一、选择题：

1. 关于容器，以下说法不正确的是？C

A. map容器可以显式定义Key的排序方式

B. 顺序容器中，在中间插入元素时，list容器的插入成本最低

C. 顺序容器都可以使用STLalgorithm里的sort函数 //list要用自己的sort

D. vector容器中的元素要求存放在连续的内存中

2. 关于构造函数，以下说法正确的是：C

A. 只有一个参数的构造函数是类型转换构造函数。//有可能是本类的const引用

B. 构造函数不可以被重载。//可以

C. 构造函数可以是私有的。

D. 构造函数可以是虚函数。//构造函数不可以是虚函数 析构函数可以是虚函数

3. 下列STL模板中，不支持随机访问迭代器的是A

A. queue //只能取队首

B. vector

C. deque

D. string

二、填空题：

描述：

对于一幅灰度的图像，积分图中的任意一点(x,y)的值是指从图像的左上角到这个点的所构成的矩形区域内所有的点的灰度值之和。

代码：

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

class IntegralImage{

// 在此处补充你的代码

};

int main(){

int H, W;

cin >> H >> W;

int \*\* image = new int\*[H];

for(int i=0;i<H;++i){

image[i]=new int[W];

}

for(int i=0;i<H;++i)

for(int j=0;j<W;++j)

cin >> image[i][j];

IntegralImage it(H,W);

for\_each(image, image+H, it);

for(int i=0;i<H;++i){

for(int j=0;j<W;++j)

cout<<it[i][j]<<" ";

cout<<endl;

}

}

输入：

第一行两个整数，分别是图像的宽、高H, W

接下来H\*W的矩阵，分别代表图像的每个像素值

输出：

积分图中每个点的值, H\*W的矩阵，每个像素之间用空格分开

样例输入：

2 3

1 2 3

1 0 0

样例输出：

1 3 6

2 4 7

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

class IntegralImage {

// 在此处补充你的代码

public:

int h;

int w;

int\*\* img;

IntegralImage(int H, int W):h(H),w(W){

img = new int\* [h];

for (int i = 0; i < h; ++i) {

img[i] = new int[w];

}

for (int i = 0; i < h; ++i) {

for (int j = 0; j < w; ++j) {

img[i][j] = -1;

}

}

}

void operator()(int \*p) {

int cnt = 0;//该往这一排写了

for (int i = 0; i < h; ++i) {

if (img[i][0] == -1) {

cnt = i;

break;

}

}

if (cnt == 0) {

img[cnt][0] = p[0];

for (int i = 1; i < w; ++i) {

img[cnt][i] = img[cnt][i - 1]+p[i];

}

}

else {

int num = p[0];

img[cnt][0] = img[cnt - 1][0] + p[0];

for (int i = 1; i < w; ++i) {

img[cnt][i] = num+img[cnt - 1][i] + p[i];

num += p[i];

}

}

}

int\* operator[] (int i) {

return img[i];

}

};

int main() {

int H, W;

cin >> H >> W;

int\*\* image = new int\* [H];

for (int i = 0; i < H; ++i) {

image[i] = new int[W];

}

for (int i = 0; i < H; ++i)

for (int j = 0; j < W; ++j)

cin >> image[i][j];

IntegralImage it(H, W);

for\_each(image, image + H, it);

for (int i = 0; i < H; ++i) {

for (int j = 0; j < W; ++j)

cout << it[i][j] << " ";

cout << endl;

}

}