STL - Biblioteca Padrão do C++

Sérgio Daniel C. Canuto

12 de março de 2011

Visão Geral

- Motivação e definição;
- E/S pelo console;
- Estruturas de dados;
- Referências;
- Exercícios.

Motivação e definição de STL

- STL (Standard Template Library) é a biblioteca padrão do C++ que implementa estruturas de dados e algoritmos básicos da ciência da computação.
- Motivação:
  1. Padronização (outras pessoas podem rapidamente entender o código);
  2. Agilidade na implementação (várias estruturas de dados e algoritmos já implementados);
  3. Confiabilidade (estruturas e algoritmos estáveis);
  4. Documentação (a biblioteca é muito bem documentada).

Instrução de compilação

- $ g++ -o programa programa.cpp
- Manual do compilador para informações adicionais:
- $ man g++

Visão Geral

- Estruturas de dados mais utilizadas:
  - vector;
  - bitset;
  - list;
  - set;
  - map.

Vector

- O vetor é bom para:
  - Acessar elementos pelo índice (tempo constante);
  - Percorrer todos elementos (tempo linear);
  - Adicionar e remover elementos do fim do vetor.

0	1	2	3	4
200	130	150	500	900

Vector - Principais Métodos

Construtores

- Criação de Vector de inteiros inicialmente vazio:
  $vector < int > v1;$
- Criação de Vector com 5 inteiros, inicializando com o valor 0:
  $vector < int > v2(5, 0);$
- Criação com inicializaçao a partir de outro vector:
  $vector < int > v3(v1);$
- Criação com inicialização por um intervalo:
  $vector < int > \ v4(v1.begin(), v1.end());$

Vector - Principais Métodos

Criação de Vetor - Exemplo 1

```
1   #include <iostream>
2   #include <vector>    // Para containers vector
3   using namespace std;
4   int main()
5   {
6    vector<int> v1(3,9);
7    vector<int>::iterator it; // Iterador
8    cout << "Vetor v1: ";
9    for (int i = 0;i < 3;++i)
10     cout << v1[i] << " "; // Indexacao semelhante a de um array
11   cout << "  Vetor v1: ";
12   for (it = v1.begin();it != v1.end();++it)
13     cout << *it << " ";
14   cout << endl;
15   return 0;
16  }
```

Vector - Principais Métodos

Criação de Vetor - Exemplo 2

```
1   #include <iostream>
2   #include <vector>    // Para containers vector
3   using namespace std;
4   int main()
5   {
6    vector<int> v1(3);
7    vector<int>::iterator it;  // Iterador
8    // Atribui valores com a notacao de array
9    v1[0] = 1;v1[1] = 2;v1[2] = 3;
10   cout << "Vetor v1: ";
11   for (it = v1.begin();it != v1.end();++it)
12     cout << *it << " ";
13   vector<int> v2(v1);  // Cria o vetor v2
14   cout << "  Vetor v2: ";
15   for (it = v2.begin();it != v2.end();++it)
16     cout << *it << " ";
17   cout << endl;
18   return 0;
19  }
```

Vector - Principais Métodos

Criação de Vetor - Exemplo 3

```
1   #include <iostream>
2   #include <vector>    // Para containers vector
3   using namespace std;
4   int main()
5   {
6    vector<int> v1(3);
7    v1[0] = 1;v1[1] = 2;v1[2] = 3;
8    cout << "Vetor v1: ";
9    for (int i = 0;i < 3;++i)
10     cout << v1[i] << " ";
11   vector<int> v2(v1.begin(),v1.end());  // Cria v2
12   cout << "  Vetor v2: ";
13   for (int i = 0;i < 3;++i)
14     cout << v2[i] << " ";
15   cout << endl;
16   return 0;
17  }
```

Vector - Principais Métodos

Atribuição, Push Back e Pop Back

- Atribuição de elemento: $v[1] = 3;$ ou $*it = 3;$, onde it é um iterator que aponta para o elemento 1;
- Atribuição de vetor: $v1 = v2;$ – todos os elementos de v2 são copiados em v1;
- Inserção ao final: $v1.push\_back(5);$
- remoção ao final: $int\ num = v1.pop\_back();$

