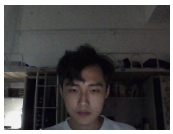
[

每日一题day15_6月5日测评结果

考生信息

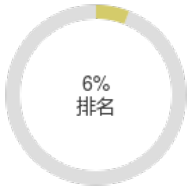


张博翔

考号：1675 | 学校：陕西科技大学 | 邮箱：1761607418@qq.com | 职位：43班 |

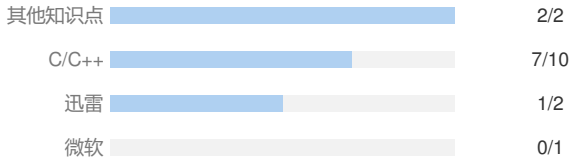
参考区域: 陕西省西安市（111.114.0.1） | 做题用时：01:52:53(2019-06-04 23:04:03 - 2019-06-05 00:57:01)

考生成绩



题型	得分	正确题数	排名	用时	是否阅卷
单选	35.0	7	19	00:48:07	--
编程	50.0	2	1	01:01:27	--

知识点技能图谱



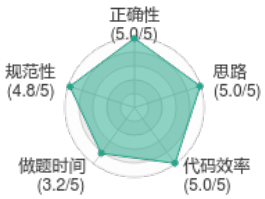
知识点	得分	正确题数
其他知识点	50.0	2
C/C++	35.0	7
迅雷	5.0	1
微软	0.0	0

历史笔试记录

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	笔试时间
1	每日一题day1_5月20日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-19 16:08:49
2	每日一题day02_5月21日	51.0%	60.0/100	单选:30.0分 编程:30.0分	否	2019-05-20 17:40:56
3	每日一题day03_5月22日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-21 12:59:15
4	每日一题day04_5月23日	8.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-22 10:40:18
5	每日一题day05_5月24日	29.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-22 20:39:16
6	每日一题day06_5月25日	10.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-24 10:41:21
7	每日一题day07_5月27日	6.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-26 15:09:41
8	每日一题day08_5月28日	18.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-27 14:06:31
9	每日一题day09_5月29日	13.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-28 15:11:53
10	每日一题day10_5月30日	6.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-29 11:32:20

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	笔试时间
11	每日一题day11_5月31日	14.0%	80.0/100	单选:30.0分 编程:50.0分	否	2019-05-30 11:22:09
12	每日一题day12_6月1日	31.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-31 10:33:31
13	每日一题day13_6月3日	9.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-02 15:20:12
14	每日一题day14_6月4日	41.0%	50.0/100	单选:25.0分 编程:25.0分	否	2019-06-03 11:53:13

编码能力



题号	正确性	提交次数	做题用时	使用语言	运行时间	占用内存	编程思路	代码规范	成绩排名
编程题1	100%	1	00:06:38	C++	5ms	492K	优	优	1%
编程题2	100%	2	00:54:49	C++	4ms	604K	优	优	1%

1 [平均分3.9分 | 96人正确/123人做题 | 用时：<1分 得分：5.0 / 5.0

对两个字符a和b进行初始化:char a[]="ABCDEF";char b[]={'A','B','C','D','E','F'};则以下叙述正确的是：

A a数组比b数组长度长
B a与b长度相同
C a与b数组完全相同
D a和b中都存放字符串

他的回答： A (正确)

正确答案： A

2 [平均分3.1分 | 75人正确/122人做题 | 用时：2分 得分：5.0 / 5.0

x是一个行列数均为1000二维数组，下面代码效率执行最高的是（ ）

A for(int j=0;j<1000;j++) for(int i=0;i<1000;i++) x[i][j]+=x[j][i];
B for(int i=0;i<1000;j++) for(int j=0;j<1000;j++) x[i][j]+=x[j][i];
C for(int i=0;i<1000;j++) for(int j=0;j<1000;j++) x[j][i]+=x[j][i];
D for(int i=0;i<1000;i++) for(int j=0;j<1000;j++) x[i][j]+=x[i][j];

他的回答： D (正确)

正确答案： D

3 [平均分2.7分 | 67人正确/124人做题 | 用时：8分 得分：5.0 / 5.0

C++中关于堆和栈的说法，哪个是错误的:

A 堆的大小仅受操作系统的限制，栈的大小一般一般较小
B 在堆上频繁的调用new/delete容易产生内存碎片，栈没有这个问题
C 堆和栈都可以静态分配
D 堆和栈都可以动态分配

他的回答： C (正确)

