STL <algorithm> sort

<container>

<iterator> vector

<function>

VECTOR - динамический массив

#include <vector>

vector <type> a; - название

vector <int> a; - пустой вектор

vector <int> a(10,2); - в векторе 10 элементов, каждый равен 2

|  |  |
| --- | --- |
| reverse(a.begin(),a.end()) |  |
| a.push\_back(5); | в конце вектора добавится 5 |
| a.size(); | вывод размера вектора |
| a.pop\_back(); | удаляет последний элемент |
| \*a | значение по указателю (адресу элемента)  название массива - а - адрес 1го элемента ->  \*(a+2) третий элемент |
| sort(a,a+n);  sort(a.begin(),a.end()) | сортирует между 2мя указателями  сортировка |
| a.insert(N,M);  a.insert(a.begin()+2,4) | добавляет определенный элемент на определенное место ( n – указатель, m – значение)  Поставит на второе место 4 |
| a.begin()  a.end()  a.rbegin()  a.rend() | Указатель на начало  Указатель на конец (после последнего элемента)  Указатель на первый элемент с конца  Указатель на конец с конца |
| a.erase(указатель) | Указатель элемента который нужно удалить |
| a.clear() | Полностью очищает вектор |
|  |  |

Вывод через цикл: for (int i=1;i<a.size();i++)cout<<a[i]<<” “;

Заполнение с клавиатуры: int n,x;

Vector <int> a;

Cin>>n;

For (int i=0;i<n;i++){

Cin>>x; a.push\_back(x); }

Работа с файлами

Freopen(“input.txt”,”r”,stdin) – считывание

Freopen(“output.txt”,”w”,stdout) – запись

Работа в терминале: ls-l (все существующие файлы); cat output.txt (показать содержимое)

Задача: Даны координаты точек х1,y1…xn,yn. Отсортировать по x, y

Способы решения: 1) Раздельные массивы для х и для у; 2) Массив размерности 2хn;

For (int i=0;i<n;i++)

For (int j=i+1;j<n-1;j++){

If (x[i]>x[j] || x[i]==x[j] && y[i]>y[j]){swap (x[i],x[j]); swap (x[i],x[j]);}}

Pair – структура данных, в которой можно хранить 2 значения

Pair <type1,type2> p;

Pair <p.first, p.second> p;

Pair <int,int> p;

p.first(5); p.second(10); -> p(5,10);

p(“Almat”,18); - p<string,int>

p<int, pair<int,int>>p; p(5,(2,3));

2=p.second.first;

3=p.second.second;

Vector of pairs

Vector <pair<int,int> >a;

Int n,x,y;

Vector <pair<int,int> >a;

Int main(){

Cin>>n;

For (int i=0;i<n;i++){

Cin>>x>>y; cin>>x>>y;

a.push\_back(make\_pair(x,y)); } pair <int,int> p;

p.first=x; p.second=y;}

sort(a.begin(),a.end());

for (int i=0;i<n;i++)cout<<a[i].first<<” “<<a[i].second<<endl;

}

Сортировка строк идет по ASCII кодам

Vector array – массив векторов

Vector <int> a[n];

1 – 256

2 – 1456

3 – 45

4 – 235

5 – 1234

6 – 12

Vector < int> a[n];

A[2] -вектор (3) с элементами (4,5)

N – городов, M – ребер

|  |
| --- |
| #include <iostream> |
| #include <vector> | | |
|  | | |
| using namespace std; | | |
|  | | |
| int main() { | | |
| int n, m, x, y; | | |
| cin >> n >> m; | | |
| vector<int> a[n]; | | |
| for (int i = 0; i < m; i++) { | | |
| cin >> x >> y; | | |
| x--; | | |
| y--; | | |
| a[x].push\_back(y); | | |
| a[y].push\_back(x); | | |
| } | | |
|  | | |
| for (int i = 0; i < n; i++) { | | |
| cout << i + 1 << " - "; | | |
| for (int j = 0; j < a[i].size(); j++) | | |
| cout << a[i][j] + 1 << " "; | | |
| cout << endl; | | |
| } | | |
|  |  |  | | |
| return 0; | | | |
| } | | | |

Vector of vector – двумерный динамический массив

Vector <int> v1,v2,v3;

Vector<vector<int> >v;

V1.push\_back(1); V1.push\_back(2); V1.push\_back(3);

V2.push\_back(4); V2.push\_back(5); V2.push\_back(6);

V3.push\_back(7); V3.push\_back(8); V3.push\_back(9);

