

2014-2015 学年第二学期期末考试 A 卷

一、选择题(每小题 2 分, 共 30 分)

- 1、关于函数调用的说法, 错误的是()。
 - A、递归函数和普通函数一样, 都可以使用全局变量。
 - B、在递归函数中, 静态的局部变量只创建一次。
 - C、嵌套调用函数时, 先调用的函数先返回, 后调用的函数后返回。
 - D、递归调用的层数不是无限制的。
- 2、在最坏情况下, 二分查找算法的时间复杂度是()。
 - A、 $O(1)$
 - B、 $O(N)$
 - C、 $O(\log(N))$
 - D、 $O(N^2)$
- 3、对于以下结构定义, `++p->str` 中的 `++` 加在()。


```
struct {
    int len;
    char *str;
}*p;
```

 - A、指针 `p` 上
 - B、指针 `str` 上
 - C、`str` 指的内容上
 - D、语法错误
- 4、以下正确的描述是()。
 - A、预处理指令只能位于 C 源程序文件的首部。
 - B、凡是 C 源程序文件中行首以“#”标识的控制行都是预处理指令。
 - C、C 语言的编译预处理就是对源程序进行初步的语法检查。
 - D、C 语言的预处理功能是指完成宏替换和包含文件的调用。
- 5、在 `VESA_800×600×24bit` 图形模式下, 要使数组 `a` 刚好能够容纳整个屏幕信息, 则 `a` 应该定义为()。
 - A、`char a[800*600];`
 - B、`char a[800][600][24];`
 - C、`struct{char c[3];}a[800][600];`
 - D、`long a[800][600];`
- 6、以下程序段的算法时间复杂度是()。


```
for (i=1;i<=n;i++)
    x++;
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        x++;
```

 - A、 $O(N)$
 - B、 $O(N^2)$
 - C、 $O(2N)$
 - D、 $O(N^3)$
- 7、假定已定义变量 `int key`; 当要读取键盘上的 Home 或 End 键并赋值给 `key` 时, 则正确的写法是()。
 - A、`key=getchar();`
 - B、`scanf("%d",&key);`
 - C、`key=fgetc(stdin);`
 - D、`key=biostkey(0);`

8、F(x)是一个带参数的宏，欲在宏替换后达到 $1+x^2$ 的计算效果，为了保证 F(a+b)*F(c+d)对于任何取值的浮点数变量 a、b、c、d 都结果正确，那么该宏正确的定义方式是()。

- A、#define F(x) x*x+1 B、#define F(x) (x)*(x)+1
C、#define F(x) ((x) *(x)+1) D、#define F(x) (x*x+1)

9、以下关于全局变量和外部变量的说法，不正确的是()。

- A、外部变量都是全局变量。
- B、静态的全局变量只能在本文件中使用，而不能在别的源文件中使用。
- C、使用外部变量之前，需要事先声明。
- D、通过 `extern` 关键字事先声明，可以使用其他源文件的静态变量。

10、根据以下定义，表达式 w.h.d[5]的值是()。

```
struct {
    int a,b,c[8];
    struct { int d[6],f;} h;
}w={1,2,{3,4,5},{6,7},8}};
```

- A、 0 B、 2 C、 5 D、 8

11、在单链表指针为 p 的结点之后插入指针为 s 的结点，正确的操作是()。

- A、p->next=s;s->next=p->next;
B、s->next=p->next;p->next=s;
C、p->next=s;p->next=s->next;
D、p->next=s->next;p->next=s;

12、以下说法不正确的是()。

- A、头文件（.h 文件）可以被单独编译。
- B、声明函数时，可以省略形式参数的名字，但是不可省略形式参数的类型。
- C、一个程序可以由多个源文件构成，其中.c 文件通常包含一些全局变量的定义，以及一些函数的实现。
- D、头文件（.h 文件）的内容通常包括宏定义、数据类型定义、全局变量声明和函数声明等。