正确答案： C

4 [平均分2.2分 | 53人正确/122人做题 | 用时：2分 得分：0.0 / 5.0

下面程序会输出什么：

```
static int a=1;
void fun1(void){    a=2;  }
void fun2(void){    int a=3;  }
void fun3(void){   static int a=4;  }
int main(int argc,char** args){
    printf(“%d”,a);
    fun1( );
    printf(“%d”,a);
    fun2( );
    printf(“%d” , a);
    fun3( );
     printf(“%d”,a);
}
```

A 1 2 3 4
B 1 2 2 2
C 1 2 2 4
D 1 1 1 4

他的回答： C (错误)

正确答案： B

---

5 [平均分2.3分 | 51人正确/113人做题 | 用时：15分 得分：0.0 / 5.0

In the main() function, after ModifyString(text) is called, what’s the value of ‘text’?

```
int FindSubString( char* pch )
{
   int   count  = 0;
   char  * p1   = pch;
   while ( *p1 != '\0' )
   {
      if ( *p1 == p1[1] - 1 )
      {
         p1++;
         count++;
      }else  {
         break;
      }
   }
   int count2 = count;
   while ( *p1 != '\0' )
   {
      if ( *p1 == p1[1] + 1 )
      {
         p1++;
         count2--;
      }else  {
         break;
      }
   }
   if ( count2 == 0 )
      return(count);
   return(0);
}
void ModifyString( char* pText )
{
   char  * p1   = pText;
   char  * p2   = p1;
   while ( *p1 != '\0' )
```

```
    {
        int count = FindSubString( p1 );
        if ( count > 0 )
        {
            *p2++ = *p1;
            sprintf( p2, "%i", count );
            while ( *p2 != '\0' )
            {
                p2++;
            }
            p1 += count + count + 1;
        }else  {
            *p2++ = *p1++;
        }
    }
}
void main( void )
{
    char text[32] = "XYBCDCBABABA";
    ModifyString( text );
    printf( text );
}
```

A XYBCDCBABABA
B XYBCBCDA1BAA
C XYBCDCBA1BAA
D XYBCDDBA1BAB

他的回答： A (错误)

正确答案： C

---

6 [平均分4.7分 | 115人正确/122人做题 | 用时：<1分 得分：5.0 / 5.0

所谓数据封装就是将一组数据和与这组数据有关操作组装在一起，形成一个集合，这集合也就是（ ）

A 类
B 对象
C 函数体
D 数据块

他的回答： A (正确)

正确答案： A

---

7 [平均分1.2分 | 28人正确/121人做题 | 用时：2分 得分：5.0 / 5.0

关于以下代码，哪个说法是正确的？

```
myClass::foo(){
    delete this;
}
..
void func(){
    myClass *a = new myClass();
    a->foo();
}
```

A 它会引起栈溢出
B 都不正确
C 它不能编译
D 它会引起段错误

他的回答： B (正确)

正确答案： B

8 [平均分1.1分 | 27人正确/122人做题 | 用时：13分 得分：5.0 / 5.0

假定CSomething是一个类，执行下面这些语句之后，内存里创建了____个CSomething对象。

```
CSomething a();
CSomething b(2);
CSomething c[3];
CSomething &ra = b;
CSomething d=b;
CSomething *pA = c;
CSomething *p = new CSomething(4);
```

A 10

B 9

C 8

D 7

E 6

F 5

他的回答： E (正确)

正确答案： E

9 [平均分3.2分 | 75人正确/116人做题 | 用时：<1分 得分：0.0 / 5.0

下面这段代码运行时会出现什么问题？

```
class A
{
public:
    void f()
    {
        printf("A\n");
    }
};

class B: public A
{
public:
    virtual void f()
    {
        printf("B\n");
    }
};

int main()
{
    A *a = new B;
    a->f();
    delete a;
    return 0;
}
```

A 没有问题，输出B
B 不符合预期的输出A
C 程序不正确
D 以上答案都不正确

他的回答： A (错误)

正确答案： B

---

10 [平均分2.3分 | 53人正确/117人做题 | 用时：2分 得分：5.0 / 5.0

下面这段代码会打印出什么？

```
class A
{
public:
    A()
    {
        printf("A ");
    }

    ~A()
    {
        printf("deA ");
    }
};

class B
{
public:
    B()
    {
        printf("B ");
    }
    ~B()
    {
        printf("deB ");
    }
};

class C: public A, public B
{
public:
    C()
    {
        printf("C ");
    }
    ~C()
    {
        printf("deC ");
    }
};
int main()
{
    A *a = new C();
    delete a;
    return 0;
}
```

