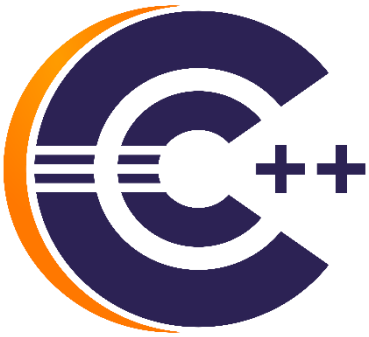
Part-2

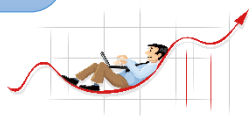


PROGRAMMING STL

WRITTEN BY SUJAN PRODHAN

Continue any part of the book under your own name. Offenses punishable by law.Various websites such as GitHub have been used to develop the book. You can visit prodhan2.blogspot.com to know more details about us.

COMPUTER SCIENCE AND ENGINEERING



In the **FIRST PART** I have added examples with explanation of examples. That will help you understand the examples..

Contents

C++ STL & Tricks-1

C++ এ কিছু build in লাইব্রেরী আছে, যাদেরকে বলা হয় standard template library. এইসব STL ব্যবহার করে অনেক প্রবলেম অনেক সহজে করে ফেলা যায়। যেমন কোন প্রবলেমে যদি stack,queue এইগুলা ব্যবহার করা লাগে, এইগুলার জন্য নতুন করে কোড লেখার প্রয়োজন পরে না। stack,queue stl ব্যবহার করেই প্রবলেম সল্ভ করা যায়।

vector:

ভেক্টর নরমাল এ্যারের মতই। এর সুবিধা হল যখন ইচ্ছে সাইজ পরিবর্তন করা যায়।

1.vector declare:

vector ব্যবহার করার জন্য header file vector include করতে হবে।

```
vector < type > Name;
```

```
#include<stdio.h>
#include<vector>
using namespace std;
int main()
{
   vector<int>v1;
   vecotr<char>v2;
   vecotr<long long>v3;
   return 0;
}
```

Vector capacity, element access, modifiers:

ভেক্টরের সাইজ জানার জন্য ব্যবহার করা হয়, size() public member function. এটি ভেক্টরের সাইজ রিটার্ন করে। complexity O(1)

ভেক্টর empty কিনা, তা জানার জন্য ব্যহহার করা হয় empty() public member function. এটি বুলিয়ান ভ্যালু রির্টান করে। ভেক্টর empty হলে true otherwise false রির্টান করে। complexity O(1)

ভেক্টরের element নরমাল এ্যারের মত [ ] অপারেটর দিয়ে access করা যায়। তাছাড়া at() public member function দিয়েও যেকোন পজিশনের ভ্যালু এ্যাকসেস ও পরিবর্তন করা যায়।

2.vector element access.cpp

```
for(int i=0;i<v.size();i++)
{
   cout<<v[i]<<" ";  //print every element of vector
}

for(int i=0;i<v.size();i++)
{
   v[i]+=10;        // add 10 to each element of vector
}
```

STL container যেগুলোতে begin() & end() public member function আছে, সেইগুলা iterator ব্যবহার করেও element access করা যায়।

3.vector access with iterator.cpp:

vector< Type >::iterator Name;

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
   vector<int>v;
   for(int i=0;i<10;i++)v.push_back(rand()%100);
   cout<<"Initial vector: ";
   for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
   cout<<endl;

   vector<int>::iterator it;
   cout<<"Access with iterator: ";
   for(it=v.begin();it!=v.end();it++)
   {
      cout<<(*it)<<" ";
   }
   cout<<endl;
   return 0;
}
```

ভেক্টরে সরাসরি ইনপুট নেয়ার কোন উপায় নেই। এর জন্য আলাদাভাবে ভ্যালু ইনপুট নিয়ে ভেক্টর modifier দিয়ে ভেক্টরে ভ্যালু assign করতে হয়। সাধারণত push_back use করে এ্যাসাইন করা হয়। এর কাজ হল ভেক্টর লিস্টের শেষে একটি এলিমেন্ট যোগ করে দেয়া।

v.push_back(x); এর মানে হল 'x' ভেক্টরের শেষে যোগ হয়ে যাবে।

4.vector user input.cpp

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
   vector<int>v;
   int N;
   cin>>N;
   for(int i=0;i<N;i++)
   {
      int x;
      cin>>x;
      v.push_back(x);
   }
   return 0;
}
```

vector এ ইনপুট নেয়ার আরেকটি উপায় হল, নির্দিষ্ট সাইজের ভেক্টর declare করে তাঁতে, scanf/cin use করে ইনপুট নেয়া।

