2016 金山 wps 技术岗

1.笔试题

2016.4.11 晚上在中山大学东校区（大学城校区）参加了金山 WPS 的笔试。记忆较为深刻的有如下几题。

题目一：

以下代码片段，输出的结果是什么?

```
vector<int> vec(5);
cout<<vec.size()<<endl;      //1
vec.reserve(100);
cout<<vec.size()<<endl;      //2
vec.resize(50);
cout<<vec.size()<<endl;      //3
cout<<vec.capacity()<<endl;//4
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 1

```
2
3
4
5
6
7
```

本题考察的是 vector 向量容器的成员函数 reserve()和 resize()的作用和区别。reserve()用来改变 vector 向量容器的容量，即 vec.capacity()的返回值。resize()用于改变 vector 的元素数量。所以代码中 1,2,3,4 的输出依次是：5，5，50，100。

题目二：

这是一道编程题，求三个矩形的交集矩形。

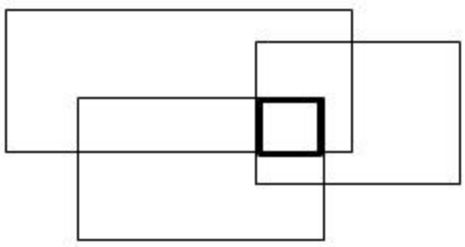
给定矩形的定义如下：

```
struct Rect{
    int x; //表示矩形的左上水平坐标
    int y; //表示矩形的左上垂直坐标
    int w; //表示矩形宽度
    int h; //表示矩形高度
};
```

```
1
```

- 2
- 3
- 4
- 5
- 6
- 1
- 2
- 3
- 4
- 5
- 6

现在给三个矩形，求三个矩形的交集，如果没有交集，那么矩形的 x，y，w 和 h 均赋值为−1。例如下面示例图，求出三个矩形相交的粗线线框表示的矩形。



解题思路：

解题思路很重要，没有集体思路，题目肯定是做出不来的。下面给出本人的解题思路：

（1）判断三个矩形有没有交集。这个是难点，该怎么做呢？可以在 x 轴

方向将三个矩形按 x 的大小从左到右排列，判断两两矩形在 x 轴方向是否有交集，如果有任意一对没有相交那么三个矩形没有交集。判断方法是如果 rectB.x>=rectA.x+rectA.w 的话，那么说明 rectA 和 rectB 之间没有交集。

同理，在 y 轴方向做同样的判断；

（2）求出任意两个矩形的交集矩形，再将交集矩形与第三个矩形再求交集，可得最后的交集矩形。

有了正确和清晰的思路，就可以写代码了，下面给出本人的实现，可供网友参考。

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

struct Rect{
    int x; //表示矩形的左上水平坐标
    int y; //表示矩形的左上垂直坐标
    int w; //表示矩形宽度
    int h; //表示矩形高度
};

//按照 x 递增排序
```

```
bool compareX(const Rect& rectA,const Rect& rectB ){
    return rectA.x<rectB.x;
}

//按照 y 递增排序
bool compareY(const Rect& rectA,const Rect& rectB ){
    return rectA.y<rectB.y;
}

//判断三个矩形是否相交
bool isIntersect(const Rect& rectA,const Rect& rectB,const Re
ct& rectC){
    Rect rectLeft,rectXMid,rectRight; //从左向右的矩形
    Rect rectTop,rectYMid,rectBelow;  //从上到下的矩形

    //将矩形按照 x 由左向右排序
    vector<const Rect> vec;
    vec.push_back(rectA);
    vec.push_back(rectB);
    vec.push_back(rectC);
    sort(vec.begin(),vec.end(),compareX);
    rectLeft=vec[0],rectXMid=vec[1],rectRight=vec[2];
```

```
    //水平方向任意两个矩形没有交集
    if(rectXMid.x>=rectLeft.x+rectLeft.w||rectRight.x>=rectXMid.x
+rectXMid.w||rectRight.x>=rectLeft.x+rectLeft.w)
        return false;

    //同理将矩形按照 y 由上往下排序
    sort(vec.begin(),vec.end(),compareY);
    rectTop=vec[0],rectYMid=vec[1],rectBelow=vec[2];

    //垂直方向任意两个矩形没有交集
    if(rectYMid.y>=rectTop.y+rectTop.h||rectBelow.y>=rectYMid.y
+rectYMid.h||rectBelow.y>=rectTop.y+rectTop.h)
        return false;
    return true; //三个矩形有交集
}

//两个矩形的交集，前提是两个矩形一定有交集
Rect intersection(const Rect& rectA,const Rect& rectB){
    Rect resRect;
    resRect.x=rectA.x>rectB.x?rectA.x:rectB.x; //选最右边的矩形
的 x 作为交集的 x
```

```
        resRect.y=rectA.y>rectB.y?rectA.y:rectB.y; //选最下面的矩形的 y 作为交集的 y
        //选择左边矩形（x 坐标较小者）的右边的作为交集矩形的右边，这样就可以求出交集矩形的宽度
        resRect.w=rectA.x+rectA.w<rectB.x+rectB.w?rectA.x+rectA.w-resRect.x:rectB.x+rectB.w-resRect.x;
        //同理，选择上面矩形（y 坐标较小者）的下边的作为交集矩形的下边，这样就可以求出交集矩形的高度
        resRect.h=rectA.y+rectA.h<rectB.y+rectB.y?rectA.y+rectA.h-resRect.y:rectB.y+rectB.h-resRect.y;
        return resRect;
}


//求三个矩形的交集
Rect threeIntersection(const Rect& rectA,const Rect& rectB,const Rect& rectC){
        Rect res;
        bool isIntersectBool=isIntersect(rectA,rectB,rectC);
        if(isIntersectBool){ //有相交
                Rect rectAB=intersection(rectA,rectB);
                res=intersection(rectAB,rectC);
```

```
        }
        else
            res.x=res.y=res.w=res.h=-1;
        return res;
    }
```

- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39

- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61

- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31

- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53

- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74

测试结果如下：

```
int  main(){
    Rect  rectA,rectB,rectC;
    //测试案例 1
    //rectA.x=0,rectA.y=0,rectA.w=1,rectA.h=1;
    //rectB.x=1,rectB.y=1,rectB.w=1,rectB.h=1;
    //rectC.x=2,rectC.y=2,rectC.w=1,rectC.h=1;

    //测试案例 2
    rectA.x=0,rectA.y=0,rectA.w=2,rectA.h=2;
    rectB.x=1,rectB.y=1,rectB.w=1,rectB.h=1;
    rectC.x=1,rectC.y=1,rectC.w=1,rectC.h=1;

    Rect  resRect=threeIntersection(rectA,rectB,rectC);
    if(resRect.x!=-1){  //有相交
        cout<<"resRect.x:"<<resRect.x<<endl;
        cout<<"resRect.y:"<<resRect.x<<endl;
        cout<<"resRect.w:"<<resRect.x<<endl;
        cout<<"resRect.h:"<<resRect.x<<endl;
    }
    else
```

```
            cout<<"not  intersect"<<endl;
        getchar();
    }
```

- 20
- 21
- 22
- 23
- 24
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

```
18
19
20
21
22
23
24
```

测试案例 1 输出：not intersect；

测试案例 2 输出：

resRect.x:1

resRect.y:1

resRect.w:1

resRect.h:1