


15道使用频率极高的基础算法实现代码

13 回复 904 查看



(<https://www.shiyanlou.com/user/8490>) 实验楼管理员  (<https://www.shiyanlou.com/vip>)

2015-11-04 16:19

来自: 数据结构与算法 (<https://www.shiyanlou.com/questions/courses/484>)


技术分享 (<https://www.shiyanlou.com/questions/?tag=技术分享>)

- 1、合并排序，将两个已经排序的数组合并成一个数组，其中一个数组能容下两个数组的所有元素;
- 2、合并两个已经排序的单链表;
- 3、倒序打印一个单链表;
- 4、给定一个单链表的头指针和一个指定节点的指针，在O(1)时间删除该节点;
- 5、找到链表倒数第K个节点;
- 6、反转单链表;
- 7、通过两个栈实现一个队列;
- 8、二分查找;
- 9、快速排序;
- 10、获得一个int型的数中二进制中的个数;
- 11、输入一个数组，实现一个函数，让所有奇数都在偶数前面;
- 12、判断一个字符串是否是另一个字符串的子串;
- 13、把一个int型数组中的数字拼成一个串，这个串代表的数字最小;
- 14、输入一颗二叉树，输出它的镜像（每个节点的左右子节点交换位置）;
- 15、输入两个链表，找到它们第一个公共节点;



全部回答



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

代码实现:

```

//链表节点
struct NodeL
{
    int value;
    NodeL* next;
    NodeL(int value_=0,NodeL* next_=NULL):value(value_),next(next_){}
};
//二叉树节点
struct NodeT
{
    int value;
    NodeT* left;
    NodeT* right;
    NodeT(int value_=0,NodeT* left_=NULL,NodeT* right_=NULL):value(value_),left(left_),right(right_){}
};

```

1、合并排序，将两个已经排序的数组合并成一个数组，其中一个数组能容下两个数组的所有元素;

合并排序一般的思路都是创建一个更大数组C，刚好容纳两个数组的元素，先是一个while循环比较，将其中一个数组A比较完成，将另一个数组B中所有的小于前一个数组A的数及A中所有的数按顺序存入C中，再将A中剩下的数存入C中，但这里是已经有一个数组能存下两个数组的全部元素，就不用在创建数组了，但只能从后往前面存，从前往后存，要移动元素很麻烦。

```

//合并排序，将两个已经排序的数组合并成一个数组，其中一个数组能容下两个数组的所有元素
void MergeArray(int a[],int alen,int b[],int blen)
{
    int len=alen+blen-1;
    alen--;
    blen--;
    while (alen>=0 && blen>=0)
    {
        if (a[alen]>b[blen])
        {
            a[len--]=a[alen--];
        }else{
            a[len--]=b[blen--];
        }
    }

    while (alen>=0)
    {
        a[len--]=a[alen--];
    }
    while (blen>=0)
    {
        a[len--]=b[blen--];
    }
}

void MergeArrayTest()
{
    int a[]={2,4,6,8,10,0,0,0,0,0};
    int b[]={1,3,5,7,9};
    MergeArray(a,5,b,5);
    for (int i=0;i<sizeof(a)/sizeof(a[0]);i++)
    {
        cout<<a[i]<<" ";
    }
}

```

2015-11-04 16:20



实验楼管理员 (<https://www.shiyanlou.com/user/8490>) 💛 (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

2、合并两个单链表;

合并链表和合并数组，我用了大致相同的代码，就不多少了，那本书用的是递归实现。

```

//链表节点

```

```

struct NodeL
{
    int value;
    NodeL* next;
    NodeL(int value_=0,NodeL* next_=NULL):value(value_),next(next_){}
};

