:09:24

Iterators

Iterator হচ্ছে এক ধরনের object যারা পয়েন্টারের মত কাজ করে। STL এর container এর সাথে এটি ব্যবহার করা যায়। প্রতিটি container এই ধরনের iterator দেয়া আছে। Container এ (বিশেষ করে মাঝ খানের দিকে) কোনো কিছু insert, erase করতে iterator বেশী ব্যবহার করা হয়।

container এর iterator অবজেক্ট ডিফাইন করার সাধারণ নিয়মঃ

```
container<T>::iterator itr_name;
```

এখানে container এর যায়গাতে vector, list, map ইত্যাদি হতে পারে।
T এর যায়গাতে যে container এর জন্য iterator ব্যবহার করা হবে তার datatype লিখতে হবে।

উহাদরণঃ

```
vector<int> v; /// vector object
vector<int>::iterator itr_v; /// vector iterator object for vector object v
```

Vector কে Traverse করার নিচের প্রোগ্রামে iterator এর ব্যবহার দেখানো হয়েছে।

```
#include<iostream>
#include<vector>
#include<cstdio>

using namespace std;

int main()
{
	/// Defining a Vector of 10 element; it auto initialize with Zero
	vector<int> v(10);

	/// Traverse Technique one
	for(int i=0;i < v.size(); i++)
	{
```

```
        printf("%d ", v[i]);
        }

        /// Traverse Technique Two by Iterator
        vector<int>::iterator itr = v.begin();

        while(itr!=v.end())
        {
        printf("%d ", *itr); /// *itr return the value of the vector object that itr points.

        itr++;
        }

        return 0;
}
```

- এখানে itr হচ্ছে Vector Iterator. এটি define এর সময় এর ভিতর v Vector এর begin পজিশন রাখা হয়েছে। এর পর while লুপ এ বলা হয়েছে itr এ যতক্ষন v Vector এর end পজিশন না আসবে ততক্ষন লুপটা চলবে। লুপের ভিতরে itr++ দিয়ে প্রতিবার পরের এলিমেন্টে যাওয়া হয়েছে।

- উপরের উদাহারণ থেকে বুঝা যাচ্ছে iterator দিয়ে সরাসরি এলিমেন্ট কে পয়েন্ট করা যায়। iterator এর যোগ-বিয়োগ পয়েন্টারের মত। যেমনঃ

- itr এ যদি প্রথমিক ভাবে প্রথম এলিমেন্ট থাকে কোনো container এর। তাহলে-

itr+1 : এটি ২ নাম্বার এলিমেন্টকে point করবে।

itr+2 : এটি ৩ নাম্বার এলিমেন্টকে point করবে।

itr+5 : এটি ৬ নাম্বার এলিমেন্টকে point করবে।

itr++ : এটি itr এর pointer পরিবর্তন করে পরের এলিমেন্ট, মানে ২ নাম্বার এলিমেন্টকে point করবে।

- Iterator এ পয়েন্টারের মত শুরুতে * (asterisk) ব্যবহার করে এলিমেন্টের মান পাওয়া যাবে।

insert()

iterator এর উপরের কাজ গুলো বুঝতে পারলে এবার vector এর insert() ফাংশনের ব্যবহার দেখা যাকঃ

insert() ফাংশনের সাধারণ ডেফিনিশন হলঃ

vector.insert(iterator_obj , total_new_elements , new_elements_value);

iterator_obj : এখানে একটি iterator অবজেক্ট দিতে হবে। যার point করবে vector এর যে অবস্থানে ডাটা insert করতে হবে তার পরের এলিমেন্টকে। যেমনঃ যদি vector এর ৩য় অবস্থানে ডাটা ইনসার্ট করতে হয় তাহলে iterator কে পয়েন্ট করতে হবে ৪র্থ এলিমেন্টকে। অর্থাৎ iterator যে এলিমেন্টকে পয়েন্ট করবে তার আগে নতুন এলিমেন্ট যুক্ত হবে।
total_new_element : মোট কয়টি এলিমেন্ট যুক্ত করতে হবে তা এখানে সংখ্যা দিয়ে বলে দিতে হবে।
new_elements_value : নতুন এলিমেন্টের মান কি হবে তা এখানে বলে দিতে হবে।

নিচের উদাহরণে insert() এর ব্যবহার দেখানো হলঃ

```
#include<iostream>
```

```cpp
#include<cstdio>
#include<vector>

using namespace std;

int main()
{
	vector<int> v;

	vector<int>::iterator itr;

	/// insert 15 elements
	for(int i=0 ; i< 15 ; i++)
	{
	v.push_back(i*10);
	}

	///display all value using iterator @ite
	itr = v.begin();
	while(itr != v.end())
	{
	printf("%d ", *itr);
	itr = itr+1;
	}

	cout << "\n\n";

	/* ------- */

	///insert single value middle of the vector

	itr = v.begin(); /// point to 1st element
	itr += 3; ///point to 4th element

	v.insert(itr, 1, 777); /// insert(position itr , total new element , new elements value)

	///displaying new dataset values using iterator @ite
	itr = v.begin();
	while(itr != v.end())
	{
	printf("%d ", *itr);
	itr = itr+1;
	}

	cout << "\n\n";

	/* ------- */

	///insert more than 1 value middle of the vector

	itr = v.begin(); /// point to 1st element
	itr += 5; ///point to 6th element

	v.insert(itr, 5, 2222); /// insert(position itr , total new element , new elements value)

	///displaying new dataset values using iterator @ite
	itr = v.begin();
	while(itr != v.end())
	{
	printf("%d ", *itr);
```

```
        itr = itr+1;
        }

        return 0;
}
```

