1. STL提供了三个序列式容器：

向量（vector）、双端队列（deque）、列表（list）

四个关联式容器：集合（set）、多重集合（multiset）、映射（map）和多重映射（multimap）。

算法：

汉诺塔，冒泡排序，斐波那切数列

1. 关于C ++中的纯虚函数和抽象类的使用

依据代码可知，构造纯虚函数A,一般作为一个抽象类接口，优点是可以和main函数等其它模块分离开，减少耦合性，main函数直接面向象类编程，属于业务层；通过B实例化A，只需要重写virtual void init(const std::string& str,int a,const int& b,int& c)const并且打开{}写内容就好，最后将抽象函数A借指针指向调用的函数A \*pA1 = new B();。

【注意：通过抽象类定义变量是不可以的，eg: Shape s;而且一旦声明纯虚函数，一定要实例化出来】

【自己的示例】

#include<iostream>

using namespace std;

//图形类

//如果说一个类，拥有一个纯虚函数

//就称这个类是一个抽象类。抽象类将所有模块都隔离开

//不管这个类中有没有成员属性，只要这个类有纯虚函数，就是一个抽象类，抽象类不能够实例化

class Shape

{

public:

//求图形面积的方法

//表示图形类声明一个方法getArea(),它是一个纯虚函数，没有函数的实现。

virtual double getArea()=0;

//virtual void print()=0;

};

//正方形

//如果说一个普通类，继承拥有纯虚函数的类,如果不重写，依然是一个抽象类

//依然不能被实例化，如果想实例化，必须重写这个父类中的纯虚函数

class Rect:public Shape

{

public:

Rect(int a){

this->a=a;

}

virtual double getArea(){

cout<<"正方形求面积"<<endl;

return a\*a;

}

private:

int a;//正方形边长

};

class Circle:public Shape

{

public:

Circle(int r)

{

this->r=r;

}

virtual double getArea(){

cout<<"圆方形求面积"<<endl;

return 3.14\*r\*4;

}

private:

int r;//半径

};

//业务层，面向抽象类编程

int main(void)

{

//直接通过抽象类定义变量是不可以的，eg: Shape s;

//main中所有使用的变量类型，都是抽象类Shaped的类型

Shape \*sp1 = new Rect(10);

sp1->getArea();

Shape \*sp2 = new Circle(20);

sp2->getArea();

return 0;

}

1. 40

#include<opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include<iostream>

#include<math.h>

using namespace std;

using namespace cv;

int main(int argc, char\*\* argv) {

//------------------------------------------标准第一步-----------------------

Mat src, gray\_src;

src = imread("C:/Users/15064/Desktop/拓比/test.png");

if (!src.data) {

cout << "could not load image...\n" << endl;

return -1;

}

namedWindow("input", CV\_WINDOW\_AUTOSIZE);

imshow("input", src);

waitKey(2);

//------------------------------------------标准第一步结束----------------------

//三通道

Mat dst;

dst.create(src.size(), src.type());

int height = src.rows;//长

int width = src.cols;//宽

int nc = src.channels();//获取通道数目

printf("图片长度：%d\n", height);

printf("图片宽度：%d", width);

for (int row = 0; row < height; row++) { //行循环

for (int col = 0; col < width; col++) { //列循环

if (nc == 1) {

int gray = gray\_src.at<uchar>(row, col);

gray\_src.at<uchar>(row, col) = 255 - gray;

}

else if (nc == 3) {

int b = src.at<Vec3b>(row, col)[0];

int g = src.at<Vec3b>(row, col)[1];

int r = src.at<Vec3b>(row, col)[2];

cout << src.at<Vec3b>(row, col) << endl;

dst.at<Vec3b>(row, col)[0] = b;

dst.at<Vec3b>(row, col)[1] = g;

dst.at<Vec3b>(row, col)[2] = r;

}

}

}

namedWindow("show1", 0);

imshow("show1", dst);

waitKey(0);

return 0;

}

编程题：



#include<opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include<iostream>

#include<math.h>

using namespace std;

using namespace cv;

Mat src, gray\_src, drawImg;

int threshold\_v = 170;

int threshold\_max = 255;

const char\* output\_win = "rectangle-demo";

RNG rng(12345);

void Contours\_Callback(int,void\*);

int main(int argc, char\*\* argv) {

src = imread("C:/Users/15064/Desktop/拓比/test.png");

if(!src.data) {

cout << "could not load image...\n" << endl;

return -1;

}

cvtColor(src, gray\_src, CV\_BGR2GRAY);

blur(gray\_src, gray\_src, Size(3, 3), Point(-1, -1));//均值模糊

const char\* source\_win = "input image";

namedWindow(source\_win, CV\_WINDOW\_AUTOSIZE);

namedWindow(output\_win, CV\_WINDOW\_AUTOSIZE);

imshow(source\_win, src);

createTrackbar("Threshold Value:", output\_win, &threshold\_v, threshold\_max, Contours\_Callback);//触动一个滑块

Contours\_Callback(0, 0);

waitKey(0);

return 0;

}

void Contours\_Callback(int,void\*) {

Mat binary\_output;

vector<vector<Point>>contours;

vector<Vec4i>hierachy;

threshold(gray\_src, binary\_output, threshold\_v, threshold\_max, THRESH\_BINARY);

imshow("binary image", binary\_output);

findContours(binary\_output, contours, hierachy, RETR\_TREE, CHAIN\_APPROX\_SIMPLE, Point(-1, -1));

}