密 封 线

成绩_____ 学号_____ 姓名_____ 班级_____ 院系_____

2014 级信息学院《C 语言程序设计》考试试题（A）

一、判断下列语句或程序段的对错。（“×”表示错，“√”表示对）（10 分）

(1) int x=0,y=x; (√)

(2) #define SIZE 10
int a[SIZE /2][SIZE]; (√)

(3) char *str；
str ="c\test"； (√)

(4) int a=2,*p; *p=a; //试卷上两处逗号像圆点，建议对错都给分
scanf(“%d”,*p); (√×)

(5) int x = （0x41 = = ‘A’）; (√)

(6) int (*pa)[8],a[6][8];
pa=a+1; (√)

(7) char str[]={“World”};
printf(“%s”,(str+2)); (√)

(8) int **pp,a[6][8];
pp=a; (×)

(9) int a = 5, b = 7, c;
执行 c=a+++b 后 a、b、c 的值分别为 6,7,12 (√)

(10) int x =‘\x123’; 16进制\xhh (×)

二、计算下列表达式的值（10 分）

假设 int 和 unsigned 类型均为 16 位长度，且各题彼此独立
设 unsigned a=8, b=1, c=5, d=7; double f;

(1) f =d%b; (0.0)

(2) a<<13||--b&&c; (0)

(3) (a&b)^(~c | d) (65535)

(4) a += b %= a+b; (9)

(5) ++c，c-d； (65535)

三、程序改错（10 分）

要求：不得改变程序框架，不得重写程序，无需文字说明，直接在代码上添加、删除和修改。

(1) 下面代码实现查找数组中是否存在某数，如果存在，指出其位置，否则提示未找到

```
#include <stdio.h>

typedef struct
{
    long num;
    short location;
} _1;
```

新增原型申明 void bubblesort(_1 data[ ],int n);----------0.5 分

新增原型申明 int search(_1 data[ ],int size,long SearchData);---- 0.5 分

```
int main(       )
{
      _1 data[6]={    {123,1},{22,2},{8976,3},{345,4},{98,5},{100,6}       };
      int i,j;
      bubblesort(data[6]);——>修改为 bubblesort(data,6);---------1 分

      long SearchData;
      puts("Pls enter a data for searching:");
      scanf("%d",SearchData);——>修改为 scanf("%ld",&SearchData);---0.5 分

      int found =-1;
      found =search(data,6,SearchData);
      if(found == -1)
                  puts("Sorry,what you are looking for is not found!");
      else        printf("what is your looking for lies in %hd\n",found );
      新增  return 0；-------------------------------------------------------------0.5 分
}

void bubblesort(_1 data[ ],int n)
{
      int i,j;

      for(i=1;i<=n-1;i++)
            for(j=1;j<=n-i; j++)
                  if(data[j-1].num > data[j].num)
                  {
                        _1    *temp;           ——>修改为_1    temp;
                        *temp       =data[j];——>修改为 temp       =data[j];
                        data[j]    =data[j-1];
                        data[j-1] = *temp;   ——>修改为 data[j-1] = temp;---3 处共 1 分
                  }
}

int search(_1 data[ ], int size, long SearchData)
{
      int left=0,right=size-1,middle;
      while(left<=right)
      {
            middle =(left + right)/2;
```

```
            if(data[middle].num == SearchData)
                continue;                       ——>修改为 break；--------0.5
            else if(data[middle].num < SearchData)
                    left =middle ++;            ——>修改为 left =middle +1；
            else    right =middle--;            ——>修改为 right =middle-1；--2 处共 0.5
        }

        if(left>right)   return -1;
        else        return data[middle].location;
    }
```

(2) 计算 30 个学生 C 语言成绩的最高分和最低分

```
#include <stdio.h>
新增原型申明 float statistic (float *p1 , int n , float* p2);----------0.5 分
void main( void )
{
    float a[30] , min ;
    int m;
    新增 float max;-----------------------------------------------------------------0.5 分

    for(m=0;m<30;m++) ;          ——>修改为 for(m=0;m<30;m++) ----0.5 分
        scanf("%f",&a[m]);
    新增 min=a[0];              //或者在 statistic 函数内部增加（*）语句，二者选一-----0.5
    max =statistic(a,30,min);       ——>修改为 max =statistic(a,30, &min);--1 分
    printf("max=%f , min=%f\n",max,min);
}

statistic (float *p1 , int n , int p2)——>修改为 float statistic (float *p1 , int n , float* p2)—0.5
{
    char i;
    int temp;       ——>修改为 float temp=p1[0]或*p1；---0.5 分
    新增*p2=p1[0];或*p2= *p1;       //（*）

    for(i=1 ; i<n ; i++)
    {
        if(p1[i]>temp)
            temp =p1[i];
        if(p1[i]<p2)            ——>修改为 if(p1[i]< *p2)-----0.5 分
            p2 = p1[i];         ——>修改为*p2 = p1[i];---------0.5 分
    }
     return temp;
}
```

