**全国计算机技术与软件专业技术资格（水平）考试**

**2016年下半年 程序员 下午试卷**

（考试时间 14:00～16:30 共 150 分钟）

|  |
| --- |
| **请按下述要求正确填写答题纸** |

1.在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。

2.在答题纸的指定位置填写准考证号、出生年月日和姓名。

3.答题纸上除填写上述内容外只能写解答。

4.本试卷共6道题，试题一至试题四是必答题，试题五至试题六选答 1 道。每

题 15 分，满分 75 分。

5.解答时字迹务必清楚，字迹不清时，将不评分。

6.仿照下面例题，将解答写在答题纸的对应栏内。

**例题**

2016 年下半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）

月（2）日。

因为正确的解答是“11 月 4 日”，故在答题纸的对应栏内写上“11”和“4”

（参看下表）。

|  |  |
| --- | --- |
| 例题 | 解答栏 |
| （1） | 11 |
| （2） | 4 |

**试题一（共15分）**

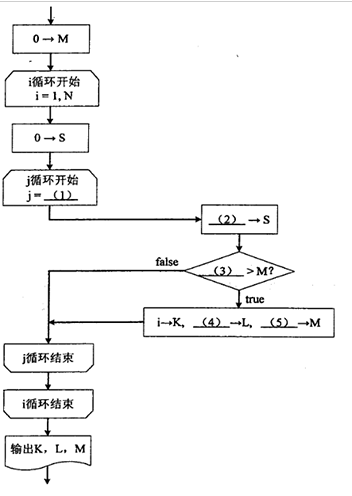
阅读以下说明和流程图，填补流程图中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

设有整数数组A[1：N]（N>1），其元素有正有负。下面的流程图在该数组中寻找连续排列的若干个元素，使其和达到最大值，并输出其起始下标K、元素个数L以及最大的和值M。

例如，若数组元素依次为3，-6，2，4，-2，3，-1，则输出K=3，L=4,M=7。该流程图中考察了A[1：N]中所有从下标i到下标j（j≥i）的各元素之和S，并动态地记录其最大值M。

**【流程图】**



注：循环开始框内应给出循环控制变量的初值和终值，默认递增值为1，格式为：循环控制变量=初值，终值

**试题二（共15分）**

阅读以下代码，回答问题：1至问题3 ，将解答填入答题纸的对应栏内。

**【代码1】**

#include<stdio.h >

void swap(int x, int y)

{

int tmp =x; x= y; y= tmp;

}

int maim()

{

int a= 3, b= 7;

printf("a1= %d b1=%d\n",a,b);

Swap( a, b);

Printf("a2 = %d b2=%d\n”,a,b)；

return 0;

}

**【代码2】**

#include<stdio.h>

#define SPACE ¨ //空格字符

Int main()

{

char str[128] =”Nothing is impossible! “;

int i,num =0,wordMark=0;

for(i=0;str[i];i++)

If(str[i]=SPACE)

WordMark=0;

else

If(wordMark=0){

wordMark=1;

Mun++;

}

Printf(“%d/n”,num)

retun 0;

}

**【代码3】**

#include<stdio.h>

#define SPACE “//空格字符

int countStrs(char \*);

int main()

{

char str[128] = " Nothing is impossible! ";

Printf(‘%d/n,（1）(str))

retum 0;

}

int countStrs(char \*p)

{

int num=0, wordMark= 0;

for(;（2）；p++) {

If(（3）=SPACE)

wordMark= 0;

else

if( !wordMark ) {

wordMark = 1;

++num

}

}

return（4）

}

**【问题1】（4分）**

写出代码1运行后的输出结果。

**【问题2】（3分）**

写出代码2运行后的输出结果。

**【问题3】（8分）**

代码3的功能与代码2完全相同，请补充3中的空缺，将解答写入答题纸的对应栏内。

**试题三（共15分）**

阅读以下说明和代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

下面的程序利用快速排序中划分的思想在整数序列中找出第k小的元素（即将元素从小到大排序后，取第k个元素）。

对一个整数序列进行快速排序的方法是：在待排序的整数序列中取第一个数作为基准值，然后根据基准值进行划分，从而将待排序的序列划分为不大于基准值者（称为左子序列）和大于基准值者（称为右子序列），然后再对左子序列和右子序列分别进行快速排序，最终得到非递减的有序序列。