Iterator দিয়ে erase() ফাংশনটিও ব্যবহার করা যায়। নিচে এরকম একটি উদাহরণ দেয়া হলঃ

```
#include<iostream>
#include<cstdio>
#include<vector>

using namespace std;

int main()
{
        vector<int> v;

        vector<int>::iterator itr;

        /// inserting some data in vector
        for(int i = 1 ; i<=15 ; i++)
        {
        v.push_back(i);
        }

        /// displaying all value using iterator
        itr=v.begin();
        while(itr != v.end())
        {
        printf("%d ", *itr);
        itr++;
        }

        /// erasing data from vector
        itr = v.begin();
        v.erase(itr+4, itr+10); /// erasing 5th element to 10th element using iterator @itr

        /// again displaying all value using iterator
        printf("\n\n");

        itr=v.begin();
        while(itr != v.end())
        {
        printf("%d ", *itr);
        itr++;
        }

        return 0;
}
```

iterator এর প্রাথমিক ব্যবহার এতটুকুই। সব ধরনের STL container এ একই ভাবে iterator ব্যবহার করা যাবে।

চতুর্থ খণ্ড

2015-10-28 21:00:50

List

STL এর আরেকটি container class হল list. এটিও vector এর মতই array এর বিকল্প হিসেবে কাজ করে।

বৈশিষ্ট্য সমূহ

- দুই দিক থেকেই অর্থাৎ front এবং back এ insert এবং delete করা যায়।
- random access করা যায়না। শুধু মাত্র sequentially access করা যায়। একে linked list এর সাথে তুলনা করা যায়।
- এটির নিজস্ব sort এবং merge ফাংশন রয়েছে।

List Declaration

```
#include<list>
...
...
...
list<T> obj;
```

এখানে T এর যায়গাতে যে ডাটাটাইপ নিয়ে কাজ করা হবে তা বলে দিতে হবে। obj এর যায়গাতে list অবজেক্টটির নাম দিতে হবে।

List এর প্রাথমিক ব্যবহার নিচের প্রোগ্রামে দেখানো হলঃ

```
#include<iostream>
#include<list>

using namespace std;

void display(list<int> &lst)
{
	///Displaying the data of list using list iterator
	list<int>::iterator itr;

	for(itr = lst.begin(); itr != lst.end() ; itr++)
	cout << *itr << " ";

	cout << endl;
}

int main()
{
	list<int> lst;

	int i;

	cout << "Inserting in back:" << endl;

	for(i=1 ; i<=10 ; i++)
	{
	lst.push_back(i);
	}

	display(lst);

	cout << "Deletion in back:" << endl;
```

```
        for(i=1; i<=5; i++)
        {
        lst.pop_back();
        }

        display(lst);

        cout << "Inserting in front:" << endl;
        for(i=1 ; i<=10 ; i++)
        {
        lst.push_front(i);
        }

        display(lst);

        cout << "Deletion in front:" << endl;
        for(i=1; i<=5; i++)
        {
        lst.pop_front();
        }

        display(lst);

        return 0;
}
```

List Class Functions

back()	
begin()	
clear()	
empty()	
end()	
erase()	
insert()	

merge()	এটা দুটো লিস্টকে merge করবে।
	যদি লিস্ট দুটো sorted হয় তাহলে এটি merge sort করবে। আর যদি unsorted হয় তাহলে শুধু একটার সাথে আরেকটা জোড়া লাগাবে।
	
	`int i;`
	`list<int> lst1, lst2;`
	
	`if(lst1.empty()==true)`
	`cout << "List empty!" << endl;`
	
	`for(i=1; i<=15; i++)`
	`{`
	`lst1.push_back(i);`
	`}`
	
	`for(i=1; i<=15; i++)`
	`{`

	lst2.push_back(i);
	}
	
	lst1.merge(lst2);
	
	list<int>::iterator itr;
	
	for(itr=lst1.begin(); itr != lst1.end(); itr++)
	{
	cout << *itr << endl;
	}
	Complexity: At most, linear in the sum of both container sizes minus one (comparisons).

pop_back()	
pop_front()	
push_back()	
push_front()	
remove()	
resize()	
reverse()	লিস্টকে reverse করে। **Complexity:** Linear in list size.
size()	
sort()	লিস্ট কে sort করে। **Complexity:** Approximately NlogN where N is the container size.
splice()	
swap()	
unique()	একই মান (value) একাধিক থাকলে সেগুলো মুছে দেয়। একাধিক মানের ভিতর শুধু প্রথম মানটি রাখে। এটি ফাংশন দিয়েও ব্যবহার করা যায়।