```

//合并两个单链表

```
NodeL* MergeList (NodeL* head1,NodeL* head2)
```

```

{
    if (head1==NULL)
        return head2;
    if (head2==NULL)
        return head1;

    NodeL* head=NULL;
    if (head1->value<head2->value)
    {
        head=head1;
        head1=head1->next;
    }else{
        head=head2;
        head2=head2->next;
    }
    NodeL* tmpNode=head;
    while (head1 && head2)
    {
        if (head1->value<head2->value)
        {
            head->next=head1;
            head1=head1->next;
        }else{
            head->next=head2;
            head2=head2->next;
        }
        head=head->next;
    }
    if (head1)
    {
        head->next=head1;
    }
    if (head2)
    {
        head->next=head2;
    }
    return tmpNode;
}

```

```
void MergeListTest()
```

```

{
    NodeL* head1=new NodeL(1);
    NodeL* cur=head1;
    for (int i=3;i<10;i+=2)
    {
        NodeL* tmpNode=new NodeL(i);
        cur->next=tmpNode;
        cur=tmpNode;
    }
    NodeL* head2=new NodeL(2);
    cur=head2;
    for (int i=4;i<10;i+=2)
    {
        NodeL* tmpNode=new NodeL(i);
        cur->next=tmpNode;
        cur=tmpNode;
    }
    NodeL* head=MergeList(head1,head2);
    while (head)
    {
        cout<<head->value<<" ";
        head=head->next;
    }
}

```

3、倒序打印一个单链表;


递归实现，先递归在打印就变成倒序打印了，如果先打印在调用自己就是顺序打印了。

```
//倒序打印一个单链表
void ReversePrintNode(NodeL* head)
{
    if (head)
    {
        ReversePrintNode(head->next);
        cout<<head->value<<endl;
    }
}

void ReversePrintNodeTest()
{
    NodeL* head=new NodeL();
    NodeL* cur=head;
    for (int i=1;i<10;i++)
    {
        NodeL* tmpNode=new NodeL(i);
        cur->next=tmpNode;
        cur=tmpNode;
    }
    ReversePrintNode(head);
}
```

2015-11-04 16:20



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

4、给定一个单链表的头指针和一个指定节点的指针，在O(1)时间删除该节点;

删除节点的核心还是将这个节点的下一个节点，代替当前节点。

//给定一个单链表的头指针和一个指定节点的指针，在 $O(1)$ 时间删除该节点

```
void DeleteNode(NodeL* head,NodeL* delNode)
{
    if (!head || !delNode)
    {
        return;
    }

    if (delNode->next!=NULL) //删除中间节点
    {
        NodeL* next=delNode->next;
        delNode->next=next->next;
        delNode->value=next->value;
        delete next;
        next=NULL;
    }else if (head==delNode) //删除头结点
    {
        delete delNode;
        delNode=NULL;
        *head=NULL;
    }else //删除尾节点，考虑到delNode不在head所在的链表上的情况
    {
        NodeL* tmpNode=head;
        while (tmpNode && tmpNode->next!=delNode)
        {
            tmpNode=tmpNode->next;
        }
        if (tmpNode!=NULL)
        {
            delete delNode;
            delNode=NULL;
            tmpNode->next=NULL;
        }
    }
}

void DeleteNodeTest()
{
    int nodeCount=10;
    for (int K=0;K<nodeCount;K++)
    {
        NodeL* head=NULL;
        NodeL* cur=NULL;
        NodeL* delNode=NULL;
        for (int i=0;i<nodeCount;i++)
        {
            NodeL* tmpNode=new NodeL(i);
            if (i==0)
            {
                cur=head=tmpNode;
            }else{
                cur->next=tmpNode;
                cur=tmpNode;
            }
            if (i==K)
            {
                delNode=tmpNode;
            }
        }
        DeleteNode(head,delNode) ;
    }
}
```

2015-11-04 16:21



实验楼管理员 (<https://www.shiyanlou.com/user/8490>) (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

3、找到链表倒数第K个节点;