例如，整数序列“19, 12, 30, 11,7,53, 78, 25"的第3小元素为12。整数序列“19, 12,7,30, 11, 11,7，53. 78, 25, 7"的第3小元素为7。

函数partition（int a[], int low,int high）以a[low]的值为基准，对a[low]、a[low+l]、…、

a[high]进行划分，最后将该基准值放入a[i] (low≤i≤high)，并使得a[low]、a[low+l]、，．．、

A[i-1]都小于或等于a[i]，而a[i+l]、a[i+2]、．．、a[high]都大于a[i]。

函教findkthElem(int a[],int startIdx,int endIdx,inr k)在a[startIdx]、a[startIdx+1]、...、a[endIdx]中找出第k小的元素。

【代码】

#include <stdio.h>

#include <stdlib.h>

Int partition（int a []，int low, int high）

{//对 a[low..high]进行划分，使得a[low..i]中的元素都不大于a[i+1..high]中的元素。

int pivot=a[low]； //pivot表示基准元素

Int i=low,j=high;

while(（1）){

While(i<j&&a[j]>pivot)--j;

a[i]=a[j]

While(i<j&&a[i]>pivot)++i;

a[j]=a[i]

}

（2）； //基准元素定位

return i；

}

Int findkthElem（int a[]，int startIdx，int endIdx, int k）

{//整数序列存储在a[startldx..endldx]中，查找并返回第k小的元素。

if (startldx<0 ||endIdx<0 || startIdx>endIdx || k<1 ||k-l>endIdx ||k-1<startIdx)

Return-1; //参数错误

if(startIdx<endldx){

int loc=partition(a, startIdx, endldx)； ∥进行划分，确定基准元素的位置

if (loc==k-1) ∥找到第k小的元素

return （3） ；

if(k-l <loc) //继续在基准元素之前查找

return findkthElem(a， （4） ，k）；

else ／／继续在基准元素之后查找

return findkthElem(a， （5） ，k);

}

return a[startIdx]；

}

int main()

{

int i, k;

int n;

int a[] = {19, 12, 7, 30, 11, 11, 7, 53, 78, 25, 7};

n= sizeof(a)／sizeof(int) //计算序列中的元素个数

for (k=1;k＜n+1；k++){

for(i=0;i＜n;i++){

printf(“%d/t”,a[i]);

}

printf(“\n”);

