面向对象程序语言C++ 标准库

周雅倩
zhouyaqian@fudan.edu.cn
2016/3/17

主要内容

- Hello World程序
- 使用字符串
- 使用批量数据
- 组织程序和数据
- 使用序列式容器
- 使用库算法
- 使用关联式容器

使用序列式容器并分析字符串

- 标准库不仅提供了有用的数据结构和函数，它还反映了一种一致性的体系结构：
  只要我们知道如何使用一个容器，我么自己就会明白如何使用所有的库容器。

把学生分类

- 问题：希望知道那些学生没通过考试。
- 检测函数：

```
bool fgrade(const Student_info &s)
{
   return grade(s) < 60 ;
}
```


解决问题-最直接方法

- 两个vector：检测每个学生的记录，一个vector保存及格的学生记录，另一个保存不及格的学生记录。

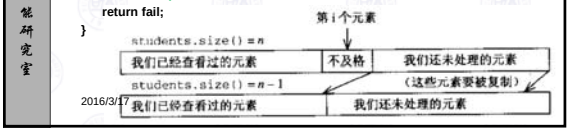
```
vector<Student_info> extract_fails(vector<Student_info> &students)
{
    vector<Student_info> pass, fail;//两个vector
    for(vector<Student_info>::size_type i=0;i!=students.size();++i){
        if(fgrade(students[i])){
            fail.push_back(students[i]);
        }else{
            pass.push_back(students[i]);
        }
    }
    students=pass;
    return fail;
}
//缺点：需要足够的内存来保存每个学生的两份副本。
```


解决问题-就地删除元素

```
vector<Student_info> extract_fails(vector<Student_info> &students)
{
    vector<Student_info> fail;//一个vector
    for(vector<Student_info>::size_type i=0;i!=students.size();){
        if(fgrade(students[i])){
            fail.push_back(students[i]);
            students.erase(students.begin()+i);
        }else{
            ++i;
        }
    }
    //students=pass;
    return fail;
}
```

循环中会改变

删除第i个元素



顺序访问和随机访问

```
vector<Student_info> extract_fails(vector<Student_info> &students)
{
    vector<Student_info> fail;//一个vector
    for(vector<Student_info>::size_type i=0;i!=students.size();){
        if(fgrade(students[i])){
            fail.push_back(students[i]);
            students.erase(students.begin()+i);
        }else{
            ++i;
        }
    }
    //students=pass;
    return fail;
}
```

顺序访问

迭代器

- 一个迭代器是一个值。它能够：
  - 标识一个容器和容器中的一个元素
  - 允许检测元素中保存的值
  - 提供在容器元素之间移动的操作
  - 使用容器可用有效处理的方式来约束可用的操作

迭代器-顺序访问

```
for(vector<Student_info>::size_type i=0;i!=students.size();++i){
        cout <<students[i].name <<endl;
}
```

```
for(vector<Student_info>::const_iterator
   iter=students.begin();iter!=students.end();++iter){
        cout << (*iter).name <<endl;
   }
```


迭代器类型

- 每个标准库容器，比如vector，都定义了两个相关的迭代器类型：

只读取访问时

container-type::const_iterator

container-type::iterator

需要改变保存在容器中的值时

```
for(vector<Student_info>::const_iterator iter=student.begin();
```

这里有 iterator->const_iterator 的自动转换

迭代器操作

```
for(vector<Student_info>::const_iterator
   iter=students.begin();iter!=students.end();++iter){
        cout << (*iter).name <<endl;
   }
```

或者：

```
cout << iter-> name <<endl;
```


students.erase(students.begin()+i)的含义

- students.erase(students.begin()+i)

对支持随机访问的容器有意义

对只支持顺序访问的容器无意义

使用迭代器取代索引

```
vector<Student_info> extract_fails(vector<Student_info> & students)
{
    vector<Student_info> fail;

    for(std::vector<Student_info>::iterator
    iter=students.begin();iter!=students.end();){
        if(fgrade(*iter)){
            fail.push_back(*iter);
            iter=students.erase(iter);
        }else{
            ++iter;
        }
    }
    return fail;
}
```

erase函数返回指向被删除元素之后跟着的元素的迭代器

重新设计数据结构以获得更好的性能

- vector的问题：插入和删除费时
- 解决：使用list

list

```
#include <list>
list<Student_info> extract_fails(list<Student_info> & students)
{
    list<Student_info> fail;

    for(std::list<Student_info>::iterator
    iter=students.begin();iter!=students.end();){
        if(fgrade(*iter)){
            fail.push_back(*iter);
            iter=students.erase(iter);
        }else{
            ++iter;
        }
    }
    return fail;
}
```


list Vs. vector

- list顺序访问比vector稍慢些
- 对于大型数据，list删除和插入要比vector快得多
- vector支持索引，而list不支持
  - vector[i] //right
  - list[i] //error!

sort函数

- 标准库中的sort函数不支持list元素排序
- 由于VC6.0的编译器不支持以下的函数：list的函数sort(bool cmp)
- 在struct Student_info中重载操作符<：

```
struct Student_info{
std::string name;
double midterm, final;
std::vector<double> homework;
bool operator<( const Student_info &y)
   { return name < y.name; }
};
```

- 对学生信息进行排序的时候调用students.sort() 。

学生信息排序问题

- 由于VC6.0的编译器不支持以下的函数：list的函数sort(bool cmp)
- 在struct Student_info中重载操作符<：

```
struct Student_info{
std::string name;
double midterm, final;
std::vector<double> homework;
bool operator<( const Student_info &y)
   { return name < y.name; }
};
```

- 对学生信息进行排序的时候调用students.sort() 。

其它问题

- 1、函数max可以用_cpp_max替换
- 2、list<Student_info>::iterator等类型定义可能无法通过编译，可在前面加std，即std::list<Student_info>::iterator

复旦大学媒体计算和网络智能研究室

主要内容

- 容器
  - vector
  - string
  - list
- 迭代器
  - iterator
- 库算法
- 编写范型函数