通过两个指针，两个指针都指向链表的开始，一个指针先向前走K个节点，然后再以前向前走，当先走的那个节点到达末尾时，另一个节点就刚好与末尾节点相差K个节点。

```
//找到链表倒数第K个节点
NodeL* FindKthToTail(NodeL* head,unsigned int k)
{
    if(head==NULL || k==0)
        return NULL;
    NodeL* tmpNode=head;
    for (int i=0;i<k;i++)
    {
        if (tmpNode!=NULL)
        {
            tmpNode=tmpNode->next;
        }else{
            return NULL;
        }
    }
    NodeL* kNode=head;
    while (tmpNode!=NULL)
    {
        kNode=kNode->next;
        tmpNode=tmpNode->next;
    }
    return kNode;
}

void FindKthToTailTest()
{
    int nodeCount=10;
    for (int K=0;K<nodeCount;K++)
    {
        NodeL* head=NULL;
        NodeL* cur=NULL;
        for (int i=0;i<nodeCount;i++)
        {
            NodeL* tmpNode=new NodeL(i);
            if (i==0)
            {
                cur=head=tmpNode;
            }else{
                cur->next=tmpNode;
                cur=tmpNode;
            }
        }
        NodeL* kNode=FindKthToTail(head,K+3) ;
        if (kNode)
        {
            cout<<"倒数第 "<<K+3<<" 个节点是: "<<kNode->value<<endl;
        }else{
            cout<<"倒数第 "<<K+3<<" 个节点不在链表中" <<endl;
        }
    }
}
```

2015-11-04 16:21



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

6、反转单链表，

按顺序一个个的翻转就是了。

//反转单链表

```
NodeL* ReverseList(NodeL* head)
{
    if (head==NULL)
    {
        return NULL;
    }
    NodeL* reverseHead=NULL;
    NodeL* curNode=head;
    NodeL* preNode=NULL;
    while (curNode!=NULL)
    {
        NodeL* nextNode=curNode->next;
        if (nextNode==NULL)
            reverseHead=curNode;


        curNode->next=preNode;
        preNode=curNode;
        curNode=nextNode;
    }
    return reverseHead;
}

void ReverseListTest()
{
    for (int K=0;K<=10;K++)
    {
        NodeL* head=NULL;
        NodeL* cur=NULL;
        for (int i=0;i<K;i++)
        {
            NodeL* tmpNode=new NodeL(i);
            if (i==0)
            {
                cur=head=tmpNode;
            }else{
                cur->next=tmpNode;
                cur=tmpNode;
            }
        }

        cur=ReverseList( head);
        while (cur)
        {
            cout<<cur->value<<" ";
            cur=cur->next;
        }
        cout<<endl;
    }
    cout<<endl;
}
```

2015-11-04 16:22



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

7、通过两个栈实现一个队列;

直接上代码:


```
//通过两个栈实现一个队列
template<typename T>
class CQueue
{
public:
    void push(const T& val)
    {
        while (s2.size()>0)
        {
            s1.push(s2.top());
            s2.pop();
        }
        s1.push(val);
    }
    void pop()
    {
        while (s1.size()>0)
        {
            s2.push(s1.top());
            s1.pop();
        }
        s2.pop();
    }

    T& front()
    {
        while (s1.size()>0)
        {
            s2.push(s1.top());
            s1.pop();
        }
        return s2.top();
    }
    int size()
    {
        return s1.size()+s2.size();
    }
private:
    stack<T> s1;
    stack<T> s2;
};

void CQueueTest()
{
    CQueue<int> q;
    for (int i=0;i<10;i++)
    {
        q.push(i);
    }
    while (q.size()>0)
    {
        cout<<q.front()<<" ";
        q.pop();
    }
}
```

2015-11-04 16:22



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

8、二分查找

二分查找记住几个要点就行了，代码也就那几行，反正我现在是可以背出来了，start=0，end=数组长度-1，while(start<=end)，注意溢出。


```
//二分查找
int binarySearch(int a[],int len,int val)
{
    int start=0;
    int end=len-1;
    int index=-1;
    while (start<=end)
    {
        index=start+(end-start)/2;
        if (a[index]==val)
        {
            return index;
        }else if (a[index]<val)
        {
            start=index+1;
        }else
        {
            end=index-1;
        }
    }
    return -1;
}
```