5.vector input.cpp

```
#include<stdio.h>
#include<vector>
using namespace std;
int main()
{
   vector<int>v(10);
   for(int i=0;i<v.size();i++)
   {
      scanf("%d",&v[i]);
   }
   return 0;
}
```

pop_back use করে ভেক্টরের শেষের element রিমুভ করে দেয়া হয়।

insert ভেক্টরের একটি নির্দিষ্ট পজিশনে ভ্যালু insert করা।

erase ভেক্টর থেকে নির্দিষ্ট পজিশনের ভ্যালু মুছে দেয়া বা একটা নির্দিষ্ট পজিশনের রেঞ্জের ভ্যালু মুছে দেয়ার জন্য ব্যবহার করা হয়। Complexity Linear.

clear পুরো ভেক্টর empty করে দিতে ব্যবহার করা হয়।

6. pop_back, insert, erase, size, empty.cpp

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
   vector<int>v;
   v.push_back(199);
   v.push_back(34);
   v.push_back(1325);
   v.push_back(45);
   v.push_back(8989);

   cout<<"Initial vector size "<<v.size()<<endl;
   cout<<"Initial vector: ";
   for(int i=0;i<v.size();i++)
   {
      cout<<v[i]<<" ";
   }
   cout<<endl;

   v.pop_back();
   cout<<"After pop_back: ";
   for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
   cout<<endl;

   v.insert(v.begin(),34);
   cout<<"After Insert 1st position: ";
   for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
   cout<<endl;

   v.insert(v.begin()+2,345);
   cout<<"After Insert 3rd position: ";
```

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

    v.erase(v.begin()+1);
    cout<<"After erase one element from 2nd position: ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;

    v.erase(v.begin()+2,v.begin()+4);
    cout<<"After erase 3rd to 4th position: ";
    for(int i=0;i<v.size();i++) cout<<v[i]<<" ";
    cout<<endl;

    v.clear();
    cout<<"Is vector empty "<<v.empty()<<endl;

    return 0;
}
```

```
Initial vector size 5
Initial vector: 199 34 1325 45 8989
After pop_back: 199 34 1325 45
After Insert 1st position: 34 199 34 1325 45
After Insert 3rd position: 34 199 345 34 1325 45
After erase one element from 2nd position: 34 345 34 1325 45
After erase 3rd to 4th position: 34 345 45
Is vector empty
```

এখানে v.begin() vector এর শুরুর iterator. এইটা অনেকটা পয়েন্টারের মত।

Line 27 এ v.begin() মানে ভেক্টরের প্রথমে ভ্যালু insert করা হচ্ছে।

Line 32 এ v.begin() মানে হচ্ছে ভেক্টরের 3rd position এ ভ্যালু insert করা হচ্ছে।

Line 42 এ v.erase(v.begin()+2,v.begin()+4) মানে হচ্ছে ভেক্টরের 3rd to 4th পজিশন পর্যন্ত সব এলিমেন্ট রিমুভ করে দেয়া। এখানে রেঞ্জ [2 , 4) এইভাবে কাজ করছে। including first limit, excluding last limit. (x<=i<y)

Some advance operation in vector:

Size define করে ভেক্টর declare করা ও initial value set করে ভেক্টর declare করা যায়।

vector< type >Name(Size);

vector< type >Name(Size , InitialValue);

7.vector size define declare.cpp

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int>v1(10);
    cout<<"vector v1: ";
    for(int i=0;i<v1.size();i++)cout<<v1[i]<<" ";
    cout<<endl;
```