四、程序填空（10 分）------每空 1 分

(1) 找出字符串 str 中连续数字字符子串，每个数字字符子串转换成一个整型数据分别存入数组 a。

```
#include <string.h>
#include <stdio.h>
void main( )
{
    char    ch;
    unsigned int i,j,bit,dit,n;
    long int a[20];
    char *str = "a123x456__789";

    for (i=0,j=0,a[0]=0,bit=1,dit=0;__i<strlen(str) or str[i]!='\0' or str[i]__ ;i++)
    {
        ch=*(str+i);
        if(____ch>='0'&&ch<='9'_____)//不能用 isdigit（ch）因为没有包含 ctype.h
        {
              a[j]*=bit;
              a[j]+=_(ch-'0')____;//或写 ch-48 也算对
              _bit=10;_____
              dit=1;
              n=j;
        }
        else
        {
             if (dit == 1)
             {      j++;       a[j]=0 ;     }
             dit=0;
             bit=1;
        }
    }

    for (i=0;i<=n;i++)
         printf("a[%d]=%ld,", i, _*(a+i) or a[i]_____);
}
```

(2) 以下程序用二维数组保存多个字符串，用二级指针处理多个字符串的排序。

```
#include <stdio.h>
#include <string.h>
void        sortstr(char **v , int n);

void        main(void)
```

```
{
    int     i;
    char    string[ ][20] ={ "pascal", "basic", "cobol", "prolog", "lisp" };
    char *pstr[ ]={string[0],string[1],string[2], string[3],string[4]} ;
    char **ppstr=pstr;
    sortstr(    ppstr, 5 );
    for (i = 0; i < 5; i++)
        printf("%s\n", ppstr[i]);
}

void        sortstr(char **v , int n)
{
    int     i, j;
     char *temp    ;                          方法 2：char temp[20];
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n-i-1; j++)
            if (strcmp(v[j], v[j+1]) >= 0)
            {    temp =v[j]    ;              方法 2：strcpy(temp,v[j]);
                 v[j] =v[j+1]       ;                    strcpy(v[j] ,v[j+1]);
                 v[j+1] =temp       ;                    strcpy(v[j+1] ,temp);
            }
}
```

五、输出程序运行结果（25 分）-----每小题 5 分

(1)

```
#include <stdio.h>
void swap(int **px,int **py);

void main(void)
{
      int a,b,c;
      int *pa=&a,*pb=&b,*pc=&c;

      scanf("%d%d%d",pa,pb,pc);
      printf("before:%d, %d, %d\n",*pa,*pb,*pc);
      if(*pa>*pb)          swap( &pa , &pb);
      if(*pb>*pc)          swap( &pb , &pc);
      if(*pa>*pb)          swap( &pa , &pb);
      printf("after:%8d%8d%8d\n",*pa,*pb,*pc);
}

void swap(int **px,int **py)
{
      int *temp;
      temp =*px;
      *px =*py;
      *py =temp;
```

```
}
```
输入：7　22　9

输出结果：

before：7，22，　9

after：　　7　　9　　22

——未写 before 和 after 扣 1 分

(2)

```
#include <stdio.h>
void    main()
{
    int     i, j, min;
    int     x = 0, y= 0;
    int     a[3][3] = {     {1, 2, 3}, {2, -3, 4},{7, 4, 7}         };

    min = a[0][0];
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
        {
            if (a[i][j] >= min)
            {   min = a[i][j];
                x = i+1;
                y = j+1;     }
        }
    printf("min=%d at (x,y):(%d,%d)\n", min, x, y);
}
```

输出结果：

min=7 at (x,y):(3,3)

——直接写 7 和 3,3 等字样，即写对数值但未写其他字符时扣 1 分

(3)

```
#include <stdio.h>
#include <stdlib.h>

void hanshu(void)
{
      static int n;
      n - =20;
      printf("hanshu:n=%d\n",n);
}

void     main(void)
{
      int n=100;
      hanshu();
      for( ;n>=60;n-=20 )
      {     hanshu( );
            printf("n=%d\n",n);     }
```

```
}
```
输出结果：
```
hanshu:n=-20
hanshu:n=-40
n=100
hanshu:n=-60
n=80
hanshu:n=-80
n=60
```