9、快速排序;

来自百度百科，说不清楚:

```
//快速排序
//之前有个面试叫我写快排，想都没想写了个冒泡，思路早忘了，这段代码来自百度百科
void Qsort(int a[],int low,int high)
{
    if(low>=high)
    {
        return;
    }
    int first=low;
    int last=high;
    int key=a[first];//用字表的第一个记录作为枢轴
    while(first<last)
    {
        while(first<last && a[last]>=key )--last;
        a[first]=a[last];//将比第一个小的移到低端
        while(first<last && a[first]<=key )++first;
        a[last]=a[first];//将比第一个大的移到高端
    }
    a[first]=key;//枢轴记录到位
    Qsort(a,low,first-1);
    Qsort(a,last+1,high);
}

void QsortTest()
{
    int a[]={1,3,5,7,9,2,4,6,8,0};
    int len=sizeof(a)/sizeof(a[0])-1;
    Qsort(a,0,len);
    for(int i=0;i<=len;i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl;
}
```

2015-11-04 16:22



实验楼管理员 (<https://www.shiyanlou.com/user/8490>) (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

10、获得一个int型的数中二进制中的个数;

核心实现就是 `while (num= num&(num-1))`，通过这个数和比它小1的数的二进制进行&运算，将二进制中1慢慢的从后往前去掉，直到没有。

```
//获得一个int型的数中二进制中1的个数
int Find1Count(int num)
{
    if (num==0)
    {
        return 0;
    }
    int count=1;
    while (num= num & (num-1))
    {
        count++;
    }
    return count;
}
```

11、输入一个数组，实现一个函数，让所有奇数都在偶数前面；

两个指针，一个从前往后，一个从后往前，前面的指针遇到奇数就往后走，后面的指针遇到偶数就往前走，只要两个指针没有相遇，就奇偶交换。

//输入一个数组，实现一个函数，让所有奇数都在偶数前面

```
void RecordOddEven(int A[],int len)
{
    int i=0,j=len-1;
    while (i<j)
    {
        while (i<len && A[i]%2==1)
            i++;


        while (j>=0 && A[j]%2==0)
            j--;

        if (i<j)
        {
            A[i]^=A[j]^=A[i]^=A[j];
        }
    }
}

void RecordOddEvenTest()
{
    int A[]={1,2,3,4,5,6,7,8,9,0,11};
    int len=sizeof(A)/sizeof(A[0]);
    RecordOddEven( A , len);
    for (int i=0;i<len;i++)
    {
        cout<<A[i]<<" ";
    }
    cout<<endl;
    for (int i=0;i<len;i++)
    {
        A[i]=2;
    }
    RecordOddEven( A , len);
    for (int i=0;i<len;i++)
    {
        cout<<A[i]<<" ";
    }
    cout<<endl;
    for (int i=0;i<len;i++)
    {
        A[i]=1;
    }
    RecordOddEven( A , len);
    for (int i=0;i<len;i++)
    {
        cout<<A[i]<<" ";
    }
}
```

2015-11-04 16:23



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

12、判断一个字符串是否是另一个字符串的子串；

我这里就是暴力的对比

```

//判断一个字符串是否是另一个字符串的子串
int substr(const char* source,const char* sub)
{
    if (source==NULL || sub==NULL)
    {
        return -1;
    }
    int souLen=strlen(source);
    int subLen=strlen(sub);
    if (souLen<subLen)
    {
        return -1;
    }

    int cmpCount=souLen-subLen;
    for (int i=0;i<=cmpCount;i++)
    {
        int j=0;
        for (;j<subLen;j++)
        {
            if (source[i+j]!=sub[j])
            {
                break;
            }
        }
        if (j==subLen)
        {
            return i ;
        }
    }
    return -1;
}

```

13、把一个int型数组中的数字拼成一个串，这个串代表的数字最小;