A A B C deA

B C A B deA
C A B C deC
D C A B deC

他的回答： A (正确)

正确答案： A

---

11 [平均分23.0分 | 105人正确/114人做题 | 提交: 1 次 得分：25.0 / 25.0

标题：查找输入整数二进制中1的个数 | 时间限制：1秒 | 内存限制：32768K | 语言限制： 不限

【查找输入整数二进制中1的个数】

请实现如下接口

```
public static int findNumberOf1( int num)
{
    /* 请实现 */
    return 0;
} 譬如：输入5 ，5的二进制为101，输出2
```

涉及知识点：

输入描述：

> 输入一个整数

输出描述：

> 计算整数二进制中1的个数

示例1：

输入

> 5

输出

> 2

代码片段

功能实现			代码提交统计			代码执行统计
	TA的	平均		TA的	平均	答案正确 ：1
总通过率	100%	92%	使用语言	C++		
基本测试用例通过率	6/6 (100%)	92%	做题用时	00:06:38	00:11:48	
边缘测试用例通过率	4/4 (100%)	92%	提交次数	1	4	

代码效率			代码规范及可读性
	TA的	参考	代码规范得分 5.0
运行时间	5ms	1s	
占用内存	492K	32768K	

他的代码：

做题用时: 6 分钟 语言：C++ 运行时间：5ms 占用内存：492K 程序状态：答案正确

```
#include <iostream>
using namespace std;
int main()
{
    int num = 0;
    while(cin >> num)
    {
```

```
        int count = 0;
        while(num)
        {
            if((num & 1) == 1)
            {
                count++;
            }
            num >>= 1;
        }
        cout << count << endl;
    }
}
```

12 [平均分18.1分 | 52人正确/72人做题 | 提交: 2 次 得分：25.0 / 25.0

标题：手套 | 时间限制：3秒 | 内存限制：32768K | 语言限制： [Python, C++, C#, Java]

【手套】

在地下室里放着n种颜色的手套，手套分左右手，但是每种颜色的左右手手套个数不一定相同。A先生现在要出门，所以他要去地下室选手套。但是昏暗的灯光让他无法分辨手套的颜色，只能分辨出左右手。所以他会多拿一些手套，然后选出一双颜色相同的左右手手套。现在的问题是，他至少要拿多少只手套(左手加右手)，才能保证一定能选出一双颜色相同的手套。

给定颜色种数n(1≤n≤13),同时给定两个长度为n的数组left,right,分别代表每种颜色左右手手套的数量。数据保证左右的手套总数均不超过26，且一定存在至少一种合法方案。

测试样例：

```
4,[0,7,1,6],[1,5,0,6]
```

```
返回：10(解释：可以左手手套取2只，右手手套取8只)
```

输入描述：

输出描述：

代码片段

功能实现			代码提交统计			代码执行统计
	TA的	平均		TA的	平均	编译错误 ：1
总通过率	100%	72%	使用语言	C++		答案正确 ：1
基本测试用例通过率	1/1 (100%)	72%	做题用时	00:54:49	01:01:05	
			提交次数	2	3	

代码效率			代码规范及可读性
	TA的	参考	代码规范得分 4.6
运行时间	4ms	3s	Line 27: Add #include for min [build/include_what_you_use] [4]
占用内存	604K	32768K	Line 3: Add #include for vector<> [build/include_what_you_use] [4]

他的代码：

做题用时: 54 分钟 语言：C++ 运行时间：4ms 占用内存：604K 程序状态：答案正确

```
class Gloves {
public:
    int findMinimum(int n, vector<int> left, vector<int> right) {
        // write code here
        int count = 0;
        int leftSum = 0;
        int rightSum = 0;
```

```
        int leftMin = 27;
        int rightMin = 27;
        for(int i = 0; i < n; i++)
        {
            //如果不存在这种颜色手套有一只不匹配则为无效解，这些手套必须全部拿上
            if(left[i] == 0 || right[i] == 0)
            {
                count += left[i];
                count += right[i];
            }
            else
            {
                leftSum += left[i];
                rightSum += right[i];
                leftMin = min(leftMin, left[i]);
                rightMin = min(rightMin, right[i]);
            }
        }
        //根据数学原理可知要把一只手所有颜色手套全拿完需要这只手所有颜色手套总和-最小值+1
        return count + min(leftSum - leftMin + 1, rightSum - rightMin + 1) + 1;
    }
};
```