Which language is better? C++ or Java?

https://www.quora.com/What-is-the-best-language-for-Competitive-Programming
https://discuss.codechef.com/questions/47211/which-language-is-better-for-competitive-coding

পঞ্চম খণ্ড

2015-11-05 19:59:45
Map নিয়ে এবার গবেষনা করা যাক।

Map

Map হচ্ছে আরেকটি c++ কন্টেইনার যা pair ভ্যালু নিয়ে কাজ করে। pair value বলতে প্রতিটি মানের জন্য একটি unique key যুক্ত একটি vector/array।

বৈশিষ্ট্য

- স্বয়ংক্রিয় (auto) sort হয়।
- একই key একাধিক থাকতে পারবেনা।

Map Declaration

```
#include<map>
...
...
...
map<T1,T2> obj;
```

এখানে T1 এর যায়গাতে key এর ডাটাটাইপ এবং T2 এর যায়গাতে যে ডাটাটাইপ নিয়ে কাজ করা হবে তা বলে দিতে হবে। obj এর যায়গাতে map অবজেক্টটির নাম দিতে হবে।

Map এর সাধারন ব্যবহার নিচের প্রোগ্রামে দেখানো হলঃ

```
#include<iostream>
#include<map>
#include<string>

using namespace std;

void display(map<string,int> m)
{
	map<string,int>::iterator itr;

	for(itr=m.begin(); itr != m.end(); itr++)
	{
	cout << (*itr).first << " " << itr->second << endl;
	/// map is a pair value container. so its iterator has special function for getting the paired
value of an element of the map. Those functions are "first" for 1st value (called KEY) and "second"
(called Value) of an element.
	/// Map iterator used in two way for getting the paired value of an element:
	/// 1st way: (*itr).first and (*itr).second
	/// 2nd way: itr->first and itr->second
	/// Both two return the first and second value of the paired element.
	}

	cout << endl;
}

int main()
{
	map<string, int> contectList;

	///add/insert new value
	contectList["Emma Watson"] = 1111111;
	contectList["Angelina Jolie"] = 2222222;
	contectList["Daniel Radcliffe"] = 3333333;

	display(contectList);

	///insert new entry
	contectList["AAA"] = 444; ///insert rule 1

	contectList.insert(pair<string, int> ("BBB", 555)); ///insert rule 2

	display(contectList);

	///delete
	contectList.erase("AAA");
```

```
        display(contectList);

        ///find
        map<string, int>::iterator itr;
        itr = contectList.find("BBB");

        if(itr!=contectList.end())
        contectList.erase(itr);

        display(contectList);

        return 0;
}
```

Map Class Functions

begin()	
clear()	
empty()	
end()	
erase()	
find()	**Complexity:** Logarithmic in size.
insert()	
size()	
swap()	

Pair

Pair হচ্ছে একটি ক্লাস, Map এর প্রতিটি এলিমেন্ট একএকটি Pair Object. Map এর insert() ফাংশনে এর ব্যবহার উপরে দেখানো হয়েছে।

Pair Declaration

```
pair<T1,T2> obj;
```

এখানে T1 এর যায়গাতে key এর ডাটাটাইপ এবং T2 এর যায়গাতে যে ডাটাটাইপ নিয়ে কাজ করা হবে তা বলে দিতে হবে। obj এর যায়গাতে map অবজেক্টটির নাম দিতে হবে।

এক-একটি pair তৈরি হয় দুটি মান দিয়ে। প্রথমটিকে **first** এবং দ্বিতীয়টিকে **second** দিয়ে ব্যবহার (access) করা যায়। উপরে map এর জন্য তৈরি **display()** ফাংশনে map এর iterator এ **first** এবং **second** এর ব্যবহার দেখানো হয়েছে।

যেহেতু এটি Map এর সাথে ব্যবহার করা হচ্ছে তাই এ নিয়ে আর বিস্তারিত কিছু বলা হচ্ছে না।

Set

Set হচ্ছে STL এর আরেকটি container. এটি হুবহু map এর মত। শুধু পার্থক্য হল map এ pair element store করা যায়। আর set এ single value element store করা যায়।

বৈশিষ্ট্য

- স্বয়ংক্রিয় (auto) sort হয়।
- একই element/value একাধিক থাকতে পারবেনা।
- Set হচ্ছে binary search tree এর একটি implementation.

Map Declaration

```
#include<set>
...
...
...
set<T> obj;
```

এখানে T কে key বলা হয়, T এর যায়গাতে যে ডাটাটাইপ নিয়ে কাজ করা হবে তা বলে দিতে হবে। obj এর যায়গাতে set অবজেক্টটির নাম দিতে হবে।

set হুবুহু map এর মত ব্যবহার করা যাবে। map এর সব ফাংশন ও set এ আছে। তাই এটি নিয়ে আর কিছু বলা হচ্ছে না।

লেখকঃ মোঃ মাহমুদুল হাসান সোহাগ
http://about.me/shohag