13、对于排序算法，下面说法正确的是()。

- A、平均情况下，选择排序算法、冒泡排序算法的时间复杂度都是 $O(N^2)$ 。
- B、平均情况下，选择排序算法、快速排序算法的时间复杂度都是 $O(N\log(N))$ 。
- C、平均情况下，归并排序算法、插入排序算法的时间复杂度都是 $O(N^2)$ 。
- D、平均情况下，归并排序算法、冒泡排序算法的时间复杂度都是 $O(N\log(N))$ 。

14、在“文件包含”预处理语句的使用形式中，当#include 后面的文件名用<>（尖括号）括起时，寻找被包含文件的方式是（ ）。

- A、仅仅搜索当前文件夹。 B、仅仅搜索源程序所在的文件夹。
C、直接按系统设定的标准方式搜索文件夹。
D、先在源程序所在的文件夹搜索，再按系统设定的标准方式搜索。

15、下列程序段的输出结果是()。

```
#define A 10
#define B (A<A+2)-2
printf("%d",B*2);
```

- A, 0 B, 2 C, -2 D, -3

二、问答题(共 25 分)

- 1、下面的程序段定义了两个结构类型和一个指针变量，试图将第二个学生的生日赋值为 1995，但不正确，请问错在何处，如何修正？（3 分）

```
struct data{
    int day,month,year;
};
struct student{
    char name[20];
    long num;
    struct data birthday;
};
struct student *p = (struct student *)malloc(10*sizeof(struct student));
*(p+1)->birthday->year = 1995;
```

- 2、下面的递归函数在实现逆序输出一个正整数的各位数字时有错误，请指出并修正。（3 分）

```
void showdigits( unsigned int n ){
    while( n>0 ){
        printf("%d",n%10);
        showdigits(n/10);
    }
}
```

- 3、以下程序对字符串进行加密，加密算法是每一字符与 15 异或，实现低四位取反，高四位保持不变。请问是否正确，若不正确请说明原因，并修正。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define LENGTH 81
int main(void){
    char *p;
    p = (char*)malloc(sizeof(char)*LENGTH);
    strcpy(p, "I am a student!");
    while (*p++) *p = *p^15;
    free(p);
}
```

4、下面是归并排序的核心模块：

```
void SortIntegerArray(int array[], int n){
    int i,n1,n2,*arr1,*arr2;
    if(n>1){
        n1=n/2;
        n2=n-n1;
        arr1=NewArray(n1,int);
        arr2=NewArray(n2,int);
        for(i=0;i<n1;i++) arr1[i]=array[i];
        for(i=0;i<n2;i++) arr2[i]=array[n1+i];
        SortIntegerArray(arr1,n1);
        SortIntegerArray(arr2,n2);
        Merge(array,arr1,n1,arr2,n2);
        FreeBlock(arr1);
        FreeBlock(arr2);
    }
}
```

对(25,50,15,35,80,85,20,40,36,70)进行归并排序，分别写出前 4 次调用 Merge 函数得到的合并数据情况。（4 分）

5、试说明数组与链表的区别，以及各自有什么优缺点。（6 分）

6、在 C 程序的多文件组织方式下，请问如何防止头文件被多重包含，具体做法是？（5 分）

三、程序填空题(每空 2 分, 共 30 分)

1、已知有理数结构类型的定义为:

```
typedef struct{
    int numerator;//分子
    int denominator;//分母, 不能是 0
}RATIONAL;
```

则有有理数加法函数是 (不考虑化简):

```
RATIONAL AddRational(RATIONAL a, RATIONAL b){
    RATIONAL t;
    _____(1)_____;
    t.denominator = a.denominator*b.denominator;
    return t;
}
```

假设有有理数化简函数是: RATIONAL SimplifyRational(RATIONAL a), 那么求 n 个有理数平均值的函数是 (考虑化简):