```
    vector<int>v2(10,-1);
    cout<<"vector v2: ";
    for(int i=0;i<v2.size();i++)cout<<v2[i]<<" ";
    cout<<endl;

    return 0;
}
```

vector v1: 0 0 0 0 0 0 0 0 0 0

vector v2: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

size define করে initial value ছাড়া ভেক্টর declare করলে initially vector এর সব এলিমেন্ট zero থাকে।

vector এর সাইজ পরিবর্তন করার জন্য resize() member function use করা হয়।

8. vector Resize.cpp

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int>v;
    for(int i=1;i<=10;i++)v.push_back(i);

    cout<<"Initial vector: ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;

    v.resize(5);
    cout<<"After resize(5): ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;

    v.resize(8,100);
    cout<<"After resize(8,100): ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;

    v.resize(15);
    cout<<"After resize(15): ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;

    return 0;
}
```

Initial vector: 1 2 3 4 5 6 7 8 9 10

After resize(5): 1 2 3 4 5

After resize(8,100): 1 2 3 4 5 100 100 100

After resize(15): 1 2 3 4 5 100 100 100 0 0 0 0 0 0 0

accumulate(), min_element(), max_element() দিয়ে ভেক্টর থেকে সহজে যোগফল, মিনিমাম ও ম্যাক্সিমাম নাম্বার বের করে ফেলা যায় এবং

sort() function দিয়ে সর্ট করা যায়। এইগুলা জন্য algorithm header file include করতে হবে।

9. Sum Min Max sort.cpp

```
#include<iostream>
#include<vector>
#include<stdlib.h>
#include<algorithm>
#include <numeric>
using namespace std;
int main()
{
   vector<int>v;
   for(int i=0;i<10;i++)v.push_back(rand()%100);
   cout<<"Initial vector: ";

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

   cout<<"sum of all element:
"<<accumulate(v.begin(),v.end(),0)<<endl;
   cout<<"Min element of vector:
"<<*min_element(v.begin(),v.end())<<endl;
   cout<<"Max element of vector:
"<<*max_element(v.begin(),v.end())<<endl;
   sort(v.begin(),v.end());

   cout<<"After sort: ";
   for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
   cout<<endl;
   return 0;
}
```

```
Initial vector: 83 86 77 15 93 35 86 92 49 21
sum of all element: 637
Min element of vector: 15
Max element of vector: 93
After sort: 15 21 35 49 77 83 86 86 92 93
```

Make vector element unique:

10.unique vector.cpp

```
#include<iostream>
#include<vector>
#include<stdlib.h>
#include<algorithm>
using namespace std;
int main()
{  vector<int>v;
   for(int i=0;i<10;i++)v.push_back(rand()%100);
   cout<<"Initial vector: ";
   for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
```

```
    cout<<endl;
    sort(v.begin(),v.end());
    v.erase(unique(v.begin(),v.end()),v.end());
    cout<<"Unique vector: ";
    for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
    cout<<endl;
    return 0;
}
```

Initial vector: 83 86 77 15 93 35 86 92 49 21

Unique vector: 15 21 35 49 77 83 86 92 93

List:

ভেক্টরের মত লিস্ট ও এক ধরনের ডাইনামিক কনটেইনার।

List declare:

List container এর জন্য list header file include করতে হবে।

list< Type >Name;

list< int >List;

List capacity, element access, modifiers:

vector এর মত list প্রায় সকল অপারেশন একই রকমের। list এর element vector মত [ ] অপারেটর দিয়ে access করা যায় না। list এর সকল element access এর জন্য iterator ব্যবহার করতে হয়। list resize করার জন্য কিছু নেই। তবে push_front(), pop_front() দুইটি public member function আছে, যা দিয়ে list এর প্রথমে element add করা যায় ও প্রথম থেকে element remove করা যায়।

11. list.cpp

```
#include<iostream>
#include<list>
#include<stdlib.h>
using namespace std;
int main()
{
    list<int>List;
    list<int>::iterator it;

    for(int i=0;i<10;i++)List.push_back(rand()%100);

    cout<<"List size: "<<List.size()<<endl;

    cout<<"Initial list: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<<" ";
    cout<<endl;

    cout<<"First element: "<<List.front()<<" Last element: "<<List.back()<<endl;

    List.push_front(10000);

    cout<<"After push element in front: ";
```

```
for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
   cout<<endl;


   List.pop_back();
   List.pop_front();


   cout<<"After poped element from front &
back: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
   cout<<endl;



   return 0;
}
```

Initial list: 83 86 77 15 93 35 86 92 49 21

First element: 83 Last element: 21

After push element in front: 10000 83 86 77 15 93 35 86 92 49 21

After poped element from front & back: 83 86 77 15 93 35 86 92 49

Some advance operation in list:

list এর কিছু public member function হল sort(), remove(), unique(). sort এর time complexity O(nlogn), remove(), unique() এর time complexity linear O(n).

