第十六章 STL和向量

1.STL

超快入门C++标准模版库，算法竞赛必看
https://zhuanlan.zhihu.com/p/344558356

- 什么是标准模版库STL（Standard Template Library），是 C++ 标准库的一部分，不需要单独安装，只需要 #include 头文件。其包含有大量的模板类和模板函数，是C++提供的一个基础模板的集合，用于完成诸如输入/输出、数学计算等功能。
  C++ 对模板（Template）支持得很好，STL 就是借助模板把常用的数据结构及其算法都实现了一遍，并且做到了数据结构和算法的分离。
  从根本上说，STL是一些容器、算法和其他一些组件的集合。
  注意，这里提到的容器，本质上就是封装有数据结构的模板类，例如list、vector、stack 等，有关这些容器的具体用法，后续章节会做详细介绍。
  C++中的模板，就好比英语作文的模板，只换主题，不换句式和结构。对应到C++模板，就是只换类型，不换方法。
  在各种容器中，最常做的操作无疑是遍历容器中存储的元素，而实现此操作，多数况会选用“送代器 （iterator）来实现。 那么，迭代器到底是什么呢?
  单来讲，送代器和 C++ 的指针非常类似， 它可以是需要的任意类型，通过送代器可以指向容器中的某个元素，如果需要，还可以对该元素进行读/写操作。 注意：容器适配器 stack和queue没有送代器，它们包含有一些成员函数，可以用来对元素进行访问。
- STL有什么优势?
  STL封装了很多实用的容器，省时省力，能够让你将更多心思放到解决问题的步骤上，而非费力去实现数据结构诸

多细节上，像极了用python时候的酣畅淋漓。

2. STL向量vector

2.1 向量定义和声明

向量（vector）是c++标准库提供的一个可变长数组类型， 属于容器，它可以像数组一样进行数据的存储和访问。vector 会根据需要自动扩展其自身的容量来容纳更多的数据 普通数组在定义初始大小后，其容量不可变；向量在定义初始大小后，仍可根据需要扩充容量。

- 向量的声明方法
  声明格式：vector<数据类型> 向量名称
  相关方法：
  vector<数据类型>v; //声明空向量
  vector<数据类型> v(n); //声明长度为n、初始化为的向量
  vector<数据类型> v(n,x); //声明长度为n、初始化为x的向量
  vector< vector<数据类型> > //声明二维向量 注：声明二维向量时最外的<>要有空格否则在较旧的编译器下可能报错!

- 向量的声明

1. 导入关于向量的头文件； `#include<vector>`
2. 定义int类型的类型v；后续均以int类型举例。 `vector<int> v`

2.2 向量的常用方法

begin()	返回指向容器中第一个元素的迭代器
end()	返回指向容器最后一个元素所在位置的下一个位置的迭代器
rbegin()	返回指向最后一个元素的迭代器
rend()	返回指向第一个元素所在位置的上一个位置的迭代器
front()	返回第一个元素的引用
back()	返回最后一个元素的引用
push_back()	在序列的尾部添加一个元素
pop_back()	移出序列尾部的元素
emplace()	C++ 11 标准新增加的成员函数，在指定的位置直接生成一个元素
emplace_back()	C++ 11 标准新增加的成员函数，在序列尾部生成一个元素

size()	返回实际元素个数
empty()	判断容器中是否有元素，若无元素，则返回 true；反之，返回 false
insert()	在指定的位置插入一个或多个元素
erase()	移出一个元素或一段元素
clear()	移出所有的元素，容器大小变为 0
swap()	交换两个容器的所有元素

2.2 向量的相关方法

向量有两种访问方法，可以像普通数组一样使用索引访问，也可以使用迭代器访问。

1. 利用索引进行访问并且修改，例如修改索引为1的元素：

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

int main(){
    vector<int> vi;
    vi.push_back(1);
    vi.push_back(2);
    cout<<vi[1]<<endl;
    vi[1] = 3;
    cout << vi[1];
    return 0;
}
```

2. 利用迭代器进行防卫并修改，例如修改第三个元素

```
cout << *(v.begin() + 2) <<endl;
*(begin() + 2) = 1;
cout << *(v.begin() + 2) <<endl;
```

3. push_back():
   `v.push_back(0)` // 在末尾添加元素0
4. emplace_back():
   `v.emplace_back(0)` //在末尾添加元素0
5. emplace():
   `v.emplace(v.begin(), 0);` //在首位置添加元素0
6. erase():
   erase(iterator pos) // 删除迭代器只想pos处的元素
   erase(iteraotr begin, iterator end) //删除迭代器指向[begin, end)的元素 例如：

```
v.erase(v.begin()); // 删除第一个元素
v.erase(v.begin(), v.end()); // 删除v的所有元素
```

7. insert():

```
insert(iterator pos, ele); // 迭代器指向pos前插入元素ele
insert(iterator pos, int n, ele);// 迭代器指向pos前插入n个元素ele
insert(iterator pos, iterator begin, iterator end);  // 迭代器指向pos前插入其他容器（不仅限于此vector）中[begin, end)的元素



v.insert(v.begin(), 20);  //首位插入20
v.insert(v.begin() + 5, 20); // 在第6位前插入20
v.insert(v.end(), g.begin(),g.end()); //在尾部插入另一个向量g的所有元素
```

8. 遍历
   i. 利用size（）遍历，size（）函数会返回向量实际存储的元素数量

```
void print_vector(vector<int> v)
{
    for(int i = 0; i < v.size(); ++i)
        cout << v[i]<< endl;
}
```

ii. 利用迭代器进行遍历
格式： vector<数据类型>::iterator 迭代器名称

```
void print_vector(vector<int> v)
{
    vector<int>::iterator it;
    for(it = v.begin();it != v.end(); it++)
        cout<< *it;
}
```

iii. 利用C++11特性进行遍历，向量v中每个元素的值会依次赋给x：

```
void print_vector(vector<int> v){
    for(int x:v) cout <<x <<endl;
}
```

2.3 二维数组的存储

- 题目描述

题目描述

如何利用向量 vector 的特性，完成二维数组的存储？

	第一列	第二列	第三列
第一行	0	1	2
第二行	1	2	3
第三行	2	3	4

例如：存储上方的二维数组

可考虑声明向量类型的向量：vector< vector<int> > v;

- 参考代码

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

int main()
{
    // 声明向量类型的向量
    vector< vector<int> > v;
    vector<int> g;
    int i, j, m;
    for (i = 0; i < 3; i++)
    {
        // 将向量 g 添加到向量 v，可表示二维数组的行
        v.push_back(g);
        // 并填充这一行各列的数据
        for (j = 0; j < 3; j++)
        {
            cin >> m;
            v[i].push_back(m);
        }
    }
    // 依次打印各行数据进行验证
    for(i = 0; i < 3; i++)
    {
        for(j = 0; j < 3; j++)
        {
            cout << v[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```