Desafios de Programação
Estrutura de Dados

Wladimir Araújo Tavares [1]

[1]Universidade Federal do Ceará - Campus de Quixadá

23 de março de 2017

STL Vector

Função	Descrição
resize(*n*, *val*)	redimensiona o vetor para ter n elementos inicializados com o valor val
push_back(*val*)	adiciona o elemento no final do vetor
pop_back()	remove o último elemento do vetor
insert(*it*, *val*)	insere o elemento val antes da posição especificada por it
sort(first,last)	ordena os elementos no intervalo [first,last) em ordem crescente

Exemplo: vector

```
#include <vector>
#include <algorithm>
#include <iostream>
#define all(v) v.begin(),v.end()
using namespace std;
typedef vector <int> vi;
int main(){
  vi v1;
  v1.resize(3, 0);//Aloca três posições e inicializa com zero
  v1[0] = 5; v1[1] = 4; v1[2]=6;
  v1.push_back(9);
  vi::iterator it = v1.begin();
  v1.insert(it,7);// insere no começo
  for(int i = 0; i < v1.size(); i++) cout << v1[i] << endl;
  int maior = *max_element(all(v1));
  cout << "maior: " << maior << endl;
  vi v2(v1.begin(), v1.begin()+3);
  for(int i = 0; i < v2.size(); i++) cout << v2[i] << endl;
  vi v3(v1); // copy v1
  cout << "sort vector" << endl;
  sort(all(v3));
  reverse(all(v3));
  for(int i = 0; i < v3.size(); i++) cout << v3[i] << endl;
}
```

STL queue

Função	Descrição
push(*val*)	insere o elemento val na fila
back()	retorna uma referência para o elemento mais novo na fila
pop()	remove o elemento mais antigo na fila
front()	retorna uma referência para o elemento mais antigo na fila

Exemplo: queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

int main()
{
   queue <int> fila;
   fila.push(2); fila.push(3);
   fila = queue <int> ();
   fila.push(2); fila.push(3); fila.push(4);
   cout << fila.size() << endl;
   cout << fila.back() << endl;
   while( !fila.empty() ){
      cout << fila.front() << " ";
      fila.pop();
   }
}
```

STL stack

Função	Descrição
push(*val*)	insere o elemento val no topo da pilha
top()	retorna uma referência para o elemento no topo da pilha
pop()	remove o elemento do topo da pilha

Exemplo: stack

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <stack>
using namespace std;
int main()
{
   stack <int> p;
   p.push(4);
   p.push(5);
   p.push(6);
   while( !p.empty() )
   {
      cout << p.top() << " ";
      p.pop();
   }


}
```

STL set

Função	Descrição
insert(val)	adiciona o elemento val, mas não permite elementos duplicados
erase(val)	remove o elemento val
erase(position)	remove o elemento apontado pelo iterator position
erase(first,last)	remove todos os elementos entre first e last
count(val)	o número de vezes que val aparece no set
find(val)	se existe o elemento val a função devolve seu iterador; caso contrário devolve end()

Exemplo Set

```
#include <iostream>
#include <set>
using namespace std;
int main(){
  set<int> myset;
  set<int>::iterator itlow,itup,it;
  for(int i = 1; i <= 10; i++) myset.insert(i);
  myset.insert(8); // nao insere
  itlow=myset.lower_bound (3);
  itup=myset.upper_bound (6);
  myset.erase(itlow,itup); // 1 2 7 8 9 10
  it = myset.find(7); // Complexidade logaritmica
  myset.erase(it); // 1 2 8 9 10
  cout << "size of set:" << myset.size() << endl;
  if( myset.count(7) == 0) cout << "7 is not a element of myset"
  for (it=myset.begin(); it!=myset.end(); ++it) //Percorre em ord
     cout << ' ' << *it;
}
```

STL map

Função	Descrição
insert(val)	adiciona o elemento val
erase(val)	remove o elemento val
erase(position)	remove o elemento apontado pelo iterator position
erase(first,last)	remove todos os elementos entre first e last
find(val)	se existe o elemento val então a função devolve seu it; caso contrário devolve map::end()

