浙江财经大学
Zhejiang University Of Finance & Economics

2014初赛讲解

信智学院　陈琰宏

1 单项选择题

1.以下哪个是面向对象的高级语言(B)。

A.汇编语言　B.C++　C.Fortran　D.Basic

2.设有100个数据元素，采用折半搜索时，最大比较次数为(B)。

A.6　B.7　C.8　D.10

$2^7>100$

3.二进制数00100100和00010101的和是(D)。

A.00101000　B.001010100　C.01000101　D.00111001

00100100

00010101

00111001

4.一棵具有5层的满二叉树中结点数为(A)。

A.31　B.32　C.33　D.16

2^n-1

1 单项选择题

5.下列对操作系统功能的描述最为完整的是(　　C　　)。

A. 负责外设与主机之间的信息交换

B. 负责诊断机器的故障

C. 控制和管理计算机系统的各种硬件和软件资源的使用

D. 将没有程序编译成目标程序

6.把8个同样的球放到5个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的放置方法？(用K表示)。

A. 56　　B. 28　　C. 18　　D. 24

枚举

0 0 0 0 8　　0 0 0 4 4　　0 0 2 2 4　　0 1 1 3 3　　1 1 1 2 3

0 0 0 1 7　　0 0 1 1 6　　0 0 2 3 3　　0 1 2 2 3　　1 1 2 2 2

0 0 0 2 6　　0 0 1 2 5　　0 1 1 1 5　　0 2 2 2 2

0 0 0 3 5　　0 0 1 3 4　　0 1 1 2 4　　1 1 1 1 4

1 单项选择题

7.断电后会丢失数据的存储器是(A)。

A. RAM　　B. ROM　　C. 硬盘　　D. 光盘

8.有向图中每个顶点的度等于该顶点的(C)。

A. 入度　　B. 出度

C. 入度和出度之和　　D. 入度和出度之差

9.若有如下程序段，其中s、a、b、c均已定义为整型变量，且a、c均已赋值，c>0。 s = a;

for(b = 1; b <= c; b++)　　s += 1;

则与上述程序段功能等价的赋值语句是(B)。

A. s = a + b　　B. s = a + c

C. s = s + c　　D. s = b + c

for循环b从1~c,实现减去s+c,s的初始值为a, 所以实现 a+c

1 单项选择题

10.链表不具有的特点是(　　B　)。

A.不必事物估计存储空间　　B.可随机访问任一元素

C.插入删除不需要移动元素　　D.所需空间与线性表长度成正比

11.下列各无符号十进制整数中，能用八位二进制表示的数中最大的是(　D　　)。

A.296　　B.133　　C.256　　D.199

0~255

12.下列几个32位IP地址中，书写错误的是(　　C　)。

A.162.105.135.27　　B.192.168.0.1

C.256.256.129.1　　D.10.0.0.1

每一部分用8位二进制表示

1 单项选择题

13.要求以下程序的功能是计算：s=1+1/2+1/3+...+1/10。

```
#include <iostream>
using namespace std;
int main() {
        int n;
        float s;
        s = 1.0;
        for(n = 10; n > 1; n--)
                s = s + 1 / n;
        cout << s << endl;
        return 0;
}
```

程序运行后输出结果错误，导致错误结果的程序行是(　C　)。

A. s = 1.0;　　B. for(n = 10; n > 1; n--)

C. s = s + 1 / n;　　D. cout << s << endl;

1 单项选择题

14.设变量x为float型且已赋值，则以下语句中能将x中的数值保留到小数点后两位，并将第三位四舍五入的是(　　C　　)。

A. x = (x * 100) + 0.5 / 100.0;

B. x = (x * 100 + 0.5) / 100.0;

C. x = (int)(x * 100 + 0.5)/100.0;

D. x = (x / 100 + 0.5) * 100.0;

B:3.146*100+0.5=315.1/100=3.151

C:315/100.0=3.15

1 单项选择题

15.有以下程序

```
#include <iostream> using namespace std;
int main() {
int s, a, n;
s = 0;
a = 1;
cin >> n;
do {
        s += 1;
        a -= 2;
} while(a != n);
cout << s << endl;
return 0;
}
```

