**[data.h]**

#ifndef DATA\_H\_

#define DATA\_H\_

#include <ctime>

#include <cassert>

#include <iostream> // nullptr

// 针对 1000x800 的窗口生成数据

namespace data {

//生成[5,795]之间的随机数

int \*GenatateRandomData(const int size, int range\_l = 5, int range\_r = 795) {

int \*data = new int[size];

assert(data);

srand(time(nullptr));

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

data[i] = rand() % (range\_r - range\_l + 1) + range\_l;

}

return data;

}

};

#endif //DATA\_H\_

**[main.cpp]**

#include "opencv2/core/core.hpp""

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <algorithm>

using namespace cv;

using namespace std;

#include "data.h"

const int kWidth = 1000; // x 宽度

const int kHeight = 800; // y 高度

Mat img = Mat::zeros(kHeight, kWidth, CV\_8UC3); //定义白板窗口

int index\_l = -1;

int index\_r = -1;

int pivot = -1;

const int kSize = 100;

// Scalar(Blue, Green, Red)

// 根据 data 数组，绘制 size 个小矩形

void Draw(int data[], int size) {

img.setTo(0);

int sz = 0;

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

sz = i \* 10;

if (i >= index\_l && i <= index\_r) {

rectangle(img, Rect(Point(sz + 2, data[i]), Point(sz + 10,795)), Scalar(255, 0, 0));

if (i == pivot) {

rectangle(img, Rect(Point(sz + 2, data[i]), Point(sz + 10, 795)), Scalar(0,255,0));

}

}

else {

rectangle(img, Rect(Point(sz + 2, data[i]), Point(sz + 10, 795)), Scalar(0,0,255));

}

}

imshow("快速排序可视化",img);

}

// 寻找 p，使得 arr[l ... p-1] <= arr[p] <= arr[p+1 ... r]

int partition(int data[], const int l, const int r) {

// v 为基准值

int random = rand() % (r - l + 1) + l;

std::swap(data[random], data[l]);

int v = data[l];

int p = l;

int i = l + 1;

pivot = l;

index\_l = l;

index\_r = r;

Draw(data, kSize);

waitKey(500);

while (true) {

if (i > r) {

break;

}

if (data[i] > v) {

p++;

std::swap(data[p], data[i]);

}

i++;

}

std::swap(data[l], data[p]);

return p;

}

void quick\_sort(int data[], const int l, const int r) {

if (l >= r) {

return;

}

int p = partition(data, l, r);

quick\_sort(data, l, p - 1);

quick\_sort(data, p + 1, r);

}

// 快速排序

void QuickSort(int data[], const int size) {

srand(time(nullptr));

quick\_sort(data, 0, size - 1);

}

int main()

{

namedWindow("快速排序可视化",1);

int size = kSize;

int \*data = data::GenatateRandomData(size); //获取数据

Draw(data, size);

imshow("快速排序可视化", img);

waitKey(1000);

QuickSort(data, size);

imshow("快速排序可视化", img);

waitKey(0);

delete[] data; //释放空间

return 0;

}

 