V.push\_back(v1); V.push\_back(v2); V.push\_back(v3);

SET – все элементы добавляются и хранятся в отсортированном виде в единственном экземпляре

#include <set>

set <int> a;

a.insert(5); - добавляется, если числа нет

a.size(); - размерность – количество уникальных элементов

a.erase(2); - удаляет элемент из сета

a.find(m); - находит элемент внутри сета, возвращает указатель m если элемент есть, если нет возвращает a.end()

В памяти все элементы хранятся не по порядку, а в виде дерева -> по индексу нельзя обратиться к определенному элементу. Для того чтобы пробегать по сету существуют ИТЕРАТОРЫ – указатель, умеет находить следующий элемент в структуре, хранит адрес элемента

a.begin(); -адрес первого эемента в сете

a.end();

Итератор создается: struct::iterator it;

Set <int>::iterator it;

For (int i:a)cout<<i<<” “; For (it=a.begin();it!=a.end();it++)cout<<\*it<<” “;

Vector <int>::iterator it1;

Set <int>a; cin>>n;

For (int i=0;i<n;i++){

Cin>>x; a.insert(x);}

Cin>>m;

If (a.find(m)==a.end())cout<<”No”; else cout<<”Yes”;

Set<int> a(b.begin(),b.end()); – все элементы вектора добавляются в сет

Set<int> a(b,b+n);

Int n,x;

Cin>>n;

Int a[n];

For (int i=0;i<n;i++)cin>>a[i];

Set<int>b(a,a+n);

Cout>>b.size();

|  |
| --- |
| #include <iostream> |
| #include <set> | |
| #include <vector> | |
|  | |
| using namespace std; | |
|  | |
| int main() { | |
| int n, x; | |
| cin >> n; | |
| int a[n]; | |
| for (int i = 0; i < n; i++) | |
| cin >> a[i]; | |
| // 1 2 1 3 1 | |
| // 1 2 3 | |
| set<int> b(a, a + n); | |
| cout << b.size(); | |
| return 0; | |
| } | |

У любой структуры есть begin, end, rbegin, rend

MAP позволяет использовать в качестве индекса любой тип данных

#include <map>

Map <key, value> a;

A[“Marat”] = 18

Key Value

Map<string,int>a;

a.insert(make\_pair(“Marat”,18));

abc kkk ppp map<string,int>a;

kkk abc llll a[“abc”]=1;

ttt abc ppp a[“kkk”]=1; a[s]++;

Посчитать сколько раз встречается каждый элемент в тексте

Map<string,int>a;

String s;

Int main(){

While (cin>>s){

A[s]++;}

Cout<<a[“abc”];

Map<string,int>::iterator it;

For (it=a.begin();it!=a.end();it++){

Cout<<(\*it).first<<” “<<(\*it).second;}

}

Map: ID,Name,Surname

Чтобы по ID вытаскивать имя -> key=ID; Value(Name,Surname);

Ma0<string,pair<string,string> >a;

String id,name,surname;

While (cin>>id>>n>>s){

A[id]=make\_pair(n,s);

} cout<<a[“18BD04”].first<<” “<<a[“18BD04”].second;

Вывести все значения

Map<string,pair<string,string> >::iterator it;

For (it=a.begin();it!=a.end();it++){ For (auto it:a){

Cout<<(\*it).first<<(\*it).second } cout<<(\*it).first<<(\*it).second }

Using namespace std;

#define fi first

#define s second

Для того, чтобы cout<<(\*it).fi<<(\*it).se;

Дана скобочная последовательность. Проверить на правильность. Есть только ().

Если ( +1, если ) -1. Счетчик всегда должен быть >=. Если в конце не 0, то неверно.

Bool f(string s){

Int cnt=0;

For (int i=0;i<s.size();i++){

If (s[i]==”(”) cnt++; If (s[i]==”)”) cnt--;

If (cnt<0)return false;

If (cnt!=0)return false; return true;}

STACK

FILO – first in last out

Push – добавляет верхний элемент

Pop – удаляет верхний элемент

Top – возвращает значение верхнего элемента

Stack <int> st;

St.push(1); St.push(2); st.push(10); St.push(5);

Cout<<st.top(); - выведет 5

St.pop(); cout<<st.top(): - выведет 10

Встречаютcя ()[]{}. Для каждой закрытой должна быть последняя такая же открытая.