```
RATIONAL AverageRationals(RATIONAL a[],int n){
    int i;
    RATIONAL t = a[0];
    for(i=1;i<n;i++)
        _____(2)_____;
    t.numerator /=n;
    _____(3)_____;
    return t;
}
```

2、下面的程序根据读入的一个数字字符串, 计算射击成绩。这样的数字字符串 (由 '0'-'9' 组成) 中的每个数字表示射击的环数成绩 (约定最后两个数字不会是 8 和 9)。用下面的积分规则计算成绩: 8 环以下只加入该环数, 8 环和 9 环除了加入该环数以外, 还可以加上奖励分数: 8 环的奖励分是接下来两发射击环数的较大者, 9 环的奖励分是接下来两发射击环数 (较大者记为 m, 较小者记为 n) 的和值与差值的积 $((m+n)*(m-n))$ 。例如: 程序若输入 123, 则输出 grade=6; 输入 1823, 输出 grade=17; 输入 1923, 输出 grade=20; 输入 18923, 输出 grade=37。

```
#include <stdio.h>
#define MIN(a,b) a>b?b:a
#define MAX(a,b) a>b?a:b
#define MULT(x,y) _____(4)_____ //宏定义
#define CTOD(c) _____(5)_____ //宏定义
int main(void) {
    char s[80], *p = s, c;
    int grade = 0, g, m, n;
    gets(s);
    while( c=*p++ ){
        switch( g=CTOD(c) ){
            case 0: case 1: case 2: case 3: case 4: case 5: case 6: case 7:
                grade += g; break;
```

```

        case 8: m = MAX(CTOD(*p),CTOD(*(p+1)));
                _____(6)_____;
                break;
        case 9: m = MAX(CTOD(*p),CTOD(*(p+1)));
                n = MIN(CTOD(*p),CTOD(*(p+1)));
                grade += g + MULT( m+n,m-n );
                break;
    }
}
printf("grade=%d\n",grade);
return 0;
}

```

- 3、在 32bit 图形模式下使用二维结构数组画点，以(300,200)为左上角，以(500,350)为右下角画一个实心的白色矩形。

```

#include<graphics.h>
typedef struct{
    char blue;
    char green;
    char red;
    char zero;
}RGB;
void main(){
    int driver=0, mode=VESA_1024×768×_____(7)_____bit;
    int x,y;
    RGB c = {0xFF, 0xFF, 0xFF, 0x00};
    RGB (*p)[1024];
    initgraph(&driver, &mode, "");
    p = (RGB (*)[1024])_vp;
    for(y=200;y<=350;y++){
        for(x=300;x<=500;x++){
            _____(8)_____;
        }
    }
    getchar();
    _____(9)_____;
}

```

- 4、小明编写了一个选择排序程序，一共有三个文件：sort.h, program.c, sort.c。为了方便查看程序是否正确运行，小明希望输出排序过程中的中间结果（即：在每一次选择操作之后，将数组的内容都打印出来）。一旦测试正确之后，他又希望能方便地关闭冗长的打印。因此，他利用了 C 语言的条件编译功能，打开或关闭输出。

这是一个多文件的程序，请完成空白处的语句，完成选择排序算法，使得程序能够正常编译链接，而且输出排序的中间结果。

文件 prog.c 的内容如下：

```
#include <stdio.h>
#include<sort.h>
int main(){
    int n, a[100], k;
    printf("Input the number of the data(<100): ");
    scanf("%d",&n);
    for(k=0;k<n;k++)
        scanf("%d",&a+k);
    selectsort(a, n);
    return 0;
}
```

文件 sort.h 的内容如下:

(10)