12. list sort, unique, remove .cpp

```
#include<iostream>
#include<list>
#include<stdlib.h>
using namespace std;
int main()
{
   list<int>List;
   list<int>::iterator it;
   for(int
i=0;i<10;i++)List.push_back(rand()%100);
   List.push_front(100);
   List.push_back(100);


   cout<<"Initial list: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
   cout<<endl;


   List.sort();


   cout<<"After sort: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
   cout<<endl;


   List.unique();
```

```
    cout<<"After quniue: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
    cout<<endl;


    List.remove(100);
    List.remove(73);


    cout<<"After remove: ";

for(it=List.begin();it!=List.end();it++)cout<<(*it)<
<" ";
    cout<<endl;


    return 0;
}
```

Initial list: 100 7 49 73 58 30 72 44 78 23 9 100

After sort: 7 9 23 30 44 49 58 72 73 78 100 100

After quniue: 7 9 23 30 44 49 58 72 73 78 100

After remove: 7 9 23 30 44 49 58 72 78

বিদ্রঃ list/vector এর সকল element unique করার জন্য আগে sort করে নিতে হবে। কারণ unique ফাংশন পাশাপাশি দুইটি ভ্যালু সেম হলে একটি সেভ করে। এইজন্য আগে সর্ট করে নিলে সবগুলো element unique হবে।

list এর মধ্যে সবচেয়ে কার্যকরী অপারেশন হল, দুইটি লিস্ট O(1) time complexity তে সংযুক্ত করা। এর জন্য list এর splice() public member function ব্যবহার করা হয়।

13. slpice.cpp

```
#include<iostream>
#include<list>
#include<stdlib.h>
using namespace std;
int main()
{
    list<int>List1,List2;
    list<int>::iterator it;
    for(int
i=0;i<10;i++)List1.push_back(rand()%100);
    for(int
i=0;i<5;i++)List2.push_back(rand()%100);


    cout<<"Initial list1: ";

for(it=List1.begin();it!=List1.end();it++)cout<<(*i
t)<<" ";
    cout<<endl;


    cout<<"Initial list2: ";

for(it=List2.begin();it!=List2.end();it++)cout<<(*i
t)<<" ";
    cout<<endl;


    List1.splice(List1.end(),List2);
```

```
    cout<<"After splice: ";

for(it=List1.begin();it!=List1.end();it++)cout<<(*it)<<" ";
    cout<<endl;


    return 0;
}
```

```
Initial list1: 7 49 73 58 30 72 44 78 23 9
Initial list2: 40 65 92 42 87
After splice: 7 49 73 58 30 72 44 78 23 9 40 65 92 42 87
```

বিদ্রঃ splice অপারেশনে time complexity সবসময় O(1) নয়। যখন কোন লিস্টের মাঝে ইটারেটর দিয়ে আরেকটি লিস্ট সংযুক্ত করা হয় তখন time complexity O(n) হয়ে যায়। n হল number of element of list. কিন্তু কোন লিস্টের প্রথমে বা শেষে অন্য লিস্ট সংযুক্ত করতে সবসময় O(1).

String:

string নরমাল স্ট্রিং এর মতই। তবে এর কিছু ফিচার আছে, যার জন্য string ব্যবহার করা অনেক সহজ। string class ব্যবহারের জন্য string header file include করতে হবে।

string capacity, element access, modifiers:

string এর প্রতিটি character [] অপারেটর দিয়ে access করা যায়।

দুইটি string কম্পেয়ার করার জন্য '==' অপারেটর ব্যবহার করা হয়।

string concat করার জন্য ব্যবহার করা হয় '+'

string append করার জন্য '+=' ব্যবহার করা যায়।

vector এর মত string এর push_back(), pop_back(), erase(), size() public member function আছে।