Vector - Principais Métodos

Push Back() e PopBack() - Exemplo

```
1   // vector::push_back
2   #include <iostream>
3   #include <vector>
4   using namespace std;
5   int main ()
6   {
7     vector<int> myvector;
8     int myint;
9           cout << "espaco fisico ocupado " << (int) myvector.capacity() << endl;
10    cout << "Digite alguns numeros inteiros ( 0  para terminar end):\n";
11    do {
12          cin >> myint;
13          myvector.push_back (myint);
14          cout << "espaco fisico ocupado " << (int) myvector.capacity() << endl;
15    } while (myint);
16    cout << "myvector armazena " << (int) myvector.size() << " numeros.\n";
17          int valorInicio=myvector.front();
18          int valorFim=myvector.back();
19          myvector.pop_back();  // Eliminando o último elemento inserido
20          cout << "myvector armazena " << (int) myvector.size() << " numeros.\n";
21          cout << "valor removido " << valorFim << endl;
22          cout << "valor no inicio " << valorInicio << endl;
23          cout << "espaco fisico ocupado " << (int) myvector.capacity() << endl;
24    return 0;
25  }
```

Vector - Principais Métodos

Swap, e Reserva de Espaço

- Troca de conteúdo entre dois objetos Vector: $v1.swap(v2)$;
- Reserva de memória para o vector: $v1.reserve(50)$. Aloca, se necessário, espaço para conter 50 elementos.

Vector - Principais Métodos

Swap - Exemplo

```
1   #include <iostream>
2   #include <time.h>
3   #include <stdlib.h>
4   #include <vector>  // Para containers vector
5   using namespace std;
6   int main(){
7    vector<int> v1,v2;
8           v1.reserve(10);
9           v2.reserve(10);
10          srand ( time(NULL) );
11          int i;
12          for(i=0;i<10;i++){ v1[i]=rand()%10; v2[i]=v1[i]+1;}
13          cout << "Conteudo dos dois vetores" << endl<< "vetor v1: " ;
14          for(int i=0;i<10;i++) cout << v1[i] <<" ";
15          cout << endl<< "vetor v2: " ;
16          for(int i=0;i<10;i++) cout << v2[i] <<" ";
17      v1.swap(v2);
18          cout<< endl << "Deopois da troca (swap)" << endl << "vetor v1: ";
19      for (int i = 0;i < 10;++i) cout << v1[i] << " ";
20          cout << endl << "vetor v2: ";
21          for (int i = 0;i < 10;i++) cout << v2[i] << " ";
22      cout << endl;
23    return 0;
24  }
25
```

Vector - Ordenação

```
1   #include <iostream>
2   #include <time.h>
3   #include <stdlib.h>
4   #include <vector>  // Para containers vector
5   #include <algorithm>
6   using namespace std;
7   int main(){
8           vector<int> v1(10);
9           srand ( time(NULL) );
10          int i;
11          for(i=0;i<10;i++){ v1[i]=rand()%100;}
12          cout << "vetor v1 antes da ordenaao:" << endl ;
13          for(int i=0;i<10;i++) cout << v1[i] <<" ";
14          cout << endl;
15          sort(v1.begin(), v1.end()); //Ordenacao em ordem crescente
16          cout << "vetor v1 depois da ordenaao em ordem crescente:" << endl ;
17          for(int i=0;i<10;i++) cout << v1[i] <<" ";
18          cout << endl;
19          sort(v1.begin(), v1.end(),greater<int>()); //Ordenacao em ordem decrescente
20          cout << "vetor v1 depois da ordenaao em ordem decrescente:" << endl ;
21          for(int i=0;i<10;i++) cout << v1[i] <<" ";
22          cout << endl;
23
24                  return 0;
25  }
```

Vector - Matrix

```
1   #include <iostream>
2   #include <vector>
3   #include <fstream>
4   using namespace std;
5   int main ()
6   {
7     vector< vector< int > > matrix;
8     int linhas=5, colunas=6;
9     matrix.resize(linhas);
10    for(int i=0; i<linhas; i++){
11      matrix[i].resize(colunas);
12    }
13    for(int i=0; i<linhas; i++){
14      for(int j=0; j<colunas; j++){
15        matrix[i][j]=i+j;
16      }
17    }
18    ofstream myfile;//imprime a matriz
19    myfile.open ("matrix.txt", ios::app) ;
20    for(int i=0; i<linhas; i++){
21      for(int j=0; j<colunas; j++){
22        myfile << matrix[i][j]<< " ";
23      }
24      myfile<<endl;
25    }
26    myfile.close();
27    return 0;
28  }
```

Bitset

- O bitset é útil para:
  - Manipular bits rapidamente;

Bitset

```
1   #include <iostream>
2   #include <string>
3   #include <bitset>
4   using namespace std;
5
6   int main ()
7   {
8     bitset<200> myset (2147483647); //inicializa com um numero decimal
9     cout << "myset has " << int(myset.count()) << " ones ";
10    cout << "and " << int(myset.size()-myset.count()) << " zeros.\n";
11
12
13    bitset<4> mybits (string("0001")); //inicializa com uma string
14    if (mybits.any())
15      cout <<"Ao menos um bit eh 1!"<<endl;
16
17    mybits[1]=1;              // 0011
18    mybits[2]=mybits[1];      // 0111
19
20    cout << mybits << endl;
21
22
23    return 0;
24  }
```