先将数字转换成字符串存在数组中，在通过qsort排序，在排序用到的比较函数中，将要比较的两个字符串进行组合，如要比较的两个字符串分别是 A,B，那么组合成，A+B，和B+A，在比较A+B和B+A，返回strcmp(A+B, B+A)，经过qsort这么一排序,数组就变成从小到大的顺序了，组成的数自然是最小的。

//把一个int型数组中的数字拼成一个串，是这个串代表的数组最小


```
#define MaxLen 10
int Compare(const void* str1,const void* str2)
{
    char cmp1[MaxLen*2+1];
    char cmp2[MaxLen*2+1];
    strcpy(cmp1,(char**)str1);
    strcat(cmp1,(char**)str2);

    strcpy(cmp2,(char**)str2);
    strcat(cmp2,(char**)str1);
    return strcmp(cmp1,cmp2);
}
void GetLinkMin(int a[],int len)
{
    char** str=(char**)new int[len];
    for (int i=0;i<len;i++)
    {
        str[i]=new char[MaxLen+1];
        sprintf(str[i],"%d",a[i]);
    }

    qsort(str,len,sizeof(char*),Compare);
    for (int i=0;i<len;i++)
    {
        cout<<str[i]<<" ";
        delete[] str[i];
    }
    delete[] str;
}
void GetLinkMinTest()
{
    int arr[]={123,132,213,231,321,312};
    GetLinkMin(arr,sizeof(arr)/sizeof(int));
}
```

2015-11-04 16:23



实验楼管理员 (<https://www.shiyanlou.com/user/8490>)  (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

14、输入一颗二叉树，输出它的镜像（每个节点的左右子节点交换位置）；

递归实现，只要某个节点的两个子节点都不为空，就左右交换，让左子树交换，让右子树交换。

```

struct NodeT
{
    int value;
    NodeT* left;
    NodeT* right;
    NodeT(int value_=0, NodeT* left_=NULL, NodeT* right_=NULL):value(value_),left(left_),right(right_){}
};

//输入一颗二叉树，输出它的镜像（每个节点的左右子节点交换位置）
void TreeClass(NodeT* root)
{
    if( root==NULL || (root->left==NULL && root->right==NULL) )
        return;
    NodeT* tmpNode=root->left;
    root->left=root->right;
    root->right=tmpNode;
    TreeClass(root->left);
    TreeClass(root->right);
}

void PrintTree(NodeT* root)
{
    if(root)
    {
        cout<<root->value<<" ";
        PrintTree(root->left);
        PrintTree(root->right);
    }
}

void TreeClassTest()
{
    NodeT* root=new NodeT(8);
    NodeT* n1=new NodeT(6);
    NodeT* n2=new NodeT(10);
    NodeT* n3=new NodeT(5);
    NodeT* n4=new NodeT(7);
    NodeT* n5=new NodeT(9);
    NodeT* n6=new NodeT(11);
    root->left=n1;
    root->right=n2;
    n1->left=n3;
    n1->right=n4;
    n2->left=n5;
    n2->right=n6;
    PrintTree(root);
    cout<<endl;
    TreeClass( root );
    PrintTree(root);
    cout<<endl;
}

```

2015-11-04 16:23



实验楼管理员 (<https://www.shiyanlou.com/user/8490>) 💛 (<https://www.shiyanlou.com/vip>)

(<https://www.shiyanlou.com/user/8490>)

15、输入两个链表，找到它们第一个公共节点；

如果两个链表有公共的节点，那么第一个公共的节点及往后的节点都是公共的。从后往前数N个节点（N=短链表的长度节点个数），长链表先往前走K个节点（K=长链表的节点个数-N），这时两个链表都距离末尾N个节点，现在可以一一比较了，最多比较N次，如果有两个节点相同就是第一个公共节点，否则就没有公共节点。

```

//输入两个链表，找到它们第一个公共节点
int GetLinkLength(NodeL* head)
{
    int count=0;
    while (head)
    {
        head=head->next;
        count++;
    }
}