```
void selectsort(int a[], int n);
void swap(int *a, int *b);
```

文件 sort.c 的内容如下:

(11)

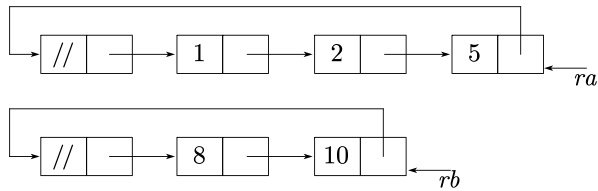
```
void selectsort(int a[], int n){
    int i, index, k;
    for(k=0;k<n-1;k++){
        index=k;
        for(i=k+1;i<n;i++)
            if(a[i]<a[index]) index=i;
        _____;
        EchoData(a,n);
    }
}
```

(12)

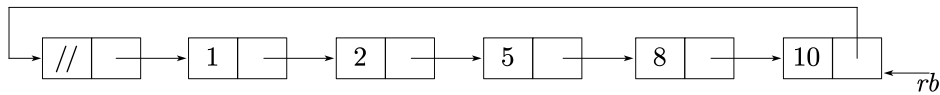
```
void swap(int *a, int *b){
    int temp = *a;
    *a = *b; *b = temp;
}
#ifdef _ECHO
void EchoData(int a[], int n){
    int k;
    for(k=0;k<n;k++) printf("%d",a[k]);
    printf("\n");
}
#else
void EchoData(char s[])
{
}
#endif
```

5、以下函数实现如图所示的功能：将用尾指针 ra、rb 表示的两个环形链表进行连接，返回连接后的环形链表的尾指针 rb。

连接前：



连接后：



```
typedef int datatype;
typedef struct node{
    datatype data;
    struct node *next;
}linklist;
.....
linklist *CONNECT(linklist *ra, linklist *rb){
    _____ (13) _____;
    p = ra->next;
    ra->next = _____ (14) _____;
    free(rb->next);
    rb->next = _____ (15) _____;
    return(rb);
}
```

四、算法设计（15 分）

1、冒泡法排序的改进版本是：当某轮次的比较结果不曾发生任何交换，就表明所有剩下的数据已经完全有序了，此时可以提前结束排序。请写出以下改进的冒泡排序函数 BubbleSort2，函数 BubbleSort2 的原型如下：

```
void BubbleSort2(int a[], int n); (5 分)
```


2、假设系统提供了 3 个功能函数：

(1) 函数 DrawTri 在屏幕上画一个三角形，声明如下：

```
void DrawTri(float u[2], float v[2], float w[2], int color);
```

其中，参数 u, v, w 表示三角形顶点的平面坐标；color 表示画笔的颜色，color=0 表示黑色，color=1 标识白色。

(2) 函数 TriArea，计算三角形面积，声明如下：

```
float TriArea(float u[2], float v[2], float w[2]);
```

其中，参数 u, v, w 表示三角形顶点的平面坐标。

(3) 函数 CalcMidPoint，计算线段中点，声明如下：

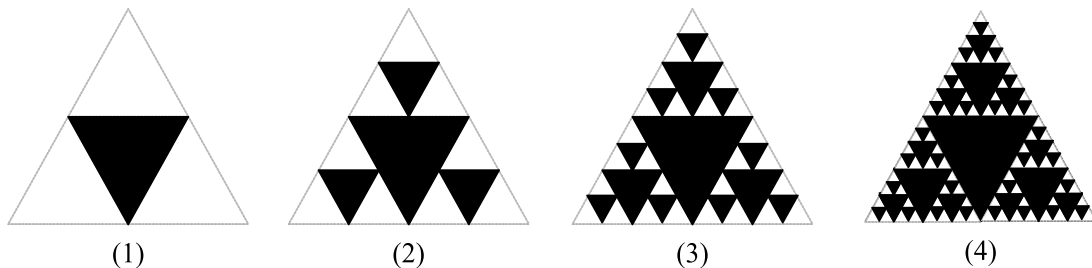
```
void CalcMidPoint(float a[2], float b[2], float m[2]);
```

其中，参数 a, b 表示线段的两个端点的平面坐标，参数 m 保存中点的坐标。

请发现图形(1)~(4)的规律，设计绘制类似图形的算法。要求：(1) 简要描述算法思路；(2) 利用上述功能函数写成相应的实现函数 Draw。函数 Draw 的原型如下：