List

- A lista é boa para:
  - Inserir e remover elementos em qualquer lugar (tempo constante);

List

```
1   #include <iostream>
2   #include <list>
3   using namespace std;
4   int main ()
5   {
6     list<int> mylist;
7     mylist.push_back (150);
8     mylist.push_back (130);
9     mylist.push_front (200);
10    mylist.push_front (300);
11    list<int>::iterator it; //ponteiro para elemento da lista
12    cout << "mylist contains:";
13    for (it=mylist.begin(); it!=mylist.end(); ++it)
14      cout << " " << *it;
15    cout << endl;
16    //300 200 150 130
17    it=mylist.begin();
18    while (it!=mylist.end()){
19      if(*it>150)
20        it = mylist.erase(it);
21      else
22        it++;
23    }
24    //150 130
25    mylist.sort();  //130 150
26    return 0;
27  }
```

List

```
1   // list::insert
2   #include <iostream>
3   #include <list>
4   using namespace std;
5   int main ()
6   {
7     list<int> mylist;
8     mylist.push_back (150);
9     mylist.push_back (130);
10    mylist.push_front (200);
11    mylist.push_front (300);
12    //300 200 150 130
13
14    list<int>::iterator it; //ponteiro para elemento da lista
15
16    for (it=mylist.begin(); it!=mylist.end(); it++){
17      if(*it==150){
18        mylist.insert(it, 149);
19      }
20    }
21
22    for (it=mylist.begin(); it!=mylist.end(); it++)
23      cout <<*it <<" ";
24    cout<<endl;
25    //300 200 149 150 130
26
27  return 0;
28  }
```

Set

- O set é bom para:
  - Procurar ou remover elementos pelo valor (tempo logarítmico);

Set

```
1   // set::insert
2   #include <iostream>
3   #include <set>
4   using namespace std;
5
6   int main ()
7   {
8     set<int> myset;
9     set<int>::iterator it;
10
11    // set some initial values:
12    for (int i=1; i<=5; i++) myset.insert(i*10);    // set: 10 20 30 40 50
13
14    cout << "myset contains:";
15    for (it=myset.begin(); it!=myset.end(); it++)
16      cout << " " << *it;
17    cout << endl;
18
19    if(myset.find(10)!=myset.end()){
20      cout<< "Tem o elemento 10!";
21      it=myset.find(10);
22      myset.erase (it);
23    }
24
25    return 0;
26  }
```

Map

- Tem como objetivo associar uma chave a um valor mapeado;
- O map é bom para:
  - Procurar ou remover elementos pela chave (tempo logarítmico);

Nome e CPF:

Chave	Valor
João	030.598.821-10
Bob	042.738.422-30
Carl	077.792.891-40

Caractere e valor ASCII:

Chave	Valor
A	64
B	65
C	66

Map

```
1   #include <iostream>
2   #include <map>
3   using namespace std;
4   int main ()
5   {
6     map<char,int> mymap;
7     map<char,int>::iterator it; //ponteiro para 1 elemento do mapeamento
8     pair<map<char,int>::iterator,bool> ret;
9
10    mymap['a']=100; //primeira forma de inserir
11    mymap.insert ( pair<char,int>('z',200) ); //outra forma de inserir
12    ret=mymap.insert (pair<char,int>('z',500) );
13    if (ret.second==false)
14    {
15      cout << "element 'z' already existed";
16      cout << " with a value of " << ret.first->second << endl;
17    }
18    // showing contents:
19    cout << "mymap contains:\n";
20    for ( it=mymap.begin() ; it != mymap.end(); it++ )
21      cout << (*it).first << " => " << (*it).second << endl;
22    if(mymap.find('a')!=mymap.end()){
23      cout<< "existe um mapeamento para a"<<endl;
24      cout <<"valor: " << mymap['a'];
25    }
26    return 0;
27  }
```

Outras Estruturas

- queue;
- priority_queue;
- stack;
- multimap;
- multiset;
- hash_map;
- hash_set.

Referências

- http://www.sgi.com/tech/stl/table_of_contents.html
- http://www.cplusplus.com/reference/stl/

Exercícios

- http://br.spoj.pl/problems/PARIDADE/
- http://br.spoj.pl/problems/POPULAR/
- http://br.spoj.pl/problems/TROCCARD/
- http://br.spoj.pl/problems/ORKUT/