(4)
```
#include <stdio.h>
struct Key
{
    char    *keyword;
    int     keyno;
};

void        main()
{
    struct Key     kd[] = { {"are", 123}, {"your", 456}, {"my", 789} };
    struct Key     *p;
    int       a;
    char      *pchr;

    p=kd;
    a = (++p)->keyno;
    pchr = p->keyword++;
    printf("p->\"%s\",a=%d\n", pchr, a);

    a =    ++p->keyno;
    pchr = p++->keyword;
    printf("p->\"%s\",a=%d\n", pchr, a);

    a = p->keyno++;
    pchr = p->keyword;
    printf("p->\"%s\",a=%d\n", pchr, a);
}
```
输出结果：
```
p->"your",a=456     ----2 分
p->"our",a=457      ----1 分
p->"my",a=789         ---2 分
```

(5)

```
#include <stdio.h>
#include <stdlib.h>
char * process(char *,char );

void main(void)
{
    char ch[100], Dr;

    puts("Pls enter a String (no more than 100 characters):");
    gets(ch);

    puts("\nPls enter a character to be process:");
    fflush(stdin);   //clear buffers of standard input device
    Dr =getchar();
    process(ch,Dr);

    puts(ch);
    system("pause");
}

char * process(char *s,char del)
{
    char *read,*write;

    read=write=s;

    while(*read)
    {
        if(*read==del)
            read++;
        else
            *write++ =*read++;
    }
    *write='\0';
    return s;
}
```

输入：test room is at D9B100&A200 rooms!

输出结果：

Pls enter a String (no more than 100 characters)://不写这一行输出建议也给分

//同上

Pls enter a character to be process://同上

test room is at D9B1&A2 rooms!

六、编写程序（35 分）

注意：不得使用全局变量，注意程序结构

(1) 编程计算 1-21 的阶乘，每行输出 3 个数的阶乘左对齐打印输出（9 分）

#include<stdio.h>----------2 分

一种方法：

int main( ) ----------主函数框架 2 分

{

    double factorial=1.0; //如果用 unsigned long、long、float 等都会有溢出问题发生

```
        ——采用低类型，扣 3 分
    int i;

    for(i=1;i<=21;i++)
    {
        factorial *=i;
        printf("%3d!=%20.0lf",i,factorial);
        if(i%3==0)    //注意实型变量不能参与%（求余）运算
            printf("\n");                -----不能表达每行 3 个阶乘扣 2 分
    }
    return 0;
}
```

第二种方法：

```
int main( )
{
    double factorial=1.0;
    int i;

    for(i=1;i<=21;i++)//外层循环
    {
        factorial =1.0;//如果放到外层循环前面，则为初始化位置放错，导致阶乘不准确
        for(int j=1;j<=i;j++)
            factorial *=j;
        printf("%3d!=%20.0lf",i,factorial);
        if(i%3==0)    //注意实型变量不能参与%（求余）运算
            printf("\n");
    }

    return 0;
}
```

第三种方法，使用函数+static 局部变量

```
double factor(double x);

int main( )
{
    double factorial=1.0;
    int i;

    for(i=1;i<=21;i++)
    {
```

```
        factorial=factor(i);
        printf("%3d!=%20.0lf",i,factorial);
        if(i%3==0)
            printf("\n");
    }

    return 0;
}

double factor(double x)
{
        static double factorial =1.0;
        factorial *=x;

        return factorial;
}
```