```
void Draw(float u[2], float v[2], float w[2], float t);
```

所绘制的图形中，最小的三角面积不大于给定的阈值参数。(10 分)



注意：图(1)中黑色三角形顶点位于大三角形的中点，其他的以此类推。

2014-2015 学年第二学期期末考试 A 卷参考答案

一、选择题(每小题 2 分, 共 30 分)

1、【正解】C

【解析】嵌套调用函数时, 先调用的函数后返回, 后调用的函数先返回, C 错误。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

2、【正解】C

【解析】在最坏情况下, 二分查找算法的时间复杂度是 $O(\log(N))$, 选 C。

【考点延伸】《考试宝典》时间复杂度

3、【正解】B

【解析】 \rightarrow 的优先级比 $++$ 高, 先进行 $p \rightarrow str$ 操作再进行 $++$ 操作, 即 $++(p \rightarrow str)$, 因此 $++$ 加在指针 str 上, 选 B。

【考点延伸】《考试宝典》专题二 2.2 运算符和表达式、专题六 6.2 指针变量

4、【正解】B

【解析】宏定义和文件包含指令只要在函数使用宏或调用包含文件中的变量、函数之前写好, 一般在文件首部, 但条件编译通常在代码中间, A 错误; C 语言的编译预处理内容不只是语法检查, 还有语义检查, 生成目标代码 obj 等工作, C 错误; C 语言的预处理功能包括宏定义、包含文件调用和条件编译, D 错误; 选 B。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

5、【正解】C

【解析】选项 D 用 $long$ 作为存储类型, 数组 a 的存储大小为 $800*600*8*8=800*600*64bit$; 选项 A、B、C 用 $char$ 作为存储类型, 数组 a 的存储大小分别为 $800*600*1*8=800*600*8bit$ 、 $800*600*24*8=800*600*192bit$ 、 $800*600*3*8=800*600*24bit$ 。题目要求刚好能够容纳整个屏幕信息, 选 C。

【考点延伸】《考试宝典》专题二 2.1 常量和变量

6、【正解】B

【解析】给定程序中有二重循环, 观察二重循环的边界条件, 可知时间复杂度为 $O(N^2)$ 。

【考点延伸】《考试宝典》时间复杂度

7、【正解】D

【解析】 $bioskey()$ 是 C 语言中的一个函数, 其功能是直接使用 BIOS 服务的键盘接口。函数原型为: $int bioskey(int cmd)$; 当 cmd 是 0 时, $bioskey()$ 返回下一个在键盘键入的值。要读取键盘上的 Home 或 End 键并赋值给 key , 则应使用语句 $key=bioskey(0)$; 选 D。

【考点延伸】《考试宝典》 $bioskey()$ 函数

8、【正解】C

【解析】若使用选项 A 中的宏, 则 $F(a+b)*F(c+d)=a+b*a+b+1*c+d*c+d+1$, 错误; 若使用选项 B 中的宏, 则 $F(a+b)*F(c+d)=(a+b)*(a+b)+1*(c+d)*(c+d)+1$, 错误; 若使用选项 C 中的宏, 则 $F(a+b)*F(c+d)=((a+b)*(a+b)+1)*((c+d)*(c+d)+1)$, 正确; 若使用选项 D 中的宏, 则 $F(a+b)*F(c+d)=(a+b*a+b+1)*(c+d*c+d+1)$, 错误; 因此, 选 C。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

9、【正解】D

【解析】静态变量分为静态局部变量和静态全局变量, 静态局部变量只能为所在函数所使用, 其他的文件无法使用; 静态全局变量是显式用 $static$ 修饰的全局变量, 作用域是声明此变量所在的文件, 其他的文件即使用 $extern$ 声明也不能使用。因此, D 错误。

【考点延伸】《考试宝典》专题二 2.1 常量和变量

10、【正解】A

【解析】根据代码段可以看出, $w.h.d=\{6,7\}$, 由于 int 数组默认数值初始化为 0, 因此未赋值的

w.h.d[5]=0, 选 A。

【考点延伸】《考试宝典》专题九 9.1 结构的定义与使用

11、【正解】B

【解析】首先让 s 的 next 指针指向 p 的 next 指针, 再将 p 的 next 指针指向 s, 即
`s->next=p->next;p->next=s;`选 B。

【考点延伸】《考试宝典》专题九 9.7 用指针处理链表

12、【正解】A

【解析】头文件.h 的内容被插入到源文件.c 中, 作为源文件.c 的内容被编译。但是头文件.h 文件本身不直接参加编译, 所以也不存在被单独编译的情况。

【考点延伸】《考试宝典》C 语言编译

13、【正解】A

【解析】平均情况下, 选择排序算法、冒泡排序算法、插入排序算法的时间复杂度都是 $O(N^2)$, 归并排序算法、快速排序算法的时间复杂度都是 $O(N\log(N))$ 。选 A。

【考点延伸】《考试宝典》排序算法、时间复杂度

14、【正解】C

【解析】当#include 后面的文件名用<> (尖括号) 括起时, 预处理程序在标准目录下查找指定的文件, 选 C。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

15、【正解】D

【解析】先将 B 替换为 $(A < A + 2) - 2$, 表达式 $B * 2$ 变为 $(A < A + 2) - 2 * 2$; 再将 A 替换为 10, 表达式变为 $(10 < 10 + 2) - 2 * 2$; 最终输出结果为 -3, 选 D。

【考点延伸】《考试宝典》专题二 2.2 运算符和表达式

二、问答题(共 25 分)

1、【解析】错在最后一个语句 `*(p+1)->birthday->year = 1995;`, `*(p+1)` 表示指针 p 加一后所指向的数组元素, 它不是指针, 因此不能用 `->` 而应该使用 `.`。修正方法如下:

`(p+1)->birthday.year=1995;`或 `*(p+1).birthday.year=1995;`或 `p[1].birthdary.year=1995;`

【考点延伸】《考试宝典》专题六 6.5 指针数组与数组指针

2、【解析】题中的代码错误地杂糅了递归和 while 两种实现方式。修正方法有如下几种:

(1)while 改成 if;

(2)在 `showdigits(n/10)` 后加 `break`;

(3)去掉 while 循环, 在 `printf("%d", n%10);` 之前加 `if(n==0)return;` 或 `if(n<=0)return;`

【考点延伸】《考试宝典》专题四 4.1 while 语句

3、【解析】语句 `free(p);` 错误, 此时指针 p 并没有指向字符串的起始地址, 有部分申请的内存没有释放。修正方式如下: 增加语句 `char *q;`, 增加一个指针; `q=p;`, 使 q 指向字符串的起始地址并保持不变, `free(p)` 改为 `free(q);`

循环条件 `(*p++)` 错误, 这个循环条件无法对字符串的每一位进行操作, 应该改为 `(*p)`, 并在循环体内加语句 `p++`;

【考点延伸】《考试宝典》专题四 4.1 while 语句、专题六 6.4 字符串的指针

4、【解析】25,50

35,80

15,35,80

15,25,35,50,80

【考点延伸】《考试宝典》归并算法

5、【解析】数组:

- 事先定义固定长度的数组;
- 在数组元素个数不确定时, 可能会发生浪费内存空间的情况;
- 缺点: 插入和删除元素需要大量的数据移动;

- 优点：随机存取，存取任一元素只需要少量时间。

链表：

- 动态存储分配的数据结构；
- 使用链表可以节省内存，提高操作效率；
- 优点：根据需要动态开辟内存空间，比较方便地插入和删除元素（结点）；
- 缺点：不能随机存取。

【考点延伸】《考试宝典》专题五 5.1 数组、专题九 9.7 用指针处理链表

6、【解析】所有头文件都应该使用#define 保护防止文件被多重包含。具体做法如下：