Stack <char> st;

Bool f(string){

For (int i=0;i<s.size();i++){

If (s[i]==”(” || s[i]==”[” || s[i]==”{”) st.push(s[i]);

Else

If (s1.empty())return false;

Char ch=st.top();

If (ch==”(” && s[i]!=”)”) return false;

If (ch==”[” && s[i]!=”]”) return false;

If (ch==”{” && s[i]!=”}”) return false;

St.pop();}

If (!st.empty()) return false;

Return true;}

Int main(){

… if (f(s)) cout<<”Yes”;}

Queue

FIFO - first in first out

Push – добавление в конец

Pop – удаление с начала

Front – значение первого в очереди

Size(); empty(;

Queue<int>q;

q.push(1); q.push(3); q.push(10); q.push(5);

cout<<q.front(); q.pop(); cout<<q.front(); - выведет 1 3

a=1; b=15

Даны a,b. Можно делать либо +1, либо \*2. За минимальное количество шагов дойти от a к b.

Массив distance:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|  | 1 | 2 | 2 | 3 | 3 | 4 | 3 | 4 | 4 | 5 | 4 | 5 | 5 | 6 |

q=1 1+1=2; 1\*2=2;

q=2 2+1=3;2\*2=4;

|  |  |  |  |
| --- | --- | --- | --- |
| 1 шаг | q=1 | 1+1=2 | 1\*2=2 |
| 2 шаг | q=2 | 2+1=3 | 2\*2=4 |
| 3 шаг | q=3  q=4 | 3+1=4  4+1=5 | 3\*2=6  4\*2=8 |
| 4 шаг | q=6  q=5  q=8 | 6+1=7  5+1=6  8+1=9 | 6\*2=12  5\*2=10  8\*2=16 |
| 5 шаг | q=7  q=12  q=10  q=9 | 7+1=8  12+1=13  10+1=11  9+1=10 | 7\*2=14  12\*2=24  11\*2=22  9\*2=18 |
| 6 шаг | q=14  q=13  q=11 | 14+1=15  13+1=14  11+1=12 | 14\*2=28  13\*2=26  11\*2=22 |

Минимальное количество шагов от 1 до 15 – 6.

Queue<int>q;

Int d[100];

Int a,b; cin>>a>>b;

q.push(a);

while (!q.empty()){

int x=q.front();

if (d[x]<=b && d[x+1]==0){d[x+1]=d[x]+1; q.push(x+1);}

if (d[x\*2]<=b && d[x\*2]==0){d[x\*2]=d[x]+1; q.push(x\*2);}

q.pop();}

cout<<d[b];

|  |
| --- |
| #include <iostream> |
| #include <queue> | |
|  | |
| using namespace std; | |
| queue<int> q; | |
| int d[100]; | |
| int a, b; | |
|  | |
| int main() { | |
| cin >> a >> b; | |
| q.push(a); | |
| while (true) { | |
| int k = q.front(); | |
| cout << k << " " << d[k] << endl; | |
| if (d[k + 1] == 0 && k + 1 <= b) { | |
| d[k + 1] = d[k] + 1; | |
| q.push(k + 1); | |
| } | |
| if (d[k \* 2] == 0 && k \* 2 <= b) { | |
| d[k \* 2] = d[k] + 1; | |
| q.push(k \* 2); | |
| } | |
| q.pop(); | |
| if (d[b] > 0) | |
| break; | |
| } | |
| cout << d[b]; | |
| return 0; | |
| } | |

SORT отсортировать числа по возрастанию

Bubble sort

Int a[100]; int n;

For (int i=0;i<n;i++)

For (int j=i+1;j<n;j++)

If (a[i]>a[j])swap(a[i],a[j]);

Отсортировать в возрастающем порядке по количеству делителей

Int f1(int n){ int cnt=0;

For (int i=1;i<=n;i++){

If (n%i==0)cnt++;

Return cnt;}

Bool f(int t, int p){

If (f1(t)<f1(p))return true;

If (f1(t)>f1(p))return false;

Return t<p;}

For (int i=0;i<n;i++)

For (int j=i+1;j<n;j++)

If (f1(a[i],a[j])==false)swap(a[i],a[j]);

Sort(a,a+n); - по стандартному способу

Sort(a,a+n,f); - по правилу функции

#include <vector>

#include <algorithm>

Vector <string>a; string s;

For (int i=0;i<n;i++){

Cin>>s; a.push\_back(s);}

Sort(a.begin();a.end());

Bool f(string s2,string s2){

If (s1.size<s2.size)return true;

If (s1.size>s2.size)return false;

Return s1>s2;}

Sort(a.begin(),a.end(),f);