(2) 编写函数完成 4*4 矩阵的转置，并计算出转置后的矩阵中的最大值以及其行列编号，main 函数负责输入矩阵数据，调用函数 process 同时完成转置并得到矩阵的最大值和行列位置信息，在 main 函数中打印输出有关信息。（9 分）

```
#include <stdio.h>                                          ----------------------0.5 分
void process(int (*p)[4],int n,int *pmax,int *prow,int *pcolumn);
//或者 void process(int p[ ][4],int n,int *pmax,int *prow,int *pcolumn);---0.5 分
int main( )                          ------------4 分
{
    int matrix[4][4],i,j;
    int max,row,column;

    for(i=0;i<4;i++)
        for(j=0;j<4;j++)
            scanf("%d",&matrix[i][j]);

    process(matrix,4,&max,&row,&column);

    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
            printf("%d ",matrix[i][j]);
        printf("\n");
    }

    printf("after transposition,max=%d,lies in row %d and column %d\n",max,row,column);
```

```
    return 0;
}
//下面函数头定义中也可以采用 int p【】[4]
void process(int (*p)[4],int n,int *pmax,int *prow,int *pcolumn)--------指针接口形参 1 分
{---------------------------------------函数定义总 4 分
    int i,j;

    for(i=0;i<n-1;i++)//先转置
        for(j=i+1;j<n;j++)
        {
            int temp;
            temp =p[j][i];
            p[j][i]=p[i][j];
            p[i][j] =temp;
        }                 -------------------------转置 2 分

    *pmax =p[0][0];
    for(i=0;i<4;i++)//再寻找最大值和位置信息
        for(j=0;j<4;j++)
            if(p[i][j]>*pmax)
            {
                *pmax =p[i][j];
                *prow =i;
                *pcolumn =j;
            }                              ----------寻找最值 1 分
}
另一种方法：process 改为
void process(int (*p)[4],int n,int *pmax,int *prow,int *pcolumn)
{
    int i,j;

    *pmax =p[0][0];
    for(i=0;i<n;i++)//思路：转置的同时寻找最大值和位置信息，降低循环次数
        for(j=0;j<n;j++)
        {
            if(i<j)
            {
                int temp;
                temp =p[j][i];
                p[j][i]=p[i][j];
                p[i][j] =temp;
            }
```

```
            if(p[i][j]>*pmax)
            {
                *pmax =p[i][j];
                *prow =i;
                *pcolumn =j;
            }
        }
}
```

以上传指针的参数设计，也可以改为传引用

```
void process(int (*)[4],int ,int &,int &,int &);//对应于上面的指针型参数改为C++的引用型参数，起到同样回传修改结果给主调函数的作用
int main(    )
{
  。。。。
  process(matrix,4,max,row,column);//直接用被引用的变量做实参，不需&取地址运算符
  。。。。
   return 0 ;
}
void process(int (*p)[4],int n,int &refmax,int &refrow,int &refcolumn)
{
    int i,j;

    refmax =p[0][0];
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
        {
            if(i<j)
            {
                int temp;
                temp =p[j][i];
                p[j][i]=p[i][j];
                p[i][j] =temp;
            }
            if(p[i][j]>refmax)
            {
                refmax =p[i][j];
                refrow =i;
                refcolumn =j;
            }
        }
}
```

(3) 编写一个函数，void fun(char *str, char *substr1, char *subtr2 , …)实现将字符串 str 中

的子串 substr1 替换为子串 substr2，并同时统计替换后 str 字符串长度，要求：接口定义中已有部分不许修改，但允许在…处自由添加参数。替换举例：如将字符串“abcedfrbcxybc”中的字串 1“bc”替换为子串 2“ghi”后为“aghiedfrghixyghi”，主函数 main 完成字符串和子串的输入，调用函数 fun 得到替换后的字符串，最后在主函数中输出新串及由 fun 函数计算出来的长度 16。如果子串 substr1 不在 str 中，则 fun 函数只统计字符串长度。（9 分）

```
#include <string.h>
#include <stdio.h>

void fun(char *str, char *substr1, char *subtr2 , int *plen);//第四个参数可以 int& reflen
                    //对应 main 中调用格式：fun( string, str1, str2, length )；
                    //fun 函数体中最后一条执行语句改为：reflen=strlen(str);
--------------------头文件和原型申明共 2 分
void main( void )------------------------共 3 分
{
    char string[100] = "abcedfrbcxybc";
    char str1[ ] =     "bc";
    char str2[ ] =     "ghi";
    int    length;
    printf( "before:%s and total size:%d\n", string ,strlen(string));
    fun( string, str1, str2, &length );
    printf( "after:%s and total size: %d\n", string, length );
}

void fun(char *str, char *substr1, char *substr2 , int *plen)---接口定义 1 分
{
    char *pfound=NULL;
    int sub1len,sub2len;

    sub1len =strlen(substr1);
    sub2len =strlen(substr2);
    pfound  =  strstr(str,substr1);//调用 strstr 库函数查找主串 str 中是否存在 substr1 子串，返回 NULL 指针（表明未找到）或指向找到的子串存储空间入口的指针

    while (pfound != NULL)----------------查找&替换 2 分
    {
        char temp[100];

        strcpy(temp,pfound+sub1len);//将对应子串后面的字符内容搬移到 temp
        strcpy(pfound,substr2);//通过字符串拷贝，用子串 2 替换掉子串 1 内容
        strcat(str,temp);//strcpy(pfound+sub2len,temp);//strcat(pfound,temp);
```

```
            //上面三种写法实现将 temp 串追加到替换后的子串 2 串结尾符\0 后面

            pfound = strstr(pfound +sub2len, substr1);
        }
        *plen =strlen(str);----------------------1 分

        return ;
    }
```