14. string.cpp

```
#include<iostream>
#include<string>
#include<stdlib.h>
using namespace std;
int main()
{
    string str0="ABCDEFGH";
    string str1="ABCDEFGH";
    string str2="Extinction is the rule.";
    string str3="Survival is the exception.";
    cout<<"Is str0 & str1 same: "<<(str0==str1)<<endl;
    string str4=str2+" "+str3;
    cout<<"Concat str2 & str3: "<<str4<<endl;

    for(char i='a';i<='g';i++)
    {
        str0+=i;
```

```
}
   cout<<"New str0: ";
   for(int i=0;i<str0.size();i++)cout<<str0[i];
   cout<<endl;

   str0.erase(0,3);

   cout<<"After erase operation in str0:
"<<str0<<endl;

   return 0;
}
```

```
Is str0 & str1 same: 1

Concat str2 & str3: Extinction is the rule.
Survival is the exception.

New str0: ABCDEFGHabcdefg

After erase operation in str0: DEFGHabcdefg
```

string এ ইনপুট নেয়ার জন্য cin/getline() ব্যবহার করা লাগে। আউটপুটের জন্য cout, আর যদি printf() দিয়ে প্রিন্ট করা লাগে তাহলে c_str() public member function দিয়ে c string এ কর্নভার্ট করে নেয়া লাগে।

15.string input.cpp

```
#include<iostream>
#include<string>
#include<stdlib.h>
using namespace std;
int main()
{
   string inp;
   cin>>inp;
   cout<<"First input: "<<inp<<endl;
   cin.ignore();
   getline(cin,inp);
   cout<<"Second input: "<<inp<<endl;
   printf("%s",inp.c_str());
   return 0;
}
```

Some advance operation in string:

min(),max() function দিয়ে দুইটি string A ও B এর মধ্যে lexicographically smallest ও largest স্ট্রিং বের করা যায়।

find() public member function দিয়ে string এ অন্য একটি প্যাটার্ন খুঁজে বের করা যায়। এটি প্রথম থেকে খুঁজে পাওয়া প্যাটার্নের পজিশন রিটার্ন করে। শেষ থেকে প্যাটার্ন স্ট্রিং খুঁজে বের করার জন্য rfind() ব্যবহার করা হয়।

String এর যেকোন পজিশন থেকে যেকোন সাইজের sub string এর জন্য substr() public member function ব্যবহার করা হয়।

16. String operation.cpp

```
#include<iostream>
#include<string>
#include<stdlib.h>
using namespace std;
int main()
```

```
{
   string str="somewhere, something incredible is waiting to be known.";


   cout<<"Find some from first: "<<str.find("some")<<endl;
   cout<<"Find some from last: "<<str.rfind("some")<<endl;


   cout<<"substring: "<<str.substr(0,9)<<endl;
   cout<<"substring: "<<str.substr(11,9)<<endl;


   cout<<"lexicographically small: "<<min(str.substr(0,9),str.substr(11,9))<<endl;
   cout<<"lexicographically large: "<<max(str.substr(0,9),str.substr(11,9))<<endl;


   return 0;
}
```

```
Find some from first: 0
Find some from last: 11
substring: somewhere
substring: something
lexicographically small: something
lexicographically large: somewhere
```

Stack:

stack হল এমন একধরনের ডাটাস্ট্রাকচার যা, "last in, first out" প্রপার্টি মেনে চলে। এরমানে হল এমন একটি লিস্ট, যেই লিস্টে যে সবার পরে insert হবে সে সবার আগে বের হবে। linked list দিয়ে স্ট্যাক implement করা যায়। কিন্তু stl stack ব্যবহার করলে সেই জিনিস আর করা লাগে না।

Stack declare:

stack ব্যবহারের জন্য header file stack include করতে হবে।

stack< Type >Name;

stack< int >Name;

Stack operation:

stack এর প্রধান অপারেশন হল, push(), pop(), top() public member function. push() দিয়ে stack এর শেষে নতুন element যোগ করা হয়। pop() দিয়ে stack এর শেষে যোগ হওয়া element ফেলে দেয়া হয়। top() দিয়ে stack এর সবার শেষে বা সবার উপরের element access করা হয়। stack এর কোন begin(), end() iterator return করে এমন public member function নেই। তাই পুরো stack একসাথে iterate করার কোন উপায় নেই।

তাছাড়াও stack এর আরও দুইটি কার্যকরী public member function হল, size(), empty().