若要使程序的输出值为2，则应该从键盘给n输入的值（ B ）。

A.-1　　B.-3　　C.-5　　D.0

2 阅读程序写结果1

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main(){ //题目大意：将字符串中的小写字母变为大写字母
5      string st;
6      int i, len;
7      getline(cin, st);
8      len = st.size();
9      for(i = 0; i < len; i++){//判断是否为小写字母
10         if(st[i] >= 'a' && st[i] <= 'z')
11             st[i] = st[i] - 'a' + 'A';
12     }          //是小写字母转化为大写字母
13     cout << st << endl;
14     return 0;
15 }
```

2 阅读程序写结果1

判断题

(1)输入的字符串可以是任意字符,包括字母、数字、各类符号甚至中文汉字。（　　T　　）　题目没有要求，字符都可以。

(2)如果去掉第10行,输出结果不变。　　（　　F　　）

去掉以后，很多字母都发生了变化。

(3)输出结果可以包含小写字母。　　（　　F　　）

小写字母都变成了大写

(4算法时间复杂度为O(1)。　　（　　F　　）

时间复杂的为O(n)

选择题

(5)输入Hello,输出的结果是(　　A　　)。

A. HELLO　　B. hello　　C. Hello　　D. ello

(6)输出的结果不可能是(　　C　　)。

A.WELCOME　　B.WELCOME-1　　C. Welcome　　D.ELCOME

不可能出现小写字母

2 阅读程序写结果2

```
#include <iostream>
using namespace std;
const int SIZE = 1e5;
int main(){    //题目的含义：寻找2~~n以内的质数
    int p[SIZE];
    int n, tot, i, cn;
    tot = 0;
    cin >> n;   //初始为1，表示是素数
    for(i = 1; i <= n; i++)p[i] = 1;
    for(i = 2; i <= n; i++){
        if(p[i] == 1)tot++;//i是素数，累加
        cn = i * 2;//用i进行筛选，
        while(cn <= n){
          p[cn] = 0; // i的倍数标记为0
          cn += i; // 表示不是素数
        }
    }
    cout << tot << endl;
    return 0;
}
```

●判断题

(1)n的值为100时程序不会运行错误。 (F)

越界

(2)该程序的时间复杂度为O(n)。 (F)

筛选法的时间复杂度为O(nlogn)

(3)将第12行的cn=2*i改成cn=i，程序输出结果不变。(T)

11行以后得代码都不会访问p[i]所以没有影响

(4)输入30，输出的结果为10。 (T)

1 3 5 7 11 13 17 19 23 29

●选择题

(5)如果n=1000000（10^6）,与第15行的赋值运算执行的次数最接近的是（ D ）埃氏筛选法的时间复杂度为O(nlogn)

A.10^6 B.10^9 C.$5*10^6$ D.10^7

(6)输入80，则输出结果为(B)。

A. 24 B.22 C.25 D.23

2 阅读程序3[6556]

此程序实现了输入字符串的冒泡排序，并按输入顺序输出每个字符串按照字典序从小到大排好序中的位置。
由于冒泡排序是稳定的，因此相同的字符串应该保持输入的顺序。

```
4  const int SIZE=100;
5  int main( ){
6      string dict[SIZE];
7      int rank[SIZE];
8      int ind[SIZE];
9      int i, j,n, tmp;
10     cin>>n;
11     for(i=1;i<=n;i++){
12         rank[i]=i;  //第12行
13         ind[i]=i;
14         cin>>dict[i];
15     }
16     for(i=1;i<n;i++)//冒泡排序
17         for(j=1;j<=n-i;j++)
18             if (dict[ind[j]]>dict[ind[j+1]]){
19                 tmp=ind[j];
20                 ind[j]=ind[j+1];
21                 ind[j+1]=tmp;
22             }
23     for(i=1;i<=n;i++)
24         rank[ind[i]]=i;
25     for(i=1;i<=n;i++)
26         cout<<rank[i]<<" ";
27     cout<<endl;
28     return 0;
29 }
```

3 阅读程序写结果3

(1)该程序的本质是按字典序对字符串排序。(F)

叙述不准确，按照字典序排序，然后输出的是原字符串在排序后的位置。

(2)如果输入0,没有任何输出。 (F)

27行有换行符输出

(3)把第六行的string 改成char ， 对程序没有影响 (F)

(4)如果去掉第12行,不影响。 (T)

24行对rank[]重复赋值，没有影响

●选择题

(5)输入7naaa\naba\nbbb\naaa\naaa\ncce\naa,输出(A)。

A.2 5 6 3 4 7 1 B.1 2 3 4 5 6 7

C.1 7 4 3 6 5 2 D.7 6 5 4 3 2 1

(6)这个程序使用的是什么排序(D).

A.插入排序 B.希尔排序

C.选择排序 D.冒泡排序

完善程序1[6557]：最大子矩阵和

1.(最大子矩阵和)给出m行n列的整数矩阵，求最大的子矩阵和(子矩阵不能为空)。
输入第一行包含两个整数m和n，即矩阵的行数和列数。之后m行，每行n个整数，描述整个矩阵。程序最终输出最大的子矩阵和。 比如在如下这个矩阵中：