```

```

    }
    return count;
}

NodeL* FindFirstEqualNode(NodeL* head1, NodeL* head2)
{
    if (head1==NULL || head2==NULL)
        return NULL;
    int len1=GetLinkLength(head1);
    int len2=GetLinkLength(head2);
    NodeL* longNode;
    NodeL* shortNode;
    int leftNodeCount;
    if (len1>len2)
    {
        longNode=head1;
        shortNode=head2;
        leftNodeCount=len1-len2;
    }else{
        longNode=head2;
        shortNode=head1;
        leftNodeCount=len2-len1;
    }
    for (int i=0;i<leftNodeCount;i++)
    {
        longNode=longNode->next;
    }
    while (longNode && shortNode && longNode!=shortNode)
    {
        longNode=longNode->next;
        shortNode=shortNode->next;
    }
    if (longNode)//如果有公共节点，必不为NULL
    {
        return longNode;
    }
    return NULL;
}

void FindFirstEqualNodeTest()
{
    NodeL* head1=new NodeL(0);
    NodeL* head2=new NodeL(0);
    NodeL* node1=new NodeL(1);
    NodeL* node2=new NodeL(2);
    NodeL* node3=new NodeL(3);
    NodeL* node4=new NodeL(4);
    NodeL* node5=new NodeL(5);
    NodeL* node6=new NodeL(6);
    NodeL* node7=new NodeL(7);

    head1->next=node1;
    node1->next=node2;
    node2->next=node3;
    node3->next=node6;//两个链表相交于节点node6

    head2->next=node4;
    node4->next=node5;
    node5->next=node6;//两个链表相交于节点node6
    node6->next=node7;

    NodeL* node= FindFirstEqualNode(head1,head2);
    if (node)
    {
        cout<<node->value<<endl;
    }else{
        cout<<"没有共同节点"<<endl;
    }
}

```

2015-11-04 16:24



(<https://www.shiyanlou.com/user/8490>)

via.zhaoyafei

文章地址: <http://www.zhaoyafei.cn/index.php/Article/articleinfo.html?id=13>
(<http://www.zhaoyafei.cn/index.php/Article/articleinfo.html?id=13>)

2015-11-04 16:24



Desperado_ (<https://www.shiyanlou.com/user/208687>)

(<https://www.shiyanlou.com/user/208687>)

没有java语言的代码实现吗。

2016-05-30 14:40

登录 (<https://www.shiyanlou.com/login?next=/questions/2210>)后回答问题

我要提问

标签

- Linux (<https://www.shiyanlou.com/questions/?tag=Linux>)
- 课程相关 (<https://www.shiyanlou.com/questions/?tag=课程相关>)
- Python (<https://www.shiyanlou.com/questions/?tag=Python>)
- 实验环境 (<https://www.shiyanlou.com/questions/?tag=实验环境>)
- C/C++ (<https://www.shiyanlou.com/questions/?tag=C/C++>)
- 技术分享 (<https://www.shiyanlou.com/questions/?tag=技术分享>)
- 课程需求 (<https://www.shiyanlou.com/questions/?tag=课程需求>)
- 功能建议 (<https://www.shiyanlou.com/questions/?tag=功能建议>)
- Java (<https://www.shiyanlou.com/questions/?tag=Java>)
- 其他 (<https://www.shiyanlou.com/questions/?tag=其他>)
- Web (<https://www.shiyanlou.com/questions/?tag=Web>)
- Hadoop (<https://www.shiyanlou.com/questions/?tag=Hadoop>)
- NodeJS (<https://www.shiyanlou.com/questions/?tag=NodeJS>)
- SQL (<https://www.shiyanlou.com/questions/?tag=SQL>)
- PHP (<https://www.shiyanlou.com/questions/?tag=PHP>)
- Shell (<https://www.shiyanlou.com/questions/?tag=Shell>)
- 常见问题 (<https://www.shiyanlou.com/questions/?tag=常见问题>)
- Git (<https://www.shiyanlou.com/questions/?tag=Git>)
- HTML (<https://www.shiyanlou.com/questions/?tag=HTML>)
- 网络 (<https://www.shiyanlou.com/questions/?tag=网络>)
- HTML5 (<https://www.shiyanlou.com/questions/?tag=HTML5>)
- 信息安全 (<https://www.shiyanlou.com/questions/?tag=信息安全>)
- Android (<https://www.shiyanlou.com/questions/?tag=Android>)
- GO (<https://www.shiyanlou.com/questions/?tag=GO>)
- NoSQL (<https://www.shiyanlou.com/questions/?tag=NoSQL>)
- Ruby (<https://www.shiyanlou.com/questions/?tag=Ruby>)
- 训练营 (<https://www.shiyanlou.com/questions/?tag=训练营>)
- Perl (<https://www.shiyanlou.com/questions/?tag=Perl>)