17.stack.cpp

```
#include<iostream>
#include<stack>
#include<stdlib.h>
```

```
using namespace std;
int main()
{
   stack<int>S;
   for(int i=0;i<5;i++)
   {
      int x=rand()%100;
      cout<<"Push: "<<x<<endl;
      S.push(x);
   }

   cout<<"Stack size: "<<S.size()<<endl;

   cout<<"Top element: "<<S.top()<<endl;

   S.pop();

   cout<<"After pop top element:
"<<S.top()<<endl;

   while(!S.empty())S.pop();

   cout<<"Is stack empty: "<<S.empty()<<endl;

   return 0;
}
```

```
Push: 83
Push: 86
Push: 77
Push: 15
Push: 93
Stack size: 5
Top element: 93
After pop top element: 15
Is stack empty: 1
```

Queue:

queue ডাটাস্ট্রাকচার "first in first out" প্রোপার্টি মেনে চলে। মানে লিস্টে যে element সবার আগে add হবে, সেই element সবার আগে বের হবে।

Queue declare:

queue ব্যবহারের জন্য header file queue include করতে হবে।

queue< Type >Name;

queue< int >Name;

Queue operation:

queue এর অপারেশন গুলা হল, push(), front(), pop(). push() public member function দিয়ে queue তে element এ্যাড করা হয়। front() দিয়ে সবার সামনের (যে আগে এ্যাড হয়েছিল) সেইটা access করা হয়। pop() দিয়ে সবার উপরের element ডিলিট করে/ ফেলে দেয়া হয়। স্ট্যাকের মতও queue তে public member function begin(), end() না থাকায় সকল element iterate করা যায় না।

queue এর সাইজ এর জন্য size() এবং empty কিনা তা জানার জন্য empty() member function ব্যবহার করা হয়।

18. queue.cpp

```
#include<iostream>
#include<queue>
#include<stdlib.h>
using namespace std;
int main()
{
   queue<int>Q;
   for(int i=0;i<5;i++)
   {
      int x=rand()%100;
      cout<<"Insert "<<x<<endl;
      Q.push(x);
   }

   cout<<"Q front element: "<<Q.front()<<endl;
   Q.pop();
   cout<<"Q front after pop: "<<Q.front()<<endl;
   cout<<"Queue size: "<<Q.size()<<endl;
   while(!Q.empty())Q.pop();
   cout<<"Is queue empty: "<<Q.empty()<<endl;
   return 0;
}
```

```
Insert 83
Insert 86
Insert 77
Insert 15
Insert 93
Q front element: 83
Q front after pop: 86
Queue size: 4
Is queue empty: 1
```

Deque:

Double ended queue. এর front ও back উভয় দিকে থেকে ভ্যালু add করা যায় ও remove করা যায়।

deque declare:

deque ব্যবহারের জন্য header file deque include করতে হবে।

```
deque< Type >Name;
deque< int >Name;
```

deque operation:

deque এর কিছু অপারেশন queue এর মতই। front থেকে element add এবং remove করার জন্য push_front() ও pop_back() ব্যবহার করা হয়। Back থেকে element add এবং remove করার জন্য push_back() ও pop_back() ব্যবহার করা হয়।

তাছাড়া deque [] operator দিয়ে access করা যায়। এর size(), erase(), clear() public member function আছে।

Priority Queue:

priority queue এমন এক ধরনের কিউ যেখানে front element সবসময় বড়টা থাকে, বা যার priority বেশি সে থাকে। priority queue এর property চেঞ্জ করে ছোট element সবসময় front এ রাখা যায়।

Priority queue declare:

Priority queue ব্যবহারের জন্য header file queue include করতে হবে।

priority_queue< Type >Name;

priority_queue< int >Name;

Priority queue operation:

priority queue অপারেশন গুলো queue এর অপারেশনের মতই। তবে এখানে front() এর পরিবর্তে আছে top(), যা সবসময় priority queue তে থাকা বড় ভ্যালু (by default) রির্টান করে।