```
 4  4
0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
```

拥有最大和的子矩阵为：

```
 9 2
-4 1
-1 8
```

其和为15

```
3  3
-2 10 20
-1 100 -2
0 -2 -3
```

最大子矩阵和为128

```
4  4
0 -2 -9 -9
-9 11 5 7
-4 -3 -7 -6
-1  7  7  5
```

最大子矩阵和为26

```
3  const int SIZE = 100;
4  int matrix[SIZE + 1][SIZE + 1];
5  int rowsum[SIZE + 1][SIZE + 1]; //rowsum[i][j]记录第i行前j个数的和
6  int m, n, i, j, first, last, area, ans;
7  int main()
8  {
9      cin >> m >> n;
10     for(i = 1; i <= m; i++)
11         for(j = 1; j <= n; j++)
12             cin >> matrix[i][j];
13     ans = matrix____1____;
14     //ans=matrix[1][1];  给定一个初始值为后面作比较定一个标准
15     for(i = 1; i <= m; i ++) //初始化rowsum[][]矩阵
16         ____2____
17         for(i = 1; i <= m; i++)//二层for循环用来求出第i行前j数的和
18             for(j = 1; j <= n; j++)
19                 rowsum[i][j] =____3____;
```

```
    for(first = 1; first <= n; first++)//开始遍历
        for(last = first; last <= n; last++)
        {
            ____4_____      //初始化，避免每次循环area赋值后带来的影响
    for(i = 1; i <= m; i++)          //寻找最大子矩阵和
    {
        area +=____5_____;
        if(area > ans)
            ans = area;    //记录当前已知的最大子矩阵和
            if(area < 0)                    //如果area<0，舍弃这一行，重新寻找
            area = 0;
        }
    }
    cout << ans << endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
const int SIZE = 100;
int matrix[SIZE + 1][SIZE + 1];
int rowsum[SIZE + 1][SIZE + 1]; //rowsum[i][j]记录第i行前j个数的和
int m, n, i, j, first, last, area, ans;
int main()
{
    cin >> m >> n;
    for(i = 1; i <= m; i++)
        for(j = 1; j <= n; j++)
            cin >> matrix[i][j];
    ans = matrix[1][1];
    //ans=matrix[1][1];  给定一个初始值为后面作比较定一个标准
    for(i = 1; i <= m; i ++)       //初始化rowsum[][]矩阵
        rowsum[i][0]=0;
        for(i = 1; i <= m; i++)
         //二层for循环用来求出第i行前j数的和
            for(j = 1; j <= n; j++)
                rowsum[i][j] = rowsum[i][j-1]+matrix[i][j];
```

```
21      for(first = 1; first <= n; first++)          //开始遍历
22          for(last = first; last <= n; last++)
23          {
24              area=0;/初始化，避免每次循环area赋值后带来的影响
25              for(i = 1; i <= m; i++) //寻找最大子矩阵和
26              {
27                  area +=rowsum[i][last]-rowsum[i][first-1];
28                  if(area > ans)
29                      ans = area;    //记录当前已知的最大子矩阵和
30                  if(area < 0) //如果area<0，舍弃这一行，重新寻找
31                      area = 0;
32              }
33          }
34      cout << ans << endl;
35      return 0;
36  }
```

(1)①处应填(D)

A.[0][0]　　B.[0][n]

C.[m][0]　　D.[1][1]

(2)②处应填(C)。

A. rowsum[i][0]=matrixi[i][1]　　B. rowsum[i][0]=1

C. rowsum[i][0]=0　　D.rowsum[i][0]=ans

(3)3处应填(D)。

A. rowsum[i][j-1]　　B. rowsum[i-1][j]+matrix[i][j]

C. matrix[i][j]　　D.rowsum[i][j-1]+matrix[i[j]

(4)4处应填(A)。

A. area=0　　B.ans=area

C. area=ans　　D.ans=0

(5)⑤处应填(B)。

A. matrix[i][last]　　B. rowsum[i][last]-rowsum[i][first-1]

C. matrix[i][fist]　　D. rowsum[i][last]-rowsum[i][first]

3 完善程序2[6558]

双栈模拟数组）只使用两个栈结构 stack1 和stack2，模拟对数组的随机读取。作为栈结构， stack1 和 stack2 只能访问栈顶（最后一个有效元素）。栈顶指针 top1 和top2 均指向栈顶元素的下一个位置。
输入第一行包含的两个整数，分别是数组长度 *n* 和访问次数 *m*，中间用单个空格隔开。
第二行包含 *n* 个整数，一次给出数组各项（数组下标从 0 到 *a*–1）。
第三行包含 *m* 个整数，需要访问的数组下标。对于每次访问，输出对应的数组元素。

3 完善程序

只使用两个栈结构 stack1 和stack2，模拟对数组的随机读取。作为栈结构，stack1 和 stack2 只能访问栈顶（最后一个有效元素）。
栈顶指针 top1 和top2 均指向栈顶元素的下一个位置。访问第i个元素时，stack1从栈底到栈顶一次存放第1到第i个元素，而stack2从栈底到栈顶依次存放第n到第（i+1)个元素

