Programação

IFMG CODAAUT

Prof. Marco Antonio M. Carvalho

UFOP
Universidade Federal de Ouro Preto

PROEX
Pró-reitoria de extensão da UFOP

decom
departamento de computação

INSTITUTO FEDERAL
MINAS GERAIS

Lembretes

- **Lista de discussão**
  - Endereço:
    - programaacao@googlegroups.com
  - Solicitem acesso:
    - http://groups.google.com/group/programaacao
- **Página com material dos treinamentos**
  - http://www.decom.ufop.br/marco/extensao/obi/
- **Repositório online de problemas das edições passadas da OBI**
  - http://br.spoj.com/problems/obi/sort=-7
- **Moodle**
  - http://programaacao.net.br/login/index.php

Avisos

- Verifica calendário de final de ano no IFMG.

Na aula de hoje

- *map*
- *multimap*
- Problemas Selecionados
- Um Problema de Lógica

map

map

- Mapas são contêineres associativos que armazenam **pares** de elementos relacionados
  - Diz-se que uma *chave* é mapeada a um *valor*;
  - As chaves são únicas;
  - A chave e o valor podem ter diferentes tipos.
- Internamente, mapas são implementados como árvores binárias de busca
  - Elementos ordenados de acordo com a chave;
  - Busca em tempo $O(\log n)$.

map

- Mapas implementam o operador **[]**
  - O que permite o acesso direto a elementos.
- Como os mapas armazenam pares de elementos, os iteradores possuem uma característica extra
  - *it->first* ou *(*it).first* acessa a **chave** do elemento referenciado pelo iterador **it**;
  - *it->second* ou *(*it).second* acessa o **valor** do elemento referenciado pelo iterador **it**.

map

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  map<char,int> first;
  //atribuição direta
  first['a']=10;
  first['b']=30;
  first['c']=50;
  first.insert(pair<char,int>('d',70));
```

map

```
  //localiza a ocorrência do elemento
   map<char,int>::iterator it = first.find('a');
   //imprime o par
   cout << it->first << '\t' << it->second << '\n';
  //imprime os pares do mapa
for ( map<char, int>::iterator it = first.begin(); it != first.end(); ++it )
        cout << it->first << '\t' << it->second << '\n';

    return 0;
}
```

map

- Para usar o método insert, precisamos fazer o cast para o tipo *pair*

pair<*tipochave*, *tipovalor*>(*chave*, *valor*);

map

- **Atenção**
  - Acessar uma chave através do operador **[ ]** insere esta chave no mapa, caso ela não esteja presente;
  - Assim, o operador **[ ]** não é adequado para verificar se uma chave está presente no mapa, é preciso utilizar o método *find*.
- O contêiner *map* possui os mesmos métodos *find*(), *count*(), *swap*() e *clear*().

multimap

multimap

- Multimapas são idênticos aos mapas
  - Entretanto, permitem a existência de chaves duplicadas.

multimap

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  multimap<char,int> first;
  //insere os pares
  first.insert(pair<char,int>('a',10));
  first.insert(pair<char,int>('b',15));
  first.insert(pair<char,int>('b',20));
  first['c']=25;
```

multimap

```
 //localiza a ocorrência do elemento
 multimap<char,int>::iterator it = first.find('a');
 //remove o elemento
 first.erase(it);
//imprime os pares do multimapa
for ( multimap<char, int>::iterator it = first.begin(); it != first.end(); ++it )
    cout << it->first << '\t' << it->second << '\n';

 return 0;
}
```

multimap

- O contêiner *multimap* possui os mesmos métodos *find*(), *count*(), *swap*() e *clear*();
- Para determinar os valores relativos a chaves repetidas utilizamos o método *equal_range*()
  - Retorna um par de iteradores, indicando um intervalo de chaves iguais.

multimap

```
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  multimap<char,int> first;
  //insere os pares
  first.insert(pair<char,int>('a',10));
  first.insert(pair<char,int>('b',15));
  first.insert(pair<char,int>('b',20));
  first.insert(pair<char,int>('b',30));
```

multimap

```
 //par de iteradores, delimitam o intervalo de chaves idênticas
 pair<multimap<char,int>::iterator, multimap<char,int>::iterator> ret;
 //determina o intervalo de chaves iguais a 'b'
 ret = first.equal_range('b');
//imprime o intervalo de chaves iguais
for ( multimap<char, int>::iterator it = ret.first; it != ret.second; ++it )
    cout << it->first << '\t' << it->second << '\n';

 return 0;
}
```

Problemas Selecionados

Problemas Selecionados

- http://www.urionlinejudge.com.br/judge/en/problems/view/1281
- http://www.urionlinejudge.com.br/judge/en/problems/view/1260
- http://www.urionlinejudge.com.br/judge/en/problems/view/1251
- http://br.spoj.com/problems/CALCULAD/
- http://br.spoj.com/problems/TIMES1/
- http://www.urionlinejudge.com.br/judge/en/problems/view/1167
- http://www.urionlinejudge.com.br/judge/en/problems/view/1258
- http://www.urionlinejudge.com.br/judge/en/problems/view/1449

Um Problema de Lógica

Um Problema de Lógica

- Qual é o próximo número da sequência abaixo?

2, 10, 12, 16, 17, 18, 19 ...

Perguntas?