Set:

set হল এমন এক ধরনের লিস্ট যেখানে সব element একবার করে থাকবে। মানে একটি value যতবারেই insert করা হোক না কেন, থাকবে একবার। set সকল element কে sorted আকারে রাখে।

Set declare:

set ব্যবহারের জন্য header file set include করতে হবে।

set< Type >Name;

set< int >Name;

Set operation:

set উল্লেখযোগ্য অপারেশনগুলো হল, insert(), erase(), clear(), find(), size(), empty(). set এ begin() ও end() public member function থাকায় iterator দিয়ে পুরো set iterate করা যায়।

19.set.cpp

```
#include<iostream>
#include<set>
using namespace std;
int main()
{
   set<int>S;
   set<int>::iterator it;
   for(int i=1;i<100;i++)
   {
      S.insert(i%10);
   }
   cout<<"Set size: "<<S.size()<<endl;
   cout<<"Set element: ";
for(it=S.begin();it!=S.end();it++)cout<<(*it)<<" ";
   cout<<endl;
   set<int>::iterator it1,it2;
   it1=S.find(4);
   it2=S.find(7);
   S.erase(it1,it2);
   cout<<"Set element after erase 4 to 6: ";
for(it=S.begin();it!=S.end();it++)cout<<(*it)<<" ";
```

```
    cout<<endl;
    S.clear();
    cout<<"Is set empty: "<<S.empty()<<endl;
    return 0;
}
```

```
Set size: 10
Set element: 0 1 2 3 4 5 6 7 8 9
Set element after erase 4 to 6: 0 1 2 3 7 8 9
Is set empty: 1
```

set এর insert(),find(),erase() সবগুলার time complexity O(log2(n))

Map:

কোন কিছুকে কী-ভ্যালু দিয়ে ম্যাপ করে রাখার জন্য map stl ব্যবহার করা হয়। ম্যাপের ব্যবহারের একটি বড় সুবিধা হল, ডাইনামিক হওয়াতে যেকোন class,string, variable ম্যাপিং করা যায়।

map declare:

map ব্যবহারের জন্য header file map include করতে হবে।

```
map< key_Type , value_type >Name;
map< int, char >Name;
```

map operation:

map ভ্যালু assign করতে হয় Map[key]=value এইভাবে। একটা নির্দিষ্ট key এর ভ্যালু পাওয়া জন্য value=Map[key] নরমাল এ্যারে index এর মত access. map এ সকল element তার key value দিয়ে sorted আকারে থাকে। map এর সকল element iterator দিয়ে iterate করা যায়। find() public function দিয়ে যেকোন key map এ আছে কিনা তা খুঁজে বের করা যায়। erase() public member function দিয়ে যেকোন key remove করে দেয়া যায়।

20. map.cpp

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
    map<char,int>Map;
    map<char,int>::iterator it;
    for(char c='a';c<='j';c++)
    {
        int value=(int)c;
        Map[c]=value;
    }
    cout<<"Map size: "<<Map.size()<<endl;
    cout<<"Map key value: \n";
    for(it=Map.begin();it!=Map.end();it++)
    {
        cout<<"Key-> "<<(*it).first<<" value-> "<<(*it).second<<endl;
    }
    cout<<"Key a value: "<<Map['a']<<endl;
    if(Map.find('a')!=Map.end())cout<<"key 'a' found"<<endl;
    else cout<<"key 'a' not found"<<endl;
```

```
    if(Map.find('z')!=Map.end())cout<<"key 'z' found"<<endl;

    else cout<<"key 'z' not found"<<endl;

    Map.clear();

    cout<<"Is map empty: "<<Map.empty()<<endl;


    return 0;

}
```

```
Map size: 10
Map key value:
Key-> a value-> 97
Key-> b value-> 98
Key-> c value-> 99
Key-> d value-> 100
Key-> e value-> 101
Key-> f value-> 102
Key-> g value-> 103
Key-> h value-> 104
Key-> i value-> 105
Key-> j value-> 106
Key a value: 97
key 'a' found
key 'z' not found
Is map empty: 1
```

map এ সকল অপারেশনের time complexity logarithmic in size. O(log2(N)).

Caution:

map এ specific key আছে কিনা তা দুইভাবে চেক করা যায়। প্রথম উপায় হল Map[key]==0 তাহলে সেই key ম্যাপে নেই আরেকটা উপায় হল find() public member function use করে।

21.map key value search technique 1.cpp

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
    map<char,int>Map;
    for(char c='a';c<='j';c++)
    {
        int value=(int)c;
        Map[c]=value;
    }

    for(char c='a';c<='z';c++)
    {
        if(Map[c]==0)cout<<"key "<<c<<" not found"<<endl;
        else cout<<"key "<<c<<" found"<<endl;
    }
```

```
    return 0;
}
```