```
#ifndef _HEADERNAME_H
#define HEADERNAME_H
...//(头文件内容)
#endif
```

【考点延伸】《考试宝典》专题八 8.4 预处理命令

三、程序填空题(每空 2 分，共 30 分)

1、【正解】(1)t.numerator=a.numerator*b.denominator+a.denominator*b.numerator

(2)t=AddRational(t,a[i])

(3) t=SimplifyRational(t)

【解析】根据在不考虑化简的情况下将两个分数相加的运算经验可知，第一空应填

t.numerator=a.numerator*b.denominator+a.denominator*b.numerator；先对 n 个有理数进行求和，第二空应填 t=AddRational(t,a[i])；在求得平均值后应对所得的平均数进行简化，第三空应填 t=SimplifyRational(t)。

【考点延伸】《考试宝典》专题九 9.1 结构体的定义与使用

2、【正解】(4)(x)*(y)

(5)((c>='1')&&(c<='9')?c-'0':0)

(6)grade+=g+m

【解析】第一个宏 MULT(x,y)的含义是 x*y，为了保证在任何 x, y 情况下，替换结果都正确，第四空应填(x)*(y)；第二个宏 CTOD(c)的含义是将数字字符 c 转换为数字，第五空应填((c>='1')&&(c<='9')?c-'0':0)；打中 8 环，除了在成绩中加入该环数外，还有奖励分，因此第六空应填 grade+=g+m。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

3、【正解】(7)32

(8)p[y][x]=c

(9)closegraph()

【解析】在 32bit 图形模式下画图，因此第七空填 32；将矩形的每一个点画成白色，因此第八空填 p[y][x]=c；关闭图形系统，第九空填 closegraph()。

【考点延伸】《考试宝典》C 语言图形库

4、【正解】(10)#define _ECHO

(11)#include "sort.h"

(12)swap(a+k,a+index)或 swap(&a[k],&a[index])

【解析】为了利用条件编译功能，定义一个宏，第十个空填#define _ECHO；引入头文件，第十一个空填#include "sort.h"；交换数组索引为 k 和数组索引为 index 的元素，第十二个空填 swap(a+k,a+index)或 swap(&a[k],&a[index])。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

5、【正解】(13)linklist *p

(14)rb->next->next

(15)p

【解析】定义一个新指针用于暂存第一个链表的头指针，因此第十三空应填 linklist *p；将第一个链表的尾元素指向第二个链表的头元素，第十四空应填 rb->next->next；将第二个链表的尾元素指向第一个链表的头元素，因此第十五空应填 p。

【考点延伸】《考试宝典》专题九 9.7 用指针处理链表

四、算法设计（15 分）

1、【解析】