printf(“elem %d=%d\n,k,findkthElem(a,0,n-1,k));//输出序列中第k小的元素

}

return 0；

}

**试题四**

阅读以下说明和代码，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

图是很多领域中的数据模型，遍历是图的一种基本运算。从图中某顶点v出发进行广度优先遍历的过程是：

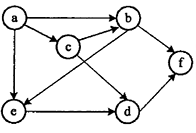
①访问顶点v；

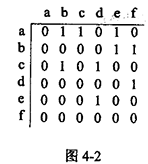
②访问V的所有未被访问的邻接顶点W1 ,W2 ,..,Wk；

③依次从这些邻接顶点W1 ,W2 ,..,Wk出发，访问其所有未被访问的邻接顶点；依此类推，直到图中所有访问过的顶点的邻接顶点都得到访问。

显然，上述过程可以访问到从顶点V出发且有路径可达的所有顶点。对于从v出发不可达的顶点u，可从顶点u出发再次重复以上过程,直到图中所有顶点都被访问到。

例如，对于图4-1所示的有向图G，从a出发进行广度优先遍历，访问顶点的一种顺序为a、b、c、e、f、d。





设图G采用数组表示法（即用邻接矩阵arcs存储），元素arcs[i][j]定义如下：

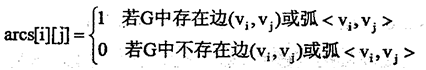


图4-1的邻接矩阵如图4-2所示，顶点a~f对应的编号依次为0~5.因此，访问顶点a的邻接顶点的顺序为b，c，e。

函数BFSTraverse(Graph G)利用队列实现图G的广度优先遍历。

相关的符号和类型定义如下：

#define MaxN：50 ／\*图中最多顶点数\*／

typedef int AdjMatrix[MaxN][MaxN]；

typedef struct{

int vexnum，edgenum; ／\*图中实际顶点数和边（弧）数\*／

AdjMatrix arcs； ／\*邻接矩阵\*／

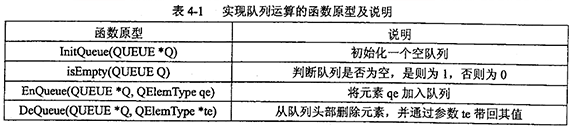
)Graph；

typedef int QElemType；

enum {ERROR=0;OK=l};

代码中用到的队列运算的函数原型如表4-1所述，队列类型名为QUEUE。

表4-1 实现队列运算的函数原型及说明



**【代码】**

int BFSTraverse(Graph G)

{//图G进行广度优先遍历，图采用邻接矩阵存储

unsigned char\*visited； //visited[]用于存储图G中各顶点的访问标志，0表示未访问

int v，w；u；

QUEUEQ Q;

∥申请存储顶点访问标志的空间，成功时将所申请空间初始化为0

visited=(char\*)calloc(G.vexnum, sizeof(char));

If(（1）)

retum ERROR;

（2） ； //初始化Q为空队列

for( v=0; v<G.vexnum; v++){

if(!visited[v]){ //从顶点v出发进行广度优先遍历

printf("%d”，v)； //访问顶点v并将其加入队列

visited[v]=l；

（3） ；

while(！isEmpty(Q)){

（4） ； //出队列并用u表示出队的元素

for(v=0;v<G.vexnum; w++){

if（G.arcs[u][w]!=0&& （5） ){ //w是u的邻接顶点且未访问过

printf("%d”，w)； //访问顶点w

visited[w]=1；

EnQueue(&Q, w);

}

}

}

}

free(visited);

return OK;

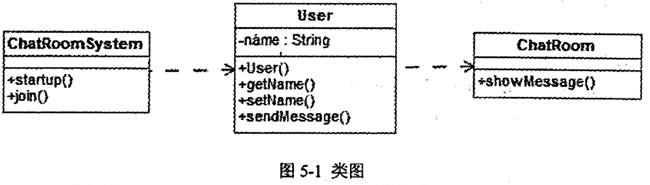
)//BFSTraverse

**试题五**

阅读以下说明和Java程序，填补代码中的空缺，将解答填入答题纸的对应栏内。

**【说明】**

以下Java代码实现一个简单的聊天室系统（ChatRoomSystem），多个用户(User)可以向聊天室( ChatRoom)发送消息，聊天室将消息展示给所有用户。类图如图5-1所示。



**【Java代码】**

class ChatRoom {

public static void showMessage(User user, Strmg message) {

System.out.println("[" + user.getName() + "] : " + message);

}

}

classUser{

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public User(String name) {

（1） =name；

}

public void sendMessage(String message) {

（2） (this, message);

}

}

public class Chat:RoomSystem {

public void startup() {

User zhang= new User("John");

User li =new User("Leo");

zhang.sendMessage("Hi! Leo! ");

1i.sendMessage("Hi! John!");

}

public void join(User user) {

（3） ("Hello Everyone! I am" + user.getName());

}

public static void main(String[] args) {

ChatRoomSystem crs= （4） ;

Crs.startup();

Crs.join( （5） )(“Wayne”))；

}

}

/\*

程序运行结果：

[John]：Hi! Leol

[Leo]：Hi! John!

[Wayne】：Hello Everyone!Iam Wayne

\*/