实验楼客户端

即开即用



会员专属

(<https://www.shiyanlou.com/vip>)

相关问题

C/C++的mem函数和strcpy函数的区别和应用 (<https://www.shiyanlou.com/questions/5274>)

python之线程、进程和协程 (<https://www.shiyanlou.com/questions/5107>)

从底层理解Python的执行 (<https://www.shiyanlou.com/questions/5247>)

20个为前端开发者准备的文档和指南（2） (<https://www.shiyanlou.com/questions/5215>)

震惊小伙伴的单行代码—Python篇 (<https://www.shiyanlou.com/questions/4157>)

九个Console命令，让js调试更简单 (<https://www.shiyanlou.com/questions/5178>)

10款最佳PHP自动化测试框架 (<https://www.shiyanlou.com/questions/2211>)

Git 远程操作的正确姿势 (<https://www.shiyanlou.com/questions/5134>)

JavaScript异步编程解决方案笔记 (<https://www.shiyanlou.com/questions/5094>)

Vim 起步的五个技巧 (<https://www.shiyanlou.com/questions/5072>)



动手做实验，轻松学IT。



公司

关于我们 (<https://www.shiyanlou.com/aboutus>)

联系我们 (<https://www.shiyanlou.com/contact>)

加入我们 (<http://www.simplecloud.cn/jobs.html>)

技术博客 (<https://blog.shiyanlou.com/>)

服务

实战训练营 (<https://www.shiyanlou.com/bootcamp/>)

会员服务 (<https://www.shiyanlou.com/vip>)

(<http://weibo.com/shiyanlou2013>)
合作

我要投稿 (<https://www.shiyanlou.com/contribute>)

教师合作 (<https://www.shiyanlou.com/labs>)

高校合作 (<https://www.shiyanlou.com/edu/>)

友情链接 (<https://www.shiyanlou.com/friends>)

学习路径

Python学习路径 (<https://www.shiyanlou.com/paths/python>)

Linux学习路径 (<https://www.shiyanlou.com/paths/linuxdev>)

[实验报告 \(https://www.shiyanlou.com/courses/reports\)](https://www.shiyanlou.com/courses/reports)

[常见问题 \(https://www.shiyanlou.com/questions/?tag=常见问题\)](https://www.shiyanlou.com/questions/?tag=常见问题)

[隐私条款 \(https://www.shiyanlou.com/privacy\)](https://www.shiyanlou.com/privacy)

[大数据学习路径 \(https://www.shiyanlou.com/paths/bigdata\)](https://www.shiyanlou.com/paths/bigdata)

[Java学习路径 \(https://www.shiyanlou.com/paths/java\)](https://www.shiyanlou.com/paths/java)

[PHP学习路径 \(https://www.shiyanlou.com/paths/php\)](https://www.shiyanlou.com/paths/php)

[全部 \(https://www.shiyanlou.com/paths/\)](https://www.shiyanlou.com/paths/)

Copyright @2013-2016 实验楼在线教育

蜀ICP备13019762号 (<http://www.miibeian.gov.cn/>) 站长统计 (http://www.cnzz.com/stat/website.php?web_id=5902315)