(4) 设有 N 个参加复试的考生，考生信息包括编号、姓名、性别和 4 门功课的成绩，main 函数负责输入各单科录取线及考生信息，并统计每个学生的总分。编写函数实现：（1）sort 函数实现按总分进行排序（降序）；（2）search 函数找出至少有一门课低于单科录取线的所有考生信息；（3）print 函数实现按总分从高到低打印排名前 20%的单科达到要求的考生信息；要求：排序、查找和打印输出分别用函数实现。（8 分）

```
#include <stdio.h>
#define N     6
typedef struct
{
    long ID;
    char name[20];
    char sex;
    int score[4];
    int status;//记录状态。0：所有单科达标  1：单科未达标
    int total;
} STUDENT;
-----------------------------------以上 1 分
void bubblesort(STUDENT data[ ],int n);
int search(STUDENT data[ ],int students,int danke[ ],int courses,STUDENT select[ ]);
void print(STUDENT stu[ ],int students,double percent);//调用前先执行降序操作
-----------------------3 个原型申明 1 分
int main( )                 -------------------------2 分
{
    STUDENT stu[N];

    int i,j;
    int danke[4];
    printf("录入并统计每个学生的总分并记录:\n");
    for(i=0;i<N;i++)
    {
        printf("No%d:\n",i+1);
        scanf("%d",&stu[i].ID);
        scanf("%s",stu[i].name);
```

```
            fflush(stdin);
            scanf("%c",&stu[i].sex);
            stu[i].total=0;
            for(j=0;j<4;j++)
            {
                int abc;
                scanf("%d",&abc);
                stu[i].total +=(stu[i].score[j]=abc);
            }
        }

        printf("录入单科录取线:");
        for(i=0;i<4;i++)
        scanf("%d",&danke[i]);

        STUDENT select[N];

        int found =search(stu,N,danke,4,select);
        if(found>=1)
        {
            printf("共有%d 个学生未达到单科线要求!\n",found);
            for(i=0;i<found;i++)
            {
                    printf("%3d%-20s%3d——%3d%3d%3d%3d\n",select[i].ID,
                    select[i].name,select[i].total,select[i].score[0],select[i].score[1],
                    select[i].score[2],select[i].score[3]);
            }
        }
        else
            printf("所有学生通过复试线\n");

        bubblesort(stu,N);//降序排列总分

        puts("打印总分排名前 20%的单科达标学生信息:");
        print(stu,N,0.2);
        return 0;
    }

    void bubblesort(STUDENT data[ ],int n)                    ------2 分
    {
        int i,j;
```

```
    for(i=1;i<=n-1;i++)
        for(j=1;j<=n-i;j++)
            if(data[j-1].total < data[j].total)
            {
                STUDENT  temp;
                temp     =data[j];
                data[j]  =data[j-1];
                data[j-1]=temp;
            }
}

int search(STUDENT data[ ],int students,int danke[ ],int courses,STUDENT select[ ])—1 分
{
    int i,j,k=0;

    for(i=0;i<students;i++)
    {
        for(j=0;j<courses;j++)
        {
            if(data[i].score[j]<danke[j])
            {
                data[i].status=1;
                select[k++]=data[i];//可以不用参数 select，只需记录状态，在主调函数
中遍历学生数组 stu 时判断状态，找出所有 status 为 1 的学生，但效率低
                break;
            }
            else
                data[i].status=0;
        }
    }
    return k;
}

void print(STUDENT stu[ ],int students,double percent)//调用前先执行降序操作----1 分
{
    int i;
    int prints;
    double value;
    value =students*percent
    if((int)value ==value)
        prints =(int)(value);
    else
```

```
        prints =(int)(value)+1;

    for(i=0;i<prints;i++)
    {
        if(stu[i].status==0)
            printf("%3d%-20s%3d——%3d%3d%3d%3d\n",stu[i].ID,stu[i].name,
            stu[i].total,stu[i].score[0],stu[i].score[1],stu[i].score[2],stu[i].score[3]);
    }
}
```