22.map key value search technique 2.cpp

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
    map<char,int>Map;
    for(char c='a';c<='j';c++)
    {
        int value=(int)c;
        Map[c]=value;
    }

    for(char c='a';c<='z';c++)
    {
        if(Map.find(c)!=Map.end())cout<<"Key "<<c<<" found"<<endl;
        else cout<<"Key "<<c<<" not found"<<endl;
    }
    return 0;
}
```

এখানে উপরে দুই ধরনের সার্চ technique এ প্রথম technique এ একটু ঝামেলা আছে। প্রথম পদ্ধতিতে প্রত্যেকবার সার্চ করার সময় ম্যাপে specific key value না থাকলেও ওই key value দিয়ে নতুন একটি স্লট map এ allocate করে নিচ্ছে এইভাবে Map[key]=0. এতে করে ম্যাপের সাইজ প্রত্যেক সার্চে বেরে যাচ্ছে। প্রোগ্রামিং কন্টেস্ট এ এই টেকনিক memory limit exceed error এর একটা কারণ হতে পারে। তাই সবসময় key search করার জন্য দ্বিতীয় technique use করা ভাল।

নিচের কোডটুকু দেখলে সমস্যাটা আরও ভাল করে বুঝা যাবেঃ

23. map search fault.cpp

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
    map<char,int>Map;
    for(char c='a';c<='j';c++)
    {
        int value=(int)c;
        Map[c]=value;
    }
    cout<<"Initially map size: "<<Map.size()<<endl;
    for(char c='a';c<='z';c++)
    {
```

```
    if(Map[c]==0){}
  }
  cout<<"After Search map size: "<<Map.size()<<endl;
  return 0;
}
```

Initially map size: 10

After Search map size: 26

এখানে দেখা যাচ্ছে সার্চ করার আগে ম্যাপের সাইজ ছিল 10, সার্চ করার পর বেড়ে হয়েছে 26.

iterator:

C++ এর iterator অনেকটা পয়েন্টারের মত, যা কোন container object এর পয়েন্টার নিয়ে একটা নির্দিষ্ট রেঞ্জের মধ্যে প্রতিটা element iterate করতে পারে। যেসব stl গুলাতে begin(),end() public member function আছে সেইগুলা iterator দিয়ে iterate করা যায়।

iterator declare এর নিয়ম হলঃ

container::iterator name;

যেমনঃ set এর iterator declare করতে হলে, set::iterator it; এইভাবে করতে হবে।

upper bound & lower bound:

C++ এর lower_bound() function sorted container এর রেঞ্জ [first,last) এর মধ্যে এমন একটি iterator position return করে যেখানে সব element range এর প্রথম থেকে ওই পর্যন্ত < val থাকে।

C++ এর upper_bound() function sorted container এর রেঞ্জ [first,last) এর মধ্যে এমন একটি iterator position return করে যেখানে সব element range এর প্রথম থেকে ওই পজিশন পর্যন্ত <=val থাকে।

upper_bound, lower_bound algorithm header file এ আছে। এদের time complexity O(log2(N)).

uppper_bound(), lower_bound() use করে sorted container এর রেঞ্জ এর মধ্যে একটা নির্দিষ্ট রেঞ্জ এর নাম্বার কতগুলো আছে তা সহজে বের করে ফেলা যায়।

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main ()
{
  int myints[] = {10,20,30,30,20,10,10,20,50,1,39,40};
  vector<int> v(myints,myints+12);
  sort (v.begin(), v.end());
  vector<int>::iterator low,up;
  low=lower_bound (v.begin(), v.end(), 20);
  up= upper_bound (v.begin(), v.end(), 30);
  cout<<"Total element in range [20,30]: "<<up-low<<endl;
  return 0;
}
```