```
void BubbleSort2(int a[],int n){
    int i,j,flag,tmp;
    for( i=n;i>0;i-- ){
        flag = 0;
        for( j=0;j<i;j++ )
            if( a[j]>a[j+1] ){
                flag = 1;
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp; //swap(&a[j], &a[j+1]);
            }
        if(!flag)break;
    }
}
```

【考点延伸】《考试宝典》冒泡排序

2、【解析】采用递归的方法绘制图形。首先连接待绘制三角形的中点，将其细分为 4 个小三角形；然后将中间的小三角形绘制为黑色；最后对其余的三个小三角形进行递归处理，直至小三角形的面积小于给定的阈值。

```
void Draw(float u[2], float v[2], float w[2], float t){
    float a[2], b[2], c[2];
    if( TriArea(u,v,w)<=t ){
        DrawTri(u,v,w,1);
        return;
    }
    CalcMidPoint(u,v,a);//计算中点
    CalcMidPoint(u,v,b);//计算中点
    CalcMidPoint(w,u,c);//计算中点
    DrawTri(a,b,c,0);//中间的黑色三角形
    Draw(u,a,c,t);//递归绘制白色三角形
    Draw(a,v,b,t);//递归绘制白色三角形
    Draw(c,b,w,t);//递归绘制白色三角形
}
```

或

```
void Draw(float u[2], float v[2], float w[2], float t){
    float a[2], b[2], c[2];
    CalcMidPoint(u,v,a);//计算中点
    CalcMidPoint(u,v,b);//计算中点
    CalcMidPoint(w,u,c);//计算中点
    if( TriArea(u,v,w)>t ){
        DrawTri(a,b,c,0);//中间的黑色三角形
        Draw(u,a,c,t);//递归绘制白色三角形
        Draw(a,v,b,t);//递归绘制白色三角形
        Draw(c,b,w,t);//递归绘制白色三角形
    }
    else DrawTri(u,v,w,1);
}
```

【考点延伸】《考试宝典》专题八 8.1 函数的定义与声明、8.2 函数的调用