```
 2 const int   SIZE = 100;
 3 int stack1[SIZE], stack2[SIZE];
 4 int top1, top2;
 5 int n, m, i, j;
 6 void clearStack()
 7 {
 8     int i;
 9     for ( i = top1; i < SIZE; i++ )
10         stack1[i] = 0;
11     for ( i = top2; i < SIZE; i++ )
12         stack2[i] = 0;
13 }
14 int main( )
15 {
16     scanf("%d%d", &n, &m);
17     for ( i = 0; i < n; i++ )
18         scanf("%d", &stack1[i]);
19     top1 = ___1____;//1 初始数据存在stack1，n-1的下一个元素为n
20     top2 = ___2____;//2 st2初始为空即-1 ， 下一个位置为0
```

```
21      for ( j = 0; j < m; j++ )
22      {
23          scanf("%d",&i);
24          while ( i < top1 - 1 )
25          {//下一次访问的元素在i之前时，
26          //通过将st1的栈顶元素取出放到st2,实现“ 指针的向前移动”
27              top1--;//从st1拷贝元素到st2 形成0~i的元素在st1
28              ___3____;//3
29              top2++;//
30          }
31          while (i > top1 - 1)
32          {//下一次访问的元素在i之后时，
33          //通过将st2的栈顶元素取出放到st1,实现“ 指针的向后移动”
34              top2--;
35              ___4____;
36              top1++;
37          }
38          clearStack();
39          printf("%d\n", stack1[___5____;]);//st1的栈顶元素为答案
40      }
41      return(0);
42  }
```

```
 2  const int   SIZE = 100;
 3  int stack1[SIZE], stack2[SIZE];
 4  int top1, top2;
 5  int n, m, i, j;
 6  void clearStack()
 7  {
 8      int i;
 9      for ( i = top1; i < SIZE; i++ )
10          stack1[i] = 0;
11      for ( i = top2; i < SIZE; i++ )
12          stack2[i] = 0;
13  }
14  int main( )
15  {
16      scanf("%d%d", &n, &m);
17      for ( i = 0; i < n; i++ )
18          scanf("%d", &stack1[i]);
19      top1 = n;//1 初始数据存在stack1，n-1的下一个元素为n
20      top2 = 0;//2 st2初始为空即-1 ， 下一个位置为0
```

```
21      for ( j = 0; j < m; j++ )
22      {
23          scanf("%d",&i);
24          while ( i < top1 - 1 )
25          {//下一次访问的元素在i之前时，
26          //通过将st1的栈顶元素取出放到st2,实现“ 指针的向前移动”
27              top1--;//从st1拷贝元素到st2 形成0~i的元素在st1
28              stack2[top2]=stack1[top1];//3
29              top2++;//
30          }
31          while (i > top1 - 1)
32          {//下一次访问的元素在i之后时，
33          //通过将st2的栈顶元素取出放到st1,实现“ 指针的向后移动”
34              top2--;
35              stack1[top1]=stack2[top2];
36              top1++;
37          }
38          clearStack();
39          printf("%d\n", stack1[top1-1]);//st1的栈顶元素为答案
40      }
41      return(0);
42  }
```

●选择题

(1)①处应填(B)。

A.0 B.n C.1 D.-1

(2)2处应填(D).

A.1 B.2 C.n D.0

(3)③处应填(A)。

A. stack2[top2]=stack1[top1] B.top1=top2

C. stack2[top1]=stack1[top1] D.stack2[top1]=stack1[top2]

(4)4处应填(C)。

A.stack1[top1]=stack2[top1] B.stack1[top2]=stack2[top1]

C.stack1[top1]=stack2[top2] D.stack1[top2]=stack2[top2]

(5)⑤处应填(C)。

A. top1 B.top2 C.top1-1 D.top2-1

今天的课程结束啦……

下课了…
同学们再见！