Exemplo map

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <map>
using namespace std;
int main()
{
   char poema [] = "Sou chama sem luz jardim \
  sem luar luar sem amor";
   map <char, int> mapa;
   map<char,int>::iterator it;
   for(int i = 0; poema[i] != '\0'; i++)
   {
      char c = poema[i];
      mapa[c]++;
   }

   for(it = mapa.begin(); it != mapa.end(); it++)
   {
      if( it->second > 0 )
      printf("mapa[%c] = %d\n", it->first, it->second );
   }
}
```

STL priority_queue

Função	Descrição	Complexidade
empty()	verifica se a fila de prioridade está vazia	O(1)
size()	devolve o número de elementos na estrutura	O(1)
push()	insere um novo elemento na fila de prioridade	O(lg n)
pop()	remove o elemento do topo da fila de prioridade	O(lg n)
top()	devolve o elemento do topo da fila de prioridade	O(1)

Exemplo 1: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
   //fila de prioridade mínima
   priority_queue <int, vector <int>, greater <int> > pq;
   pq.push(30); pq.push(20);
   pq.push(25); pq.push(40);
   while( !pq.empty() )
   {
      cout << pq.top() << " ";
      pq.pop();
   }
   cout << endl;
   //20 25 30 40
}
```

Exemplo 2: priority_queue

```cpp
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
   //fila de prioridade mínima
   priority_queue <int, vector <int>, greater <int> > pq;
   pq.push(30); pq.push(20);
   pq.push(25); pq.push(40);
   while( !pq.empty() )
   {
      cout << pq.top() << " ";
      pq.pop();
   }
   cout << endl;
   //20 25 30 40
}
```

Exemplo 3: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
   //fila de prioridade máxima
   priority_queue <int, vector <int>, less <int> > pq;
   pq.push(30); pq.push(20);
   pq.push(25); pq.push(40);
   while( !pq.empty() )
   {
      cout << pq.top() << " ";
      pq.pop();
   }
   cout << endl;
   //40 30 25 20
}
```

Exemplo 4: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
typedef bool (*comp)(int,int);
bool compare(int a, int b)
{
   return (a<b);
}
int main()
{
 int v[] = {10,60,50,20};
 priority_queue <int> pq1(v,v+4); //default less<int>
 priority_queue <int, vector <int>, comp> pq2(compare);
 pq2.push(10); pq2.push(60);
 pq2.push(50); pq2.push(20);
 while( !pq2.empty() )
 {
   cout << pq2.top() << " ";
   pq2.pop();
 }
 cout << endl;
}
```

Exemplo priority_queue

```
#include <iostream>
#include <queue>
using namespace std;

class Human {

    public:
        string name;
        int age;
        Human(string name, int age);
};
Human::Human(string name, int age) : name(name), age(age) {}
bool operator<(Human a, Human b) {return (a.age < b.age);}
int main() {

    Human p1("Child",5);
    Human p2("Grandfather",70);
    priority_queue<Human> Q;
    Q.push(p1);
    Q.push(p2);
}
```

Exemplo UnionFind

```
#include <iostream>
#include <vector>
#include <stdio.h>
using namespace std;
class UnionFind {
 private:
  vector <int> p, rank, setSize;
  int numSets;
 public:
  UnionFind(int N):numSets(N){
   rank.resize(N,0); p.resize(N,0);setSize.resize(N,1);
   for(int i=0;i<N;i++) p[i]=i;
  }
  int findSet(int i) {
    return (p[i]==i)? i : (p[i] = findSet(p[i]));
  }
  bool isSameSet(int i, int j){
    return findSet(i) == findSet(j);
  }
```

Exemplo UnionFind

```
void unionSet(int i, int j){
    if( !isSameSet(i,j)){
    numSets--;
    int x = findSet(i), y = findSet(j);
    if(rank[x]>rank[y]){ p[y]=x; setSize[x] += setSize[y]; }
    else{
     p[x]=y; setSize[y] += setSize[x];
     if(rank[x]==rank[y]) rank[y]++; }
    }
  }
  int numDisjointSets(){ return numSets; }
  int sizeOfSet(int i){ return setSize[findSet(i)];}
};
```

Exemplo UnionFind

```
int main(){
 UnionFind Set(10);
 Set.unionSet(5,6);
 Set.unionSet(6,7);
 cout << Set.numDisjointSets() << endl;
 Set.unionSet(0,1);
 cout << Set.numDisjointSets() << endl;
 Set.unionSet(5,8);
 cout << Set.numDisjointSets() << endl;
 cout << Set.sizeOfSet(5) << endl;
}
```

Exemplo UnionFind

```
class SegmentTree {
 private:
  vector <int> st,A;
  int n;
  int left(int p) { return p << 1;}
  int right(int p){ return (p << 1) + 1; }

  void build(int p, int L, int R){
   if(L==R){
    st[p]=A[L];
   }
   else{
    build(left(p), L, (L+R)/2);
    build(right(p), (L+R)/2+1, R);
    int p1 = st[left(p)];
    int p2 = st[right(p)];
    st[p] = p1